Lecture Notes in Computer Science 7619

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Jaakko Hollmén Frank Klawonn
Allan Tucker (Eds.)

Advances in Intelligent Data Analysis XI

11th International Symposium, IDA 2012
Helsinki, Finland, October 25-27, 2012
Proceedings

Springer

Volume Editors

Jaakko Hollmén
Aalto University School of Science
Department of Information and Computer Science
P.O. Box 15400, 00076 Aalto, Finland
E-mail: jaakko.hollmen@aalto.fi

Frank Klawonn
Ostfalia University of Applied Sciences
Department of Computer Science
Salzdahlumer Straße 46/48, 38302 Wolfenbüttel, Germany
E-mail: f.klawonn@ostfalia.de

Allan Tucker
Brunel University, School of Information Systems
Computing and Mathematics
Uxbridge, Middlesex UB8 3PH, UK
E-mail: allan.tucker@brunel.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-34155-7 e-ISBN 978-3-642-34156-4
DOI 10.1007/978-3-642-34156-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012949063

CR Subject Classification (1998): H.3, H.4, I.2, F.1, H.2.8, J.3, I.4-5

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

© Springer-Verlag Berlin Heidelberg 2012
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

We are pround to present the proceedings of the 11th International Symposium on Intelligent Data Analysis, which was held during October 25–27 in Helsinki, Finland. The series started in 1995 and was held biennially until 2009. In 2010, the symposium re-focussed to support papers that go beyond established technology and offer genuinely novel and game-changing ideas, while not always being as fully realized as papers submitted to other conferences. IDA 2012 continued this approach and included an important and still emerging class of problems: the analysis of data from networked digital information systems such as mobile devices, remote sensors, and streaming applications.

The symposium seeks first-look papers that might elsewhere be considered preliminary, but contain potentially high-impact research. The IDA Symposium, which is A-ranked according to ERA, is open to all kinds of modelling and analysis methods, irrespective of discipline. It is an interdisciplinary meeting that seeks abstractions that cut across domains.

IDA solicits papers on all aspects of intelligent data analysis, including papers on intelligent support for modelling and analyzing data from complex, dynamical systems. IDA 2012 particularly encouraged papers about:

Novel applications of IDA techniques to, for example:

- Networked digital information systems
- Novel modes of data acquisition and the associated issues
- Robustness and scalability issues of intelligent data analysis techniques
- Visualization and dissemination of results

Intelligent support for data analysis goes beyond the usual algorithmic offerings in the literature. Papers about established technology were only accepted if the technology was embedded in intelligent data analysis systems, or was applied in novel ways to analyzing and/or modelling complex systems. The conventional reviewing process, which favors incremental advances on established work, can discourage the kinds of papers that IDA 2012 has published. The reviewing process addressed this issue explicitly: referees evaluated papers against the stated goals of the symposium, and any paper for which at least one Program Chair advisor wrote an informed, thoughtful, positive review was accepted, irrespective of other reviews.

We were pleased to have a very strong program. We received 88 submissions from people in 24 countries on five continents.

As in IDA 2011, we included a poster session for PhD students to promote their work and for the first time we also introduced the use of a 2-minute video slot for all PhD posters and standard posters.

We were honored to have distinguished invited speakers at IDA 2012: Arno Siebes from the University of Utrecht in The Netherlands, who talked about ways of getting multiple good models that complement each other in the insight

they give. Paola Sebastiani from Boston University in the USA, who talked about the intelligent data analysis of human genetic data. Gavin Cawley from the University of East Anglia in the UK, who talked about avoiding over-fitting in model selection.

The conference was held at Finlandia Hall in Helsinki. We wish to express our gratitude to all authors of submitted papers for their intellectual contributions; to the Program Committee members and the additional reviewers for their effort in reviewing, discussing, and commenting on the submitted papers; to the members of the IDA Steering Committee for their ongoing guidance and support; and to the Senior Program Committee for their active involvement. We thank Richard van de Stadt for running the submission website and handling the production of the proceedings. Special thanks go to the Poster Chair, Frank Höppner, and the Frontier Prize Chairs, Elizabeth Bradley and João Gama. We gratefully acknowledge those who were involved in the local organization of the symposium: Anu Juslin, Outi Elina Kansanen, Mikko Korpela, Janne Toivola, Prem Raj Adhikari, and Olli-Pekka Rinta-Koski. The help of volunteers during the IDA 2012 symposium is also thankfully acknowledged.

We are grateful for our sponsors: Aalto University, the Helsinki Insitute for Information technology (HIIT), and Algodan — Finnish Centre of Excellence for Algorithmic Data Analysis Research. We are especially indebted to KNIME, who funded the IDA Frontier Prize for the most visionary contribution presenting a novel and surprising approach to data analysis in the understanding of complex systems.

August 2012

Jaakko Hollmén
Frank Klawonn
Allan Tucker

Organization

General Chair

Jaakko Hollmén	Aalto University, Finland

Program Chairs

Frank Klawonn	Ostfalia University of Applied Sciences, Germany
Allan Tucker	Brunel University, UK

Poster Chair

Frank Höppner	Ostfalia University of Applied Sciences, Germany

Frontier Prize Chairs

Elizabeth Bradley	University of Colorado, USA
João Gama	University of Porto, Portugal

Local Organizing Committee

Anu Juslin	Aalto University, Finland
Outi Elina Kansanen	Aalto University, Finland
Mikko Korpela	Aalto University, Finland
Janne Toivola	Aalto University, Finland
Prem Raj Adhikari	Aalto University, Finland
Olli-Pekka Rinta-Koski	Aalto University, Finland

Program Chair Advisors

Michael Berthold	University of Konstanz, Germany
Liz Bradley	University of Colorado, USA
João Gama	University of Porto, Portugal
Jaakko Hollmén	Aalto University, Finland
Frank Höppner	Ostfalia University of Applied Sciences, Germany
Joost Kok	Leiden University, The Netherlands
Xiaohui Liu	Brunel University, UK

Arno Siebes	Universiteit Utrecht, The Netherlands
Hannu Toivonen	University of Helsinki, Finland
David Weston	Imperial College, UK

Program Committee

Fabrizio Angiulli	University of Calabria, Italy
Alexandre Aussem	University of Lyon 1, France
Christian Borgelt	European Centre for Soft Computing, Spain
Henrik Boström	Stockholm University, Sweden
Andre de Carvalho	University of Sao Paulo, Brazil
Jose del Campo	University of Malaga, Spain
Bruno Crémilleux	University of Caen, France
Werner Dubitzky	University of Ulster, UK
Sašo Džeroski	Jožef Stefan Institute, Slovenia
Fazel Famili	IIT - National Research Council Canada, Canada
Ad Feelders	Utrecht University, The Netherlands
Ingrid Fischer	University of Konstanz, Germany
Élisa Fromont	University of Jean Monnet, France
Johannes Fürnkranz	Technische Universität Darmstadt, Germany
Alex Gammerman	University of London, UK
Ricard Gavaldà	Technical University of Catalonia, Spain
Kenny Gruchalla	National Renewable Energy Lab, USA
Lawrence Hall	University of South Florida, USA
Patrik Hoyer	University of Helsinki, Finland
Eyke Hüllermeier	University of Marburg, Germany
Wesley Kerr	Adknowledge Inc., USA
Rudolf Kruse	Otto von Guericke University Magdeburg, Germany
Nada Lavrač	Jožef Stefan Institute, Slovenia
Jose Lozano	The University of the Basque Country, Spain
George Magoulas	Birkbeck College, University of London, UK
Maria-Carolina Monard	University of Sao Paulo, Brazil
Giovanni Montana	Imperial College, UK
Mohamed Nadif	Descartes University of Paris, France
Detlef Nauck	British Telecom, UK
Andreas Nürnberger	Otto von Guericke University Magdeburg, Germany
Panagiotis Papapetrou	Aalto University, Finland
Mykola Pechenizkiy	Eindhoven University of Technology, The Netherlands
José-Maria Peña	Polytechnic University of Madrid, Spain
Ruggero Pensa	University of Turin, Italy
Alexandra Poulovassilis	University of London, UK

Miguel A. Prada	University of Leon, Spain
Ronaldo Prati	Universidade Federal do ABC, Brazil
Fabrizio Riguzzi	ENDIF - University of Ferrara, Italy
Stefan Rüping	Fraunhofer IAIS, Germany
Antonio Salmerón Cerdán	University of Almeria, Spain
Vítor Santos Costa	University of Porto, Portugal
Roberta Siciliano	University of Naples Federico II, Italy
Maarten van Someren	Universiteit van Amsterdam, The Netherlands
Myra Spiliopoulou	Otto von Guericke University Magdeburg, Germany
Stephen Swift	Brunel University, UK
Maguelonne Teisseire	TETIS - Irstea, France
Evimaria Terzi	Boston University, USA
Panayiotis Tsaparas	University of Ioannina, Greece
Antti Ukkonen	Yahoo! Research Barcelona, Spain
Antony Unwin	University of Augsburg, Germany
Veronica Vinciotti	Brunel University, UK
Zidong Wang	Brunel University, UK

Additional Referees

Darko Aleksovski	Dragi Kocev
Satrajit Basu	John N. Korecki
Igor Braga	Mikko Korpela
Christian Braune	Antonio LaTorre
Bertrand Cuissart	Thomas Low
Jeroen De Knijf	Gjorgji Madjarov
Ivica Dimitrovski	Christian Moewes
Fabio Fassetti	Jesus Montes
Tatiana Gossen	Santiago Muelas
Pascal Held	Ilia Nouretdinov
Dino Ienco	Luigi Palopoli
R.P. Jagadeesh Chandra Bose	Hai Nhat Phan
Baptiste Jeudy	Simona E. Rombo
Julia Kiseleva	Bernard Zenko
Arto Klami	

Table of Contents

Invited Papers

Selected Contributions

Over-Fitting in Model Selection and Its Avoidance

Gavin C. Cawley

University of East Anglia, Norwich NR4 7TJ, UK
g.cawley@uea.ac.uk

Abstract. Over-fitting is a ubiquitous problem in machine learning, and a variety of techniques to avoid over-fitting the training sample have proven highly effective, including early stopping, regularization, and ensemble methods. However, while over-fitting in training is widely appreciated and its avoidance now a standard element of best practice, over-fitting can also occur in model selection. This form of over-fitting can significantly degrade generalization performance, but has thus far received little attention. For example the kernel and regularization parameters of a support vector machine are often tuned by optimizing a cross-validation based model selection criterion. However the cross-validation estimate of generalization performance will inevitably have a finite variance, such that its minimizer depends on the particular sample on which it is evaluated, and this will generally differ from the minimizer of the true generalization error. Therefore if the cross-validation error is aggressively minimized, generalization performance may be substantially degraded. In general, the smaller the amount of data available, the higher the variance of the model selection criterion, and hence the more likely over-fitting in model selection will be a significant problem. Similarly, the more hyper-parameters to be tuned in model selection, the more easily the variance of the model selection criterion can be exploited, which again increases the likelihood of over-fitting in model selection.

Over-fitting in model selection is empirically demonstrated to pose a substantial pitfall in the application of kernel learning methods and Gaussian process classifiers. Furthermore, evaluation of machine learning methods can easily be significantly biased unless the evaluation protocol properly accounts for this type of over-fitting. Fortunately the common solutions to avoiding over-fitting in training also appear to be effective in avoiding over-fitting in model selection. Three examples are presented based on regularization of the model selection criterion, early stopping in model selection and minimizing the number of hyper-parameters to be tuned during model selection.

Keywords: Model selection, over-fitting, early stopping, regularization.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, p. 1, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Intelligent Data Analysis of Human Genetic Data

Paola Sebastiani*

Department of Biostatistics, Boston University School of Public Health 801 Massachusetts Avenue, Boston 02118 MA, USA

Abstract. The last two decades have witnessed impressive developments in the technology of genotyping and sequencing. Thousands of human DNA samples have been genotyped at increasing densities or sequenced in full using next generation DNA sequencing technology. The challenge is now to equip computational scientists with the right tools to go beyond mining genetic data to discover small gold nuggets and build models that can decipher the mechanism linking genotypes to phenotypes and can be used to identify subjects at risk for disease. We will discuss different approaches to model genetic data, and emphasize the need of blending a deep understanding of study design, with statistical modeling techniques and intelligent data approaches to make analysis feasible and results interpretable and useful.

1 Introduction

Advances in the technology of parallel genotyping and sequencing have changed human genetics that in less than 10 years has moved from a hypothesis driven, candidate gene based approach to a hypothesis generation approach. Massive genotyping of well designed observational studies has generated large amount of data and many important discoveries. Although impressive, the yield of discoveries has been underwhelming, particularly considering the high expectation that motivated the rush to genome wide association studies (GWAS). The driving hypothesis of the GWAS era was the "common disease, common variant hypothesis" [1], which argued that the genetic profile of common diseases was determined by combination of genetic variants that are common in the population and have small effects. Based on this hypothesis, the genetic basis of common diseases could have been discovered by searching for common variants with different allele frequencies between subjects with disease (cases) and disease-free controls. These gold variants would have made it possible to measure individual risk to diseases such as diabetes, cardiovascular disease, Alzheimer's disease, but also estimate genetic predisposition to response to treatments and make personalized medicine a step closer.

Eight years of this "GWAS rush" have accumulated thousands of variants that have been associated with many diseases. For example, on August 10 2012, the

* Research supported by the National Heart Lung Blood Institute (R21HL114237) and National Institute on Aging (U19AG023122).

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 2–6, 2012.
© Springer-Verlag Berlin Heidelberg 2012

catalog of published genome-wide association studies maintained by the National Institute of Genome Research in the USA included 1338 publications and 6858 SNPs that have been significantly associated with many common diseases and replicated in more than one study (http://www.genome.gov/gwastudies/), [2]. While many of these variants are used by genetic testing companies in arguable genetic testing kits, the literature is rich of negative results that show how genetic data do not seem to improve our ability to differentiate susceptibility to disease compared to traditional, non-genetic risk factors [3].

What did go wrong? Was the "common disease, common variant hypothesis" wrong and rare rather than common variants are more likely to decipher the genetic basis of common disease? Where is the "missing heritability"? I believe it is too early to ask these questions because the majority of GWAS analyses have been very naive, very few analyses have attempted to tackle the complexity of genetic data, and much remains to be done with genetic data.

2 Alternative Approaches

The typical analysis of GWAS has been done one SNP at a time, using logistic regression for qualitative traits (presence/absence of disease), linear regression for quantitative traits (for example lipid levels), and survival analysis using Cox proportional hazards regression for time of events traits (age of onset of disease) [4]. Typically the genetic information is coded linearly, and when multiple genetic variants are associated with a trait, the multivariate set of variants is collapsed into a univariate score, called a genetic risk score, that is associated with the trait of interest using a simple regression model [5].

Models that are routinely used in data mining and knowledge discovery offer alternative and more flexible approaches to describe the genetic basis of complex traits. Examples are support vector machines [6], classification and regression trees, random forests [7], and Bayesian networks [8,9].

Networks have the advantage to represent the mutual associations between many variables, rather than the effect of input variables on a defined output variable. In Bayesian networks the associations between variables are represented by conditional probability distributions, and by using Bayes Theorem one can show how one or more events affect the outcome of one or more variables in the network. In this way, a Bayesian network model can be used for prognostic and diagnostic tasks.

Naive Bayes classifiers are simple Bayesian networks that have proved to be very effective in relating many genetic variants to binary traits [10]. Although Naive Bayes classifiers are not commonly used in statistical genetics, we showed that they are not so different from analysis based on a genetic risk score in [5], and in particular we showed that there is a mathematical relation that links a prediction rule based on a Naive Bayes classifier to a prediction rule based on logistic regression with genetic data summarized into a genetic risk score. The advantage of working with Bayesian networks modeling, even in the simple form of a Naive Bayes classifier, is their flexibility and the fact that this approach can

be easily generalized to model multiple traits simultaneously, to include multiple interacting genes and interaction between genetic and non genetic factors [9].

When the goal of the analysis is genetic risk prediction, an ensemble of Bayesian classifiers can improve prediction accuracy and appears to be robust to inclusion of false positive associations. However, inclusion of too many false positive can negatively impact the prediction accuracy and better methods are needed to select important genetic variants. In [11] we introduced a simple forward search to build nested Naive Bayes classifiers that can be used to also dissect a complex genetic traits. We introduced the idea of genetic risk profiles, that are determined by the vector of predictive probabilities of the set of nested classifiers. By cluster analysis, these genetic risk profiles can be used to identify subgroups of individuals who share similar risk for disease based not on the same number of genetic variants, but on patterns of genetic variants that determine similar risks.

3 Asking the Right Questions

Before investigating alternative methods of analysis of genetic data, it is important to ask some preliminary questions and set proper expectations. The starting point of older genetic studies was aggregation analysis to understand whether a disease clusters in families and to quantify the contribution of genetics to its susceptibility [12]. A related question is whether we should expect to predict every genetic trait with the same accuracy. The group of Peter Visscher has made some important contributions that relate the predictability of a trait to its prevalence in the population and its familial aggregation [13]. Their analysis showed that traits that are very common in the population and display low familial aggregation cannot be predicted with high accuracy even when all the associated genetic variants are know, while less common diseases with high degree of familial aggregation can be predicted with almost perfect accuracy when all the genetic variants are known. Being aware of the expected value of return of genetic data is important to decide the most appropriate method of analysis, and to assess the validity of results. For example, we should not expect to reach high accuracy in prediction of a trait that does not have a strong genetic basis. It is also important to understand the contribution of genetic and non genetic factors to susceptibility to disease, and whether the value of genetic data changes over time.

Another important feature of GWAS is the study design. The majority of GWAS are from observational studies that were designed to answer specific epidemiological questions. Study designs impose restriction on the randomness of the data that cannot be ignored: for example case control studies have to be analyzed using the retrospective likelihood rather than the prospective likelihood. In [9] we developed a Bayesian network to predict the risk of stroke in patients with sickle cell anemia. The study design was a nested case control study in which cases with disease and disease free controls were selected by design. The network modeled disease status as one of the root nodes so that the

conditional distributions of genetic variants were estimated conditionally on the disease status.

Another common study design in genetic epidemiology is the family based design, in which families are selected because of cluster of phenotype of interest and family members are then included in the sample. In family based designs, familial relation make observations dependent, and the strength of the correlation between subjects in the same family depends on the strength of the genetic basis of the trait. Ignoring familial relations can dramatically inflate the rate of false positive associations. Unfortunately, very few intelligent data analysis methods seem to be ready to accommodate this type of study design.

4 Conclusions

Intelligent data analysis of genetic data can help to compute susceptibility to complex genetic traits that are highly heritable. Many approaches are available, some are more promising than others. It is important to open good communication channels with geneticists, understand the issues of study design and the limitations that these studies impose on the type of analyses that are appropriate. Blending a deep understanding of study design with statistical modeling techniques and intelligent data analysis approaches will help to fully mine the wealth of genetic data.

References

1. Lander, E.: The new genomics: Global views of biology. Science 274, 536–539 (1996)
2. Manolio, T.A.: Cohort studies and the genetics of complex disease. Nat. Genet. 41(1), 5–6 (2009)
3. Manolio, T.A.: Genomewide association studies and assessment of the risk of disease. N. Engl. J. Med. 363(2), 166–176 (2010)
4. Sebastiani, P., Timofeev, N., Dworkis, D.A., Perls, T.T., Steinberg, M.H.: Genome-wide association studies and the genetic dissection of complex traits. Am J. Hematol. 84(8), 504–515 (2009)
5. Sebastiani, P., Solovieff, N., Sun, J.X.: Nave bayesian classifier and genetic risk score for genetic risk prediction of a categorical trait: Not so different after all! Front. Genet. 3, 26 (2012)
6. Wei, Z., Wang, K., Qu, H.Q., Zhang, H., Bradfield, J., Kim, C., Frackleton, E., Hou, C., Glessner, J.T., Chiavacci, R., Stanley, C., Monos, D., Grant, S.F.A., Polychronakos, C., Hakonarson, H.: From disease association to risk assessment: an optimistic view from genome-wide association studies on type 1 diabetes. PLoS Genet. 5(10), e1000678 (2009)
7. McKinney, B.A., Reif, D.M., Ritchie, M.D., Moore, J.H.: Machine learning for detecting gene-gene interactions: a review. Appl. Bioinformatics 5(2), 77–88 (2006)
8. Jiang, X., Barmada, M.M., Cooper, G.F., Becich, M.J.: A bayesian method for evaluating and discovering disease loci associations. PLoS One 6(8), e22075 (2011)
9. Sebastiani, P., Ramoni, M., Nolan, V., Baldwin, C., Steinberg, M.: Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia. Nature Genetics 37(4), 435–440 (2005)

10. Okser, S., Lehtimäki, T., Elo, L.L., Mononen, N., Peltonen, N., Kähönen, M., Juonala, M., Fan, Y.M., Hernesniemi, J.A., Laitinen, T., Lyytikäinen, L.P., Rontu, R., Eklund, C., Hutri-Khnen, N., Taittonen, L., Hurme, M., Viikari, J.S.A., Raitakari, O.T., Aittokallio, T.: Genetic variants and their interactions in the prediction of increased pre-clinical carotid atherosclerosis: the cardiovascular risk in young finns study. PLoS Genet. 6(9) (September 2010)
11. Sebastiani, P., Solovieff, N., Dewan, A.T., Walsh, K.M., Puca, A., Hartley, S.W., Melista, E., Andersen, S., Dworkis, D.A., Wilk, J.B., Myers, R.H., Steinberg, M.H., Montano, M., Baldwin, C.T., Hoh, J., Perls, T.T.: Genetic signatures of exceptional longevity in humans. PLoS One 7(1), e29848 (2012)
12. Ott, J.: Analysis of Human Genetic Linkage. Johns Hopkins University Press, Baltimore (1999)
13. Wray, N.R., Yang, J., Goddard, M.E., Visscher, P.M.: The genetic interpretation of area under the roc curve in genomic profiling. PLoS Genet 6(2), e1000864 (2010)

Queries for Data Analysis

Arno Siebes

Algorithmic Data Analysis Group
Universiteit Utrecht
arno@cs.uu.nl

Abstract. If we view data as a set of queries with an answer, what would a model be? In this paper we explore this question. The motivation is that there are more and more kinds of data that have to be analysed. Data of such a diverse nature that it is not easy to define precisely what data analysis actually is. Since all these different types of data share *one* characteristic – they can be queried – it seems natural to base a notion of data analysis on this characteristic.

The discussion in this paper is preliminary at best. There is no attempt made to connect the basic ideas to other – well known – foundations of data analysis. Rather, it just explores some simple consequences of its central tenet: data is a set of queries with their answer.

1 Prologue

Not too long ago, data to be analysed was a (rectangular) table filled with numbers; though perhaps not always to be interpreted as numbers. This is no longer true, one might, e.g., want to analyse a large graph or a social network or a set of molecules or a collection of (multi-media) documents or click-stream data. Obviously these are all data sets in one way or another. The ways in which they are analysed, however, is as varied as their type. That is, data analysis techniques are often targeted at very specific data types. Clearly, this is necessary. One cannot expect algorithms to produce meaningful results whatever data is thrown at them. Or, can one?

Whatever kind of data one has, one can be sure of thing: the data can be queried. Perhaps only in a very limited form – has object x property y? – but even that forms a basis for data analysis. If this is all their is, one can, e.g., mine for all *frequent* queries, properties that are satisfied by at least $\theta\%$ of the objects. While this may not seem much, this is sufficient to built, e.g., classifiers. If there is *structure* in the collection of queries one can ask, the possibilities for data analysis become even larger. For, one can search for *syntactic* structure in the collection of queries that is reflected in the data, i.e., in the *semantics* of the queries. Phrased differently, such structure may allow us to reduce the data. Reducing the data is one of the main goals of exploratory data analysis. For, the reduction provides insight in the data.

This is in a nutshell what we set out to do in this paper: data analysis based on queries. This paper is preliminary at best. We only sketch the outlines of the

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 7–22, 2012.
© Springer-Verlag Berlin Heidelberg 2012

beginnings of such a theory of data analysis. We present some – very naive – algorithms, but no experiments. Its goal is not to convince you that this is the way to go; in fact, I'm not – yet – convinced it is. Rather, its goal is to show that such an approach is possible. So, it is an unusual paper – which is a prerogative of invited papers – but within the main goals of the IDA conference: thinking about the nature of data analysis.

2 What Is Data?

2.1 Considering Some Well-Known Options

In data analysis we are dealing with finite collections of finite objects. We may want to form an opinion on a potentially infinite collection of objects, indeed, we may even expect a potentially infinite stream of data, but at any point in time we have only seen information on a finite set of objects. In turn, each object may be arbitrarily complex and may even be potentially infinite in size – e.g., the world wide web – but at any point in time we have only access to a finite description of that object. Given that we have a computational view on the world, such a finite description of an object can be effectively be given as a natural number. Hence, our data set can always be seen as a finite set of natural numbers. In turn, such a set itself can, again, be effectively represented by a natural number – or a string of natural numbers, if preferred. Indeed, viewing data as a, potentially infinite, string of bits has proven to be highly successful in Algorithmic Information Theory [7]. For example, by identifying such a bit string with the shortest program that generates it, one can distinguish random data from data with structure. The more the string is compressed, the more structure it contains.

Knowing whether a data set is random or contains structure is, of course, the first question any data analyst wants to have answered. But if it contains structure, intelligent data analysis requires that that structure is brought to the open. It is not given that that is best done by the program that yields the best compression of the data. If only because our encoding in bit strings may hide details that are interesting and the compression may not even respect the boundaries between the individual objects. So, while Algorithmic Information Theory yields highly interesting data analysis methods – e.g., the Google distance [2] – it is not sufficient.

The type of analysis one can meaningfully torture data with depends only on properties of that data. This is, of course, a well-known fact in Statistics where it is known as "theory of measurement" [5]. For example, to perform ordinal analysis on data, the perceived order in the measurements – the domain of an attribute – should be meaningful for the objects we are modelling. Clearly, one can take one's favourite theory of measurement – yes, there is more than one [5] – and check what type your new kind of data is and you know which type of analysis you can perform. This is fine, but often one wants to analyse the data in a non-classical way. I can't, e.g., think of a classical analogue of the page rank problem [12]. That doesn't mean, however, that page rank is only applicable to a

linked collection of websites, other kinds of graphs may yield equally interesting results.

To capture this applicability, a far more detailed theory of measurement is necessary. We need to know which operators are available on the data and which laws these operators obey; of course, both the operators and their laws have, again, to be meaningful for the objects – and their characteristics – we are modelling. This is something a computer scientist can understand. It is simply the requirement that we have a *type system* such as we have for many programming languages. Type theory allows for pretty complex type structures, using, e.g., dependent types or – somewhat simpler – polymorphic typing [14]. Then by type-checking the applicability of an analysis method to a set of data can be determined. This makes data simply a bag of values of some given type. One example of such an approach is given in [8], in which the author develops a theory for (part of) machine learning based on higher order logic.

Type systems allow reasoning about abstract, complex, types, such as, e.g., lists of elements of an arbitrary type. When we want to become more practical, however, such abstract types have to be grounded. For, type systems usually build types recursively from a set of basic types. So, if we want to employ type theory we have to decide what our basic types are. We have to be careful, though. For, standard basic types are sets such as Integers, Reals, and Booleans. That choice, would make us end up where we started: data are numbers. Somehow we need a more abstract starting point.

2.2 Queries as Data

As noted in the Prologue, the fundamental property of data is that it can be queried. That is true not only for standard databases, but also for, e.g., document collections or, indeed, the world wide web. We, collectively, maintain sets of data because we and others can later search that data. So, why not use the query as our basic data type?

The standard point of view is that queries on data result again in data; in fact, that is exactly the famous "closed" property of the relational algebra [3]. In fact, for the practical purposes of querying that is the only sensible point of view. But that doesn't make it the only point of view.

Pattern miners have such a different point of view. Patterns are defined by properties that the objects in the database may or may not satisfy [9]. Patterns are, e.g., deemed interesting if they occur often – frequent pattern mining – or not very often, yet – emergent pattern mining. The point is that patterns are simply binary queries.

It is well-know – c.f., the famous "twenty questions" game – that all queries can be emulated by a string of binary queries. Clearly, that is not what we intend here. For, that would lead us straight back to Algorithmic Information Theory. For the moment we want to stay clear from both Scylla and Charybdis – Algorithmic Information Theory and Type Theory; or better, data as numbers.

Pattern miners usually count the support of a pattern, that is, the number of objects that satisfy the query. Rather than using the absolute value for the

support we opt for the relative variant. This leads us to the following definition of data.

Definition 1. *Let Q be a set of queries, a data set D for Q is given by a function*

$$D : Q \to [0, 1].$$

The set of all data sets for Q is denoted by $\mathcal{D}_Q$.

Note that we could – and perhaps should – be more careful here. As it is defined, $\mathcal{D}_Q$ is an uncountable set. Clearly, the intention is that for each $q \in Q$, $D(q)$ is some computable number.

Furthermore, note that with this definition we, implicitly, assume that each object can answer each query. For example in a multi-relational case, this assumption doesn't hold. While it is easy to formulate a more general definition that doesn't rely on this assumption, this would complicate our discussion unduly. Hence we refrain from doing so.

Example 1. To specify a transaction database, we first need the set of items it is defined over. Denote this set by $\mathcal{I}$. A transaction database is then given by a function

$$D : \mathcal{P}(\mathcal{I}) \to [0, 1],$$

which assigns a support to each item set.

Note that such functions may be inconsistent as a database. For example, with $I \subseteq J$ and $D(I) < D(J)$. Again, this can easily be repaired, but to keep this paper simple, we refrain from doing so. We simply assume that our data sets are consistent.

This example may seem the perfect illustration of the naivety of our definition of data. For rather than simply list all transactions, we opt for the far larger set of all item sets with their support! This is, however, a feature, not a bug.

2.3 Data in a Metric Space

Given that $\mathcal{D}_Q$ consists of functions into $[0, 1]$, it is obvious that it is a metric space. Metrics are simply inherited from more general function spaces. For the purposes of this paper, we use the well-known ℓ_2 metric.

Definition 2. *Let Q be a set of queries and let $\mathcal{D}_Q$ be the set of all data sets over Q. The distance d on $\mathcal{D}_Q$ is defined by:*

$$d(D_1, D_2) = \sum_{q \in Q} (D_1(q) - D_2(q))^2$$

That is, the more similar the answer to all queries for two data sets is, the more similar the two data sets are. This highlights that Q determines what is interesting in the data. If Q cannot distinguish between two data sets, they are for all practical purposes the same.

It is, of course, possible that not all queries in Q are equally important. For example, that different answers to q_1 are far worse for the user than different answers to q_2. In such cases, a weighted variant of the ℓ_2 metric is preferable. To keep our discussion as simple as possible, we do not use a weighted version.

3 What Is Data Analysis?

3.1 Structure in Query Space

Structure in the collection of queries means that if we know the answer to some of the queries, we can compute the answer to other queries. To do this, we need to know which queries, if any, can be used to compute the answer to a given query. For this we take our cue from type theory as briefly discussed in section 2.1. That is we assume that there is structure in the set of queries Q.

More precisely, we assume that we can *compute* with queries. Which is, of course, a usual property of query languages. Examples are the algebra from relational databases [3] or – simpler – the union operator for item sets. Since we want to keep our discussion as simple as possible, we model ourselves on this latter example. That is, we assume only one operator, denoted by $\otimes$. This operator works purely on a syntactical level, it manipulates the query expressions – whatever they are – it does not take their result on a data set into account.

Example 2. In our running example, $Q = \mathcal{P}(\mathcal{I})$ and, as already suggested above, $\otimes = \cup$.

Clearly, the syntactical fact that $I = I_1 \cup I_2$ does not mean that the support of I can be easily determined from the supports of I_1 and I_2. In fact, as we will see later, cases where one can determine the support is the kind of structure we are after; but for the moment we stay on the syntactical level.

By assuming an operator $\otimes$ on Q, we implicitly assume a – probably large – set of equations that hold on Q. Syntactical equations like:

$$q = q_1 \otimes q_2$$

These equations are immaterial to us, but we do assume that there is some algebraic structure on Q, associated with $\otimes$. In line with our running example we assume a monoid structure which is commutative and idempotent. For the sake of brevity, we will call this a query monoid.

Definition 3. *Q is a* query monoid *with respect to $\otimes$ iff it is a commutative monoid with respect to $\otimes$ and $\otimes$ is an idempotent operator on Q. That is,*

- $\forall q_1, q_2 \in Q : q_1 \otimes q_2 \in Q$;
- $\forall q_1, q_2, q_3 \in Q : q_1 \otimes (q_2 \otimes q_3) = (q_1 \otimes q_2) \otimes q_3$;
- $\exists e \in Q \; \forall q \in Q : e \otimes q = q \otimes e = q$;
- $\forall q_1, q_2 \in Q : q_1 \otimes q_2 = q_2 \otimes q_1$;
- $\forall q \in Q : q \otimes q = q$.

The set of expressions in the monoid - i.e., the monoid itself - is denoted by E_Q.

Example 3. Our running example is obviously a query monoid, with $\emptyset$ as its unit element. Having $\emptyset$ as an item set is slightly unorthodox, for other examples, a unit element may make more sense.

The monoid structure on Q gives us two things. Firstly, if we assume an order $\prec$ on the elements of Q – a lexicographic order – the expressions in E_Q have a normal form; simply by having the factors listed in this order.

Definition 4. *An expression $q_{i_1} \otimes \cdots \otimes q_{i_n} \in E_Q$ is in normal form iff e does not occur in the expression and for any pair q_{i_j} and q_{i_k} we have*

$$q_{i_j} \prec_l q_{i_k} \leftrightarrow j < k$$

Secondly – and more importantly – the monoid structure gives us the concept of a *generator*. A generator of Q is a subset that can generate each $q \in Q$.

Definition 5. *A subset $Q' \subseteq Q$ is a* generator *of Q iff*

$$\forall q \in Q \; \exists q_1, \ldots q_n \in Q' : q = q_1 \otimes \cdots \otimes q_n$$

That is, $E_{Q'} \equiv Q$. A generator is called a base *if it contains no proper subset which is also a generator.*

Clearly, each query monoid has at least one generator – Q generates itself – and thus at least one base. There is no guarantee, however, that there is exactly one base. For our running example, however, there is a unique base.

Example 4. The set $\{\{I\} \mid I \in \mathcal{I}\} \cup \emptyset$ is the unique base of Q. Any superset of this set is a generator of Q and vice versa.

If Q has two bases, say $B = \{b_1, \ldots, b_n\}$ and $T = \{t_1, \ldots, t_m\}$, B and T differ in at least one element. Without loss of generality, assume that b_1 is such an element, i.e., $b_1 \in B \backslash T$. Since T is a base, we know that there is a $J \subseteq \{1, \ldots, m\}$ such that

$$b_1 = \bigotimes_{j \in J} t_j$$

Expressing the t_i in the elements of B – which we can do since B is a base – we get that there is an $I \subseteq \{1, \ldots, n\}$ such that

$$b_1 = \bigotimes_{i \in I} b_i$$

In fact, since B is a base, we must have that:

$$b_1 = b_1 \bigotimes_{i \in I \backslash \{1\}} b_i$$

That is, for Q to have two bases, $b_1 = b_1 \bigotimes_{i \in I \backslash \{1\}} b_i$ must be possible in E_Q. This, e.g., the case when Q is not only a monoid, but also a group. For example, if Q has base $\{x, y, x^{-1}, y^{-1}\}$ then it also has base $\{xy^{-1}, y, x^{-1}\}$.

Whenever the equality cannot hold Q has necessarily a unique base. For example, whenever Q has a lattice structure which is compatible with $\otimes$ in the sense that the following laws hold in E_Q:

- $q_1 \preceq q_1 \otimes q_2$,
- $q_1 = q_1 \otimes q_2 \leftrightarrow q_2 \preceq q_1$,

Q has exactly one base.

One could argue that having multiple bases is actually a good thing. For, each base is a point of view on the data: different bases may discern different structure. However, such a discussion is beyond our scope. Here we assume that a fixed base $B \subseteq Q$ has been chosen. Thus, all generators we consider are supersets of B; we denote this set by $\mathcal{G}_{B,Q}$. The expressions we consider are always in normal form with regard to the generator G.

3.2 Evaluating Expressions

The monoid structure on Q is purely syntactical. Our goal is, of course, semantic structure, i.e., structure in D. To extend our syntactic structure to a semantic one, we first have to define how expressions over a generator G can be evaluated.

The basic idea is that given an evaluation - i.e., a value - for the queries in G, we want to evaluate - give a value to - expressions over G. That is, we are given data D_G for G, i.e.,

$$D_G : G \to [0, 1]$$

and we want to extend this to E_G using some evaluation function:

$$ev : E_G \times D_G \to [0, 1].$$

We have quite some freedom in defining ev. It should, of course, respect the monoid structure on Q, i.e., it should be some sort of homomorphism. That is, for $e_1, e_2 \in E_G$ we expect something like

$$ev(e_1 \otimes e_2, D_G) = g(f(ev(e_1, D_G)), f(ev(e_2, D_G)))$$

to hold for some (computable) functions f and g. As before, we keep it simple here. That is, no functions just multiplication.

Definition 6. *Let $e \in E_G$, the evaluation function ev is defined recursively by:*

- *If $e = g_i \in G$, then $ev(e, D_G) = D_G(g_i)$*
- *If $e = e_i \otimes e_j$, then $ev(e, D_G) = ev(e_i, D_G) \times ev(e_j, D_G)$*

Clearly, ev on its own doesn't give us an element of $\mathcal{D}_Q$, for the simple reason that for any $q \in Q$, there may be many expressions $e \in E_G$ with $e = q$ in the monoid. Moreover, two such expressions may evaluate differently.

Hence, to extend data for a generator G to data for Q we have to decide which expression is used for which query. This is done by a *cover function*:

Definition 7. *Let $G \in \mathcal{G}_{B,Q}$. A cover of Q by G is a function:*

$$cov : Q \to E_G$$

such that for all $g \in G$, $cov(g) = g$ and for all $q \in Q$, $cov(q) = q$ in E_Q, i.e., in the monoid on Q. The set of all covers of Q by G is denoted by Cov_G.

Slightly abusing notation, we also use ev as a function $\mathcal{D}_G \times Cov_G \to \mathcal{D}_Q$ by

$$ev(D_G, cov)(q) = ev(cov(q), D_G)$$

3.3 Models

The straight forward thing to do may now seem to define models of D_Q by a triple (G, D_G, cov) for which

$$ev(D_G, cov) = D_Q.$$

However, this would imply that the only worthwhile structure one can discern in data faithfully encodes all particularities and details including noise. Using the metric on $\mathcal{D}_Q$ we could relax the condition to

$$d(ev(D_G, cov), D_Q) = \epsilon$$

for some acceptable ϵ. The problem with such a definition is that the cover function may be badly chosen. That is, G could *model* D_Q using D_G much more faithfully than the chosen structure reveals, if only a different cover function would be used. It would be best if the cover function was optimal. More in particular, let $G \in \mathcal{G}_{B,Q}$ and $D_G \in \mathcal{D}_G$, the optimal cover function cov_{D_G} is defined by

$$cov_{D_G} = \underset{cov \in Cov_G}{\operatorname{argmin}} \{m(ev(D_G, cov), D_Q)\}$$

Note that it is by no means given that there is a unique cover function that achieves this minimum. If there are more, we pick the first in some (lexicographic) order. Models are then defined as follows.

Definition 8. *Let $G \in \mathcal{G}_{B,Q}$ and $D_G \in \mathcal{D}_G$. The pair (G, D_G) is an ϵ-model of $D_Q \in \mathcal{D}_Q$ iff*

$$d(ev(D_G, cov_{D,G}), D_Q) = \epsilon$$

The set of ϵ-models is denoted by $\mathcal{M}^{\epsilon}_{Q,D}$, the set of all models by $\mathcal{M}_{Q,D}$.

Clearly, our models are not necessarily very good. If a models ϵ is large, one may question its value. Choosing an appropriate cut-off for an acceptable tolerance is a task for the – intelligent – data analyst, it has no theoretical solution. However, it is, of course, possible to design algorithms that return acceptable models whatever the threshold is.

4 Structure Function

If an acceptable similarity - i.e., threshold for ϵ - has been determined the task seems simply to find a model that satisfies this threshold. But this begs the question: are all models with the same similarity equally good? Surely they are not. For the purposes of this paper our goal is simple models, where simple is defined by the size of the generator.

4.1 Structure in $\mathcal{M}_{Q,D}$

Since for each $G \in \mathcal{G}_{B,Q}$ we have that $B \subseteq G \subseteq Q$ and vice versa, $\mathcal{G}_{B,Q}$ is a sub-lattice of $\mathcal{P}(Q)$. This lattice structure is inherited by $\mathcal{M}_{Q,D}$ in the following sense.

Let $G_1 \subseteq G_2 \in \mathcal{G}_{B,Q}$. Clearly, any $D \in \mathcal{D}_{G_2}$ induces data in $\mathcal{D}_{G_1}$, by restricting it to the queries in G_2, abusing notation we denote both by D. Clearly, for any $D' \in \mathcal{D}_Q$ we have that

$$d(ev(D, cov_{D,G_2}), D') \leq d(ev(D, cov_{D,G_1}), D').$$

In fact, it is easy to see that

$$\exists q \in Q : cov_{D,G_2}(q) \neq cov_{D,G_1})(q) \Rightarrow \\ d(ev(D, cov_{D,G_2}), D') < d(ev(D, cov_{D,G_1}), D').$$

It is not surprising that by making the generator larger we can get more similar to D. This observation, however, points to structure on $\mathcal{M}_{Q,D}$ we can exploit in our search for good models.

Definition 9

$$\mathcal{M}^k_{Q,D} = \{(G, D_G) \in \mathcal{M}_{Q,D} \mid |G \setminus B| = k\}$$

That is, $\mathcal{M}^k_{Q,D}$ consists of those models whose generator has k elements next to the base B. The idea is now, of course, that we search level-wise through $\mathcal{M}_{Q,D}$. Just as in [15] we will do that using a structure function.

4.2 The Structure Function

For a given k, the question which models in $\mathcal{M}^k_{Q,D}$ are to be preferred has an obvious answer: those who are most similar to D, of course. As before, it is not given that there is just one model with this property. However, using the lexicographical order on the elements of E_Q, we can easily define an order on $\mathcal{G}_B$. Given this order, we simply pick the first model that is most similar to D.

Definition 10

$$M^k_{Q,D} = \underset{(G,D') \in \mathcal{M}^k_{Q,D}}{\operatorname{argmin}} \{d(ev(D', cov_{D',G}), D)\}$$

Based on these optimal models, we define the structure function as follows.

Definition 11. *The structure function $\mathcal{S}_{Q,D} : \mathbb{N} \to \mathcal{M}_{Q,D}$ is given by:*

$$\mathcal{S}_{Q,D}(k) = M^k_{Q,D}$$

As in [15] we call this a structure function because of its similarity to the celebrated Kolmogorov structure function [7].

4.3 What Is Data Analysis? Revisited

The structure function gives a method to analyse data. Given a data set D we:

- first determine which set of queries Q are important for us on D;
- next we choose a base $B \subseteq Q$ and so determine the set of generators $\mathcal{G}_{B,Q}$;
- then we compute the structure function $\mathcal{S}_{Q,D}$;
- and choose the smallest k for which $\mathcal{S}_{Q,D}(k)$ is similar enough – for us – to D.

Note that we do not imply that all models we have seen *before* $M^k_{Q,D}$ are not interesting. Far from it, we do believe that these models give important insight in the structure of D. It are the models *after* $M^k_{Q,D}$ that are less interesting, simply because the hide structure by having a too large generator.

If there is more than one base, it is probably good to compute the structure function for each base. For, each base may shed a different light on the structure in the data and understanding the data is all what intelligent data analysis aims to achieve.

How Small Should k Be? Whatever value of ϵ one chooses, there will always be a model that is at least ϵ-close to the data, for the simple reason that Q models itself. Hence, the data analyst has complete freedom in choosing ϵ. Freedom brings responsibility: what is a reasonable choice?

Using the structure function, we can turn this question upside down. rather than limiting ϵ we can also limit k. That is, the question becomes: what is a good value for k?

In the traditional view, databases have a domain, Dom. If we disregard multiples – and multiples can be addressed using weights – a data set is always smaller than its Dom. Again traditionally, queries return some sort of "subset" of the original database. That means that the number of queries is related to the number of "subsets" i.e.,

$$|Q| \sim 2^{|Dom|}.$$

So, if $k = O(\log(|Q|))$, our model is roughly equally sized with the data set in its traditional form. Not much of a reduction and not easily understandable at all. In other words, one would like $k \ll O(\log(|Q|))$ to call our model good.

Hence, while it is impossible to give a definitive answer to the question how small k should be, we can say that

$$k = O(\log\log(|Q|))$$

would be a good demonstration of structure in the data.

5 Algorithms

Computing the structure function means that we have to compute $M^k_{Q,D}$ which is defined as:

$$\underset{(G,D')\in\mathcal{M}^k_{Q,D}}{\operatorname{argmin}} \{d(ev(D', cov_{D',G}), D)\}$$

That is, we have to minimize not only over the generators, but also over the data on this generator. Fortunately, this latter part is easy, for our metric is a quadratic function. In fact, it is easy to see that the following theorem holds by simply computing the minimum of the given quadratic expression.

Theorem 1. *Let $G \in \mathcal{G}_{B,Q}$, $cov \in Cov_G$ and $D \in \mathcal{D}_Q$. Then there is a unique $D_{cov} \in \mathcal{D}_G$ that minimizes*

$$d(ev(D_{cov}, cov), D)$$

The consequence of this theorem is that a simple exhaustive search is able to compute $M^k_{Q,D}$:

- for each generator with k non-base elements;
- determine each cover of Q
- determine the data in $\mathcal{D}_G$ that minimizes $d(ev(D_G, cov), D)$;
- choose the generator, cover, and data that lead to the overall minimum.

Unfortunately, this algorithm is infeasible. The search space is far too big. Moreover, there is no structure in $\mathcal{M}_{Q,D}$ we can exploit. In fact, it is easy to see that even if $G_1 \subseteq G_2$, there is not necessarily a simple relation between the two optimal cover functions. Hence, we have to resort to heuristics.

5.1 Heuristic Algorithms

First note that computing D_{cov} requires us to compute the minimum of a quadratic function with $|Q|$ terms. The value of this function is completely determined by the values assigned to the $g \in G$, for given a fixed cover

$$d(ev(D', cov), D) : \mathcal{D}_G \to \mathbb{R}.$$

Hence, if we assume that D is noise free on the $g \in G$ and thus set

$$\forall g \in G : D_{cov(g)} = D(g)$$

we don't have to minimize $d(ev(D', cov), D)$, we only have to compute it. Moreover, this assumption allows us to compute the optimal cover for each $q \in Q$ independently. For the optimal covers for queries $q_1, q_2 \in Q$ are no longer coupled via the minimization of $m(ev(D', cov), D)$. As an aside, note that this heuristic allows us to re-use computations. For, if

$$G_1 = G_2 \cup \{g\}$$

and we know the optimal cover for each $q \in Q$ from G_2, we only have to compute the optimal cover for those $q \in Q$ for which g may be part of the cover, to get the optimal cover from G_1.

Still computing the optimal cover – given $D_{cov(g)}$ – for all $q \in Q \backslash G$ is still a tall order. The heuristic to sidestep this problem is to compute $d(ev(D', cov_{D',G}), D)$ not on Q, but only on a random subset $Q' \subseteq Q$; assuming that a model that is

good on Q' will also be good on Q. The "optimal" model computed using these two heuristics is denoted by M^*_G.

These heuristics alone do not make our algorithm feasible, however. The remaining problem is that if $G_1 = G_2 \cup \{g\}$, $M^*_{G_1}$ can be both more similar and less similar to D than $M^*_{G_2}$. Hence, we still have to search through all generators with k non-base elements to find the (heuristically) optimal model. The heuristic we employ to get around this problem is the beam search. Hence our heuristic algorithm is as follows.

- We first compute M^*_B and $m(M^*_B, D)$
- Then we compute M^*_G for all generators that have one non-base element, as well as $m(M^*_G, D)$. Our set of base models H_1 consists of the *width* models most similar to D.
- Given H_k, H_{k+1} is computed as follows:
  - For each $M^*_G \in H_k$
  - Extend G with one more query in all possible ways
  - For each of these extended generators eG, compute M^*_{eG} and $m(M^*_{eG}, D)$ and add them to the set of candidates Can_{k+1}.
  - H_{k+1} is the set of *width* best candidates in Can_{k+1}.

To assess the quality of the chosen model(s) it is, of course, good to compute $d(ev(D', cov_{D',G}), D)$ with regard to Q rather than its random subset Q' used in the algorithm.

6 Related Work

The original motivation for this research lies in observations we made in our MDL based pattern set mining research. The goal of the Krimp algorithm is to find a small set of patterns that together describe the database well, i.e., compress the database well [16]. In follow-up research we noted that the resulting models – known as code-tables – have many more desirable properties. For example, the implicitly define a data distribution that characterises the database rather accurately. This distribution has been used for, e.g., classification [18] and imputation [19]. It is this same distribution that allows us to answer queries on the database from the code table only. Hence the question: what if we put this property central? This paper presents some first steps towards answering this question.

Another source of inspiration are the *non-derivable* item sets of Calders and Goethals [1]. An item set is derivable if its support can be computed from the support of others using the inclusion-exclusion principle. Given that patterns are queries, the relation with this paper is obvious. The difference of their approach with ours is twofold. Firstly, the derived support has to be exactly equal to the support in the database, we only require a more or less the same answer. The second difference is that we attempt to abstract from the specific type of data, we only require a monad structure. In fact, we only use monads as an example of the kind of structure one may want to impose and discover.

Our running example is very much related Webb's self-sufficient item sets [20]. One of its criteria is independence. Given our monad structure with multiplication as the sole operator, we also necessarily filter on independence. The difference is again that for us the monad structure is an example. That is also why we do not attempt to quantify our similarity using statistical tests – as is done for self-sufficient item sets – but only use the parameter ϵ.

Finally, since we only aim to approximate queries there is a clear relationship with so-called fault tolerant patterns [13]. A fault tolerant pattern is a normal pattern – i.e., query – but objects that almost satisfy it are also counted in the result. Both this and our approach aim to reduce the number of patterns. The difference lies in the way to achieve this reduction. Fault tolerant patterns do this by "assigning" objects to patterns, we do it by discovering structure in patterns.

7 Epilogue

Intelligent or exploratory data analysis is – for me – the search for structure in data. But, what is structure? In this paper we rely on two types of structure; a *mathematical* structure on the syntax of queries and a *computational* structure on the semantics of queries. Data analysis takes place in the interaction of these two types of structure.

7.1 Mathematical Structure

In this exploratory paper we assume a monoidal structure on the set of queries. This is not a random choice. Monoids are a well-known structure, both in Category Theory and in Type Theory.

If mathematics is the science of structure, category theory is its pinnacle [6]. It is the branch of mathematics were structure is studied in its most abstract form. In one direction, categorical theorems are theorems in many other branches of mathematics. In the other direction, category theory links branches of mathematics that are seemingly unrelated.

Since both Category Theory and Type Theory are concerned with structure, it is not surprising that the former plays a large role in the latter, e.g., to define semantics. As a further example, monoids are a well-known construction in functional languages, though often as *monad* rather than monoid [14].

Closer to our use, in a recent paper it is argued that relational – or SQL – databases are monads whereas nonSQL databases are co-monads [10]; the (categorical) dual of a monoids. In fact, there are more attempts to formalize databases through category, including query languages, schema transformations [17] etcetera.

The advantage for us of the categorical view – together with queries as functions into $[0, 1]$ – is that our (naive) algorithms are applicable to a wide variety of data types. Whether they are graphs, or item sets, or document collections, as long as they have a meaningful monoid structure the structure function is well-defined.

7.2 Computational Structure

While perhaps not immediately obvious, our structure function – indeed, our complete approach – fits nicely in the compression for knowledge approach of Algorithmic Information Theory. For, we encode Q by E_G using the cover function. Hence, by choosing suitable code words, we are compressing Q much as we did previously for the Krimp algorithm [16].

There is a major difference with AIT, though. We do not consider *all* programs. There are resource bounded variants of Kolmogorov complexity [7] in AIT, but we are even stricter than that. We only consider very specific programs. For, we rely on cover functions and expression evaluation. For this reason, our approach is actually close to the Minimum Description Length principle [4]. MDL is a principle to select a model from a given set of models. This is very much what we do.

The reason we are doing this is comprehensibility, as we already noted in Section 2.1. There is actually more to comprehensibility than our note there. Clearly, by allowing any program, we can discover very rich structure. However, rich structure is not necessarily easy to grasp. What if computing the answer to a query takes a very long time? Do we then really understand what is going on? See the notion of computational depth [7] in Algorithmic Information Theory for more on this.

One could say that are interest lies in relative complexity $C(x \mid y)$, in the sense that we would like to have the complexity of D given a model $M^k_{Q,D}$ – i.e., $C(D \mid M^k_{Q,D})$ – as low as possible – we want to get as much information out of the data as possible – while having comprehensible and easily computable models. Our research goal in this respect is to find "optimal" mathematical (syntactical) and computational (semantic) structure combinations.

As an aside, note that our use of queries is reminiscent of oracles in computability theory. One form such an oracle can take is that of an (infinite) bit string. By *querying* this bit string a Turing machine may compute things it cannot compute without. For example, if the bit string encodes whether or not Turing machines halt on inputs. Sequences that are incompressible even given another incompressible sequence as oracle are more random than "normal" random sequences [11].

Similarly, we give our programs data they can query: the data D_G on generator G. In AIT, all structure in D should be encoded by the pair (G, cov_G). However, as before, we sin against this ideal for simplicity and comprehensibility. For example, all structure in D that is not captured by Q will necessarily remain undetected by our models.

7.3 Data Analysis

In this paper we have introduced a form of data analysis where we search for syntactic structure that is reflected by the semantics. More in particular, a model allows one to compute the answer to any query on the data approximatively. The cover function is essential for this computation. Still, a model does not contain

the cover function. A model is given by a pair (G, D_G), i.e., by a generator an a data set over that generator.

The reason that we leave the generator out is that the pair (G, D_G) contains all the useful information that is present in (Q, D_Q). Everything else can be derived from it. That is, (G, D_G) is a reduced form of (Q, D_Q). Undoing reductions automatically – in this case, answering queries on D_Q using (G, D_G) only – is often impossible. For example, after a Latent Semantic Analysis part of the original data is lost.

Still, it would be interesting to find out whether some canonical order on the expressions in E_G could be used to recover the cover functions automatically. Very much like in the code tables used by Krimp.

The algorithms we introduced in this paper are rather naive. While the structure function gives you exactly what you want, it is computationally infeasible. The heuristics come with a cost: we have no guarantees how close our results will be to the optimal results. Hence, an important open problem is: device algorithms with guaranteed bounds.

Acknowledgements. I'm grateful to the Chairs of IDA 2012 for inviting me to talk – and write – about issues I've been thinking about for years already.

References

1. Calders, T., Goethals, B.: Mining All Non-derivable Frequent Itemsets. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 74–85. Springer, Heidelberg (2002)
2. Cilibrasi, R., Vitányi, P.: The google similarity distance. IEEE Transactions on Knowledge and Data Engineering 19(3) (2007)
3. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13(6), 377–387 (1970)
4. Grünwald, P.D.: Minimum description length tutorial. In: Grünwald, P.D., Myung, I.J. (eds.) Advances in Minimum Description Length. MIT Press (2005)
5. Hand, D.J.: Statistics and the theory of measurement. Journal of the Royal Statistical Society. Series A 159(3), 445–492 (1996)
6. Mac Lane, S.: Categories for the Working Mathematician. Springer (1971)
7. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and its Applications. Springer (1993)
8. Lloyd, J.W.: Logic for Learning. Springer (2003)
9. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. In: Data Mining and Knowledge Discovery, pp. 241–258 (1997)
10. Meijer, E., Bierman, G.M.: A co-relational model of data for large shared data banks. Commun. ACM 54(4), 49–58 (2011)
11. Nies, A.: Computability and Randomness. Oxford University Press (2009)
12. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1999)
13. Pei, J., Tung, A.K.H., Han, J.: Fault tolerant pattern mining: Problems and challenges. In: DMKD (2001)
14. Pierce, B.C.: Types and Programming Languages. MIT Press (2002)

15. Siebes, A., Kersten, R.: A structure function for transaction data. In: Proc. SIAM conf. on Data Mining (2011)
16. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: Proc. SIAM Conf. Data Mining, pp. 393–404 (2006)
17. Spivak, D.I.: Functorial data migration. Information and Computation 217, 31–51 (2012)
18. van Leeuwen, M., Vreeken, J., Siebes, A.: Compression Picks Item Sets That Matter. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 585–592. Springer, Heidelberg (2006)
19. Vreeken, J., Siebes, A.: Filling in the blanks - krimp minimization for missing data. In: Proceedings of the IEEE International Conference on Data Mining (2008)
20. Webb, G.I.: Self-sufficient itemsets: An approach to screening potentially interesting associations between items. ACM Transactions on Knowledge Discovery from Data 4(1) (2010)

Parallel Data Mining Revisited. Better, Not Faster

Zaenal Akbar, Violeta N. Ivanova, and Michael R. Berthold

Nycomed-Chair for Bioinformatics and Information Mining
Dept. of Computer and Information Science, University of Konstanz
Box 712, D-78457 Konstanz, Germany
firstname.lastname@uni-konstanz.de

Abstract. In this paper we argue that parallel and/or distributed compute resources can be used differently: instead of focusing on speeding up algorithms, we propose to focus on improving accuracy. In a nutshell, the goal is to tune data mining algorithms to produce better results in the same time rather than producing similar results a lot faster. We discuss a number of generic ways of tuning data mining algorithms and elaborate on two prominent examples in more detail. A series of exemplary experiments is used to illustrate the effect such use of parallel resources can have.

1 Introduction

Research in Parallel Data Mining traditionally is focused on accelerating the analysis process - understandable in times of limited compute power and increasingly complex analysis algorithms. More recently, however, data mining research has split into two main themes: "big data" type analyses, where the goal is still the efficient mining of insights from increasingly large datasets on the one hand and more elaborate mining algorithms in order to find better and more representative patterns in not necessarily such large data repositories, on the other.

With regard to the latter, usage of parallel compute engines has not gained much attention as the compute time tends to be less critical – especially in times when computational resources even on desktop computers are available in such abundance. Nevertheless many if not all relevant algorithms rely on heuristics or user supplied parameters to somewhat reduce the otherwise entirely infeasible hypothesis space.

In this paper, we argue that modern architectures, which provide access to numerous parallel computing resources, emphasized by the recent advance of multi-core architectures, can also be utilized to reduce the effect of these user parameters or other algorithmic heuristics. Instead of trying to provide the same results faster then conventional algorithms we claim that interest in this area of analysis methods should also consider using parallel resources to provide **better** results in **similar time**. In the following we refer to this use of parallel resources

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 23–34, 2012.
© Springer-Verlag Berlin Heidelberg 2012

as *tuning* models (in parallel) to distinguish it from the more common attempts to simply accelerate algorithms.

Obviously a number of algorithms can very easily be extended in such a manner. Most prominent are, of course, simple ensemble methods. Instead of sequentially training a bag of models it is embarassingly simple to train those in parallel. However, for many well known methods this type of straight forward parallelization is not applicable. Boosting already requires information about other models predictions and many other, presumably easy algorithms are not as easily parallelizable. Just try to parallelize a decision tree induction or a clustering algorithm. In addition, early experiments indicate that simply randomizing the collection of models is suboptimal – steering the parallel search by controlling diversity offers substantial promise here.

In this paper we propose two generic approaches for this type of parallel search algorithm. The resulting framework is demonstrated on two well-known algorithms that form the basis for many data mining methods.

Please note that this paper is meant as a demonstration of how parallel resources can be used for improving the quality of solutions of heuristic data mining algorithms in general. Therefore the parallel approaches discussed in this paper are simple and intuitive and do not aim to outperform optimized algorithms of the same type in practice. We strongly believe that the increasing amount of available parallel commodity hardware will lead to a rethinking of the design of heuristic data mining algorithms and the aim of this paper lies on this line of research.

2 Generic Approaches to Parallel Tuning

In order to better describe the generic approaches we are proposing in order to tune the accuracy of data mining algorithms let us first look at standard algorithms and how they search for a matching hypothesis in a given set of training data. Many of these algorithms (and these are the ones we are interested in here) follow some iterative scheme, where at each iteration a given model is refined. We can formalize this as follows:

$$m' = s\left(r(m)\right)$$

where m is the original model, for instance a neural network or a decision tree under construction, and m' is the next intermediate model, e.g. the neural network after one more gradient descent step or after another branch has been added to the decision tree. The two functions $s(\cdot)$ and $r(\cdot)$ describe a selection resp. refinement operation. In the case of the neural network, the refinement operator represents the gradient descent step and there is no real selection operator. For the decision tree, the refinement operator would actually return a set of expanded decision trees and the selection operator picks the one with the highest information gain.

As mentioned in the introduction, most existing algorithms employ some sort of heuristic optimization. Gradient descent performs a local optimization during

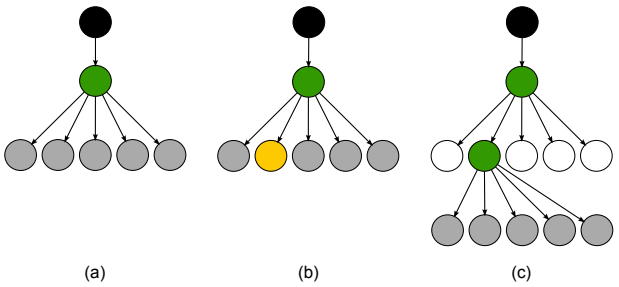


Fig. 1. The classic heuristic (often greedy) search algorithm. On the left (a), the current model m is depicted in green, the refinement options $r(m)$ are shown grey. The selection operator s picks the yellow refinement (b) and the next level then continues the search based on this choice.

the refinement operator. Greedy searches usually embed heuristics in both operators, they only generate a subset of all possible refinements and the selection operator has usually no way of estimating absolute quality but has to rely on a local heuristic, i.e. a greedy heuristic, as shown in Figure 1.

The formalization shown above now allows us to motivate the two general parallelization tuning methods - we can reduce (or – in extreme cases – even eliminate) the influence of the heuristics affecting either the selection function $s(\cdot)$ or the refinement operator $r(\cdot)$ or both.

2.1 Widening

This first approach, called *Widening* invests parallel resources into reducing the impact of the refinement heuristic by investigating more alternatives in parallel. In essence, this is similar to a beam search. Using the formalization from above, we can express widening as follows:

$$\{m'_1, \ldots, m'_{k'}\} = s\left(\{r(m_1), \ldots, r(m_k)\}\right).$$

The refinement operator $r(\cdot)$ does not necessarily change in this context and it can, as above, also return an entire set of refined models. The main point here is that we invest our available k parallel resources into running $r(\cdot)$ in parallel k times resulting in a much larger set of refined models. The selection function $s(\cdot)$ is now not forced to pick one locally optimal model but instead picks k' which are then refined again in parallel. In many cases $k = k'$, i.e. the number of explored models (or the width of the beam) will remain constant.

It is worth noting that this approach allows us not only to perform a beam search and explore the currently best k models but also to use a selection function which rewards diversity of the models and hence supports broader exploration of the hypothesis space. In recent work on the use of beam search in data mining algorithms evidence has arisen that such diverse space exploration is actually beneficial.

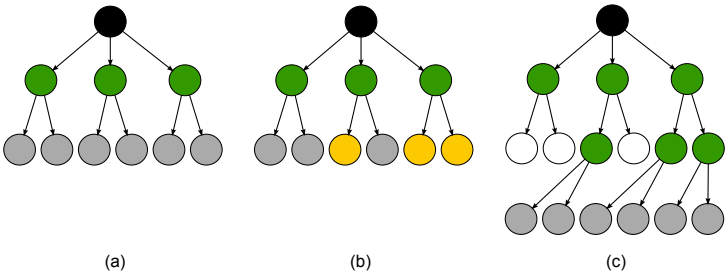


Fig. 2. Widening. From a set of models m (green circles), the refinement operator creates (in parallel) several sets of models (grey), shown on the left (a). The selection now picks a subset of the refined models (yellow circles in (b) and the search continues from those on the right (c).

Figure 2 shows how the widening approach works. At each step, the algorithm selects the k best models to be explored in the next step.

2.2 Deepening

The second approach, called *Deepening* focuses on reducing the impact of the selection heuristic:

$$m' = s_{\text{deep}}\left(r(m)\right).$$

In contrast to the widening approach, here only the selection function changes. For example, with regard to a decision tree: in order to determine the best split at each step only the immediate splits are considered. As we will describe in the next section, one could imagine looking further ahead to better estimate the quality of each split. In effect, the idea is to look forward and subsequently investigate how future steps of the refinement and selection process will be affected by the current choice. One could formalize this as follows:

$$s_{\text{deep}}\left(r(m)\right) = f\left(s\left(r(m)\right), s\left(r(s(r(m)))\right), \dots\right)$$

so the deeper selection operator is a function f of the normal, one-step-look-ahead selection criteria and the quality of further refinements of the original model. Figure 3 shows how the deepening approach works. At each step, the algorithm explores additional future models, selects the currently best model which, in turn, will lead to the best model in the future.

3 Example One: Tuned Set Covering

To illustrate the potential of the approaches discussed in the previous section, we chose the set cover problem, a problem that underlies quite a few data mining algorithms, for instance when trying to find the minimum number of rules or item sets.

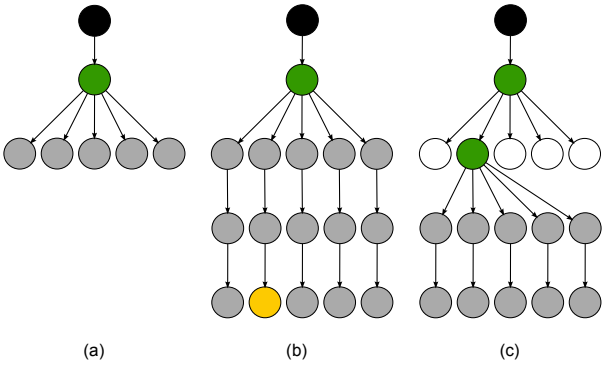


Fig. 3. Deepening. From one model m (green circles), the refinement operator creates (in parallel) several sets of models (grey), shown on the left (a). Note how, in order to apply the selection operator each of those choices is now expanded at a much greater depth (in parallel) using the classic heuristic search process. The selection then picks the refined models (yellow circles in (b)) that indicate the best performance at the deeper level, however the search continues from the models at the first level (c).

Although finding an optimal solution for the set cover problem is an NP-complete problem [1], a greedy algorithm, which at each step selects the subset with the largest number of uncovered elements, is widely used. Regardless of its simplicity, this classic greedy algorithm performs surprisingly well [2]. It can be shown that it achieves an approximation ratio of $H(n)$, where n is the size of the set to be covered.

However, please bear in mind that we are not interested in presenting a competitive new parallel set covering algorithm but that we are using this problem to illustrate how the tuning approaches presented above can be utilised. Hence we skip reviewing the state of the art in algorithmic improvements for the set covering problem here.

3.1 The Set Cover Problem

We consider the standard (unweighted) Set Cover problem. Given a universe X of n items and a collection $\mathcal{S}$ of m subsets of X : $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. We assume that the union of all of the sets in $\mathcal{S}$ is X, with $|X| = n$: $\bigcup_{S_i \in \mathcal{S}} S_i = X$. The aim is to find a sub-collection of sets in $\mathcal{S}$, of minimum size, that covers all elements of X.

3.2 Greedy Set Covering

The greedy algorithm [2] attempts to construct the minimal set cover in the following way. It starts with the empty set being the temporary cover and at each

step selects and adds a single subset to it. The subset selected is the one, which contains the most elements that are not yet covered by the temporary cover. To be consistent with the terminology previously defined: if C is the temporary cover, a refinement generated by $r(C)$ represents the addition of a single subset, not yet part of C, to C. From all the possible refinements, generated by $r(\cdot)$, the one with the largest number of elements is chosen as the new temporary cover.

3.3 Tuning of Set Covering

Two possible tuning possibilities of the Set Cover algorithm can be derived from the simple algorithm described above.

Widened Set Covering. In contrast to the greedy algorithm, the widening of the greedy algorithm builds k temporary covers in parallel. The focus in this algorithm is to use resources to explore (possibly very) large number of refinements in parallel, depending on the available resources. Here k is referred to as "widening" parameter.

A single iteration of the widened algorithm then operates as follows. Let $C_1, \cdots C_k$ represent the k temporary covers. A refinement of C_i is created by adding a new subset to C_i. For each C_i, the k refinements which contain the largest number of elements, are selected. This results in $k*k$ refinements in total. From those, the top k refinements are selected, resulting in k new temporary covers $C_1^{'}, \cdots, C_k^{'}$. The choice of parameter k depends on the available compute resources. If k parallel resources (cores, computers) are available the running time of the algorithm will remain the same as the one for the simple greedy algorithm. As we will see later, the quality of the solutions will increase with larger k, due to more options being explored.

Deepened Set Covering. Unlike the widening of the greedy algorithm, the focus of the deepening is to not explore several options in parallel but use the additional computing resources to evaluate the quality of the candidates in more depth. This results in a less greedy algorithm as the choice at each step is based on more knowledge about the "future" quality of that solution.

At a given step, the algorithm explores the k refinements, which cover the largest number of elements. From them the algorithm selects only one as a new temporary cover. To select from the k refinements in the selection stage, the algorithm builds "deeper covers" for each of them l steps ahead in parallel. The deeper cover for a refinement i is built by performing the simple greedy selection l times starting from refinement i. We will refer to this deeper cover as *l-deep cover*. The refinement with the largest l-deep cover is selected as a temporary cover for the next iteration of the algorithm. A breadth parameter k determines how many refinements will be explored at each stage, and a depth parameter l determines how many steps "ahead" will the algorithm do to build the future covers.

3.4 Experimental Evaluation

In order to compare the greedy heuristic and the two tuning approaches – widening and deepening, we used a number of publicly available benchmarks. We explicitly selected data sets, which pose a challenge for the simple greedy algorithm in order to demonstrate the merits of the tuned approaches. The greedy set covering algorithm is usually at a disadvantage when at many steps it has to select among many equally good subsets. So we picked data sets exhibiting this property.

The two data sets discussed here, $Rail - 507$ and $Rail - 582$, stem from the OR library [3] and arose from a real-life railway-crew scheduling problem. $Rail - 507$ is based on $|X_{507}| = 507$ elements and $|\mathcal{S}_{507}| = 63,009$ sets whereas $Rail - 582$ consists of $|X_{582}| = 582$ elements and $|\mathcal{S}_{582}| = 55,515$ sets. For both data benchmarks the sets themselves are fairly small none of them has a cardinality of more than 12. So the greedy algorithm quite naturally often encounters cases where different alternatives have exactly the same benefit. In addition, we ignored the cost information as we are dealing with the unicost version of the set cover problem here.

To assess a possible improvement achieved by parallelization, we compared the sizes of the obtained solutions, e.g. the number of sets in the final cover that each of the algorithm returns. The focus of our experiments lies on how the quality of the solutions changes, as we vary the number of used parallel resources. To evaluate the average performance of the algorithms with respect to the parallelization parameter, for each data set each algorithm was run randomized 50 times and the mean and the standard deviation of the size of the cover was reported. If our initial hypothesis is correct, the average performance should go up (here: the size of the cover should go down) when more parallel resources are used. Note that for the deepening the depth parameter is set to be maximal for all experiments. That is, during the selection the look ahead is conducted until the complete cover is reached.

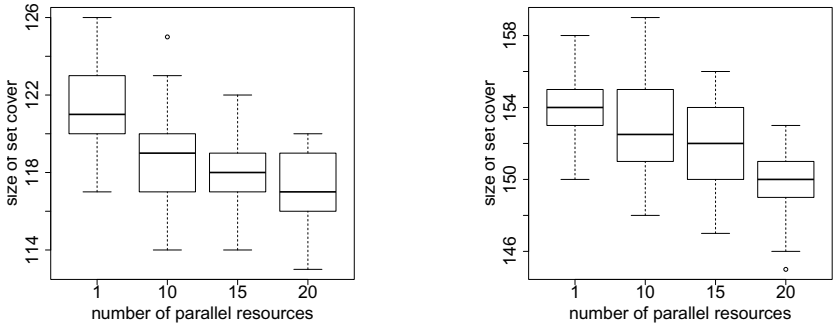


Fig. 4. Results for Widening on two data sets from the OR library (see text for details). As expected, with increasing widening of the search, the performance gets better and the standard deviation goes down.

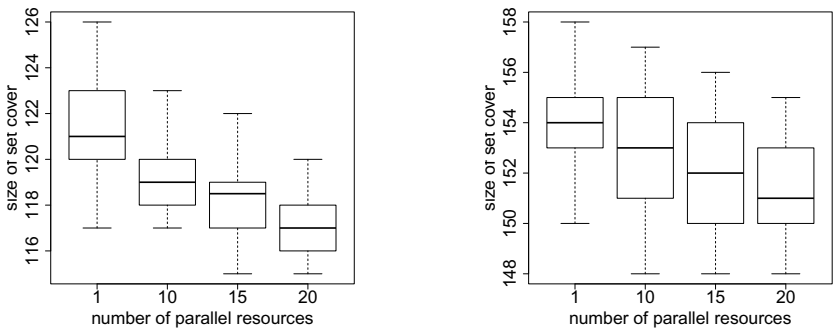


Fig. 5. Results for Deepening on two data sets from the OR library showing a similar trend to the widening approach

Figures 4 and 5 summarize the results. Note that number of parallel resource = 1 corresponds to the simple greedy heuristic. From the results one can conclude that when increasing the parallel resources, the mean of the obtained set cover decreases. A decrease in standard deviation also indicates that we are indeed converging towards a optimal solution.

4 Example Two: Tuned Decision Tree Induction

As a second example we investigated the well known decision tree models. The most prominent examples for decision tree learning are CART [4], ID3 [5], C4.5 [6] but many variations exist. Typically, however, they all start from a root node and grow the tree by splitting the data set recursively until a given stopping criterion is satisfied. The partitioning (or splitting) criterion is usually based on a greedy approach which picks the locally best attributes at each node. To determine the quality of a potential split, a measure for the expected information gain is derived from the available training data.

Mapping this to our representation, the refinement operator would create a series of possible splits at each node and the selection operator picks the split with the highest information gain (or another, similar measure of split quality).

4.1 Widening Decision Tree

From the discussion above it is quite obvious that random forests [7] are an incarnation of our widening approach by creating k trees on random data (and feature) subsets. However, in the end, the entire ensemble is used which is not the focus of our work – classic widening will pick the best model from the k models that were generated.

A straightforward extension would be to simply start with random starting points but poll the resulting trees (and the various possible refinements from each model in the current set) and use a selection criteria which rewards both tree size as well as accuracy estimates.

4.2 Deepening Decision Tree

Deepening decision tree induction requires more advance algorithmic modifications. The refinement operator expands candidate refinement solutions in parallel deeper until a given level k. A greedy selection operator is then employed to select the best solution to explored at the next iteration based on this deeper split quality estimate.

This seems an obvious extension of the standard decision tree learning methods which one would expect intuitively to perform well. However, the impact of the local, greedy choices on larger decision trees is not quite as dramatic as one expects. "Missed" opportunities at higher levels can easily be fixed by splitting (then just in neighboring branches) on that same attribute further down. In [8] such effects of "look ahead strategies" have been mentioned before and in [9] it was even reported that applying look-ahead in decision trees can produce trees that are both larger and less accurate than trees built by the normal algorithm.

In order to emphasize the potential benefit of tuning on this type of model learning we focused on decision stumps [10] which are essentially trees limited to a certain depth. Instead of constructing the tree until a certain criterion is met, these algorithms stop when a certain depth is reached. These types of models should be more sensitive to suboptimal local greedy choices when selecting splits and one should expect to see an effect of the deepening procedure described above.

4.3 Experimental Evaluation

Figure 6 shows results from widening (left) and deepening (right) on the Libras-Movement Dataset[1] with decision stumps limited to a depth of 6. We again ran 50 experiments on randomly chosen subsets of the training data (picking 90% of the data randomly each time). Also here we can see how the quality of the decision tree (here measured by the performance on the test data) improves when we conduct deeper refinements. Also the standard deviation of the accuracy goes down, indicating that we are converging towards an optimal solution.

5 Related Work

A wealth of literature exists relating to parallelized approaches for data mining and other algorithms. However to the best of our knowledge, no attempt has been made to employ parallel resources to improve accuracy in a similarly structured way. Nevertheless, many of the ideas presented elsewhere can be cast into the framework presented here one way or the other.

Some strategies, similar to the deepening strategy we discussed here, have already been proposed as a means to improve the accuracy of existing heuristics on sequential algorithms. For instance in [11] a "lookahead-search" approach to improve some greedy algorithms was discussed. With regard to decision trees in

[1] UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/index.html`

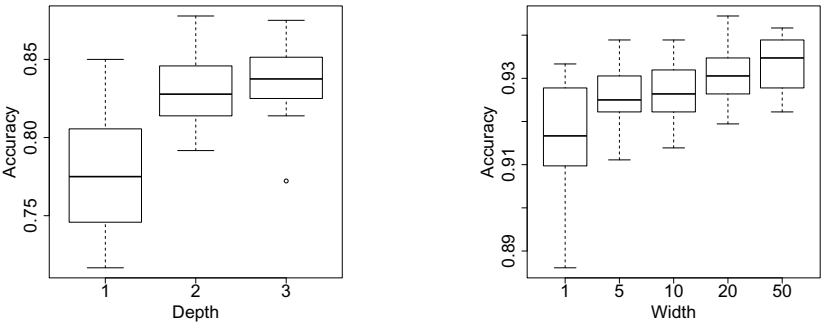


Fig. 6. Widening and Deepening decision tree induction (see text for details)

particular a number of sequential approaches have employed look-ahead strategies. In [9] the authors compare no lookahead and one-level lookahead and show the benefit of this type of lookahead were small or even none which confirms our observations with unstumped decision trees discussed earlier.

In [12] the lookahead approach is used differently. Instead of trying to find the split that produces the optimal tree, a greedy measure is used to find the locally optimal binary split. The subtree that is grown after this split is considered for the split decision. Their results show the algorithm to be useful in many cases. And finally in [13] two other lookahead-based algorithms for anytime induction of decision trees are presented. The depth-k look-ahead algorithm tests the effect of a split further down the tree until level k and in the second algorithm, each candidate split is evaluated by summing up the estimated size of each subtree, taking the smallest subtree as the optimal subtree at this stage.

Even more literature exists for the Set Cover problem as this is one of the most fundamental and best studied problems in optimization. A large amount of heuristics to solve it are also available, some of which focus on improving the accuracy of the result, others on the efficiency of obtaining a solution. Many of those ideas could also be resued for widening or deeping the search. The performance of the simple greedy algorithm comes very close to the best possible result for a polynomial-time algorithm, with optimality guarantees of $\ln(n)$. Various sequential algorithms have been developed to improve the bound and achieve $\log c$ factor from the optimal solution, such as [14], where the authors propose deterministic polynomial time method for finding a set cover in a set system (X, R) of VC-dimension d such that the size of the solution is at most $O(d \log(dc))$ of the optimal size, c.

Multiple parallel approaches exist for the set covering problem to improve the efficiency of the existing sequential algorithms. In [15] parallelization has been used to improve efficiency by "bucketing" utility values (the number of new elements covered) by factors of $(1 + \epsilon)$ and processing sets within a bucket in parallel. The bucketing approach ensures diversity of the explored hypothesis space, which could be particularly interesting for the purposes discussed in this

paper. In [16], a similar approach has been employed, again resulting in a linear-work RNC $(1+\epsilon)H_n$-approximation algorithm.

A veritable flood of publications is available on various aspects of other data mining algorithms for various types of distributed and parallel architectures. The focus of those papers is almost exclusively on reproducing the same results as the sequential algorithm but in a much shorter time. However the scope of this paper is not to present yet another parallel algorithm that outspeeds the sequential version. Instead, we present a general approach of how increasing computing power can be used to increase accuracy and reduce the impact of the underlying heuristics. Nevertheless in order to develop truly useful tuned algorithms it is worth investigating this literature and learning from the lessons presented there.

6 Conclusion

We have introduced a generic approach to make use of parallel resources to improve the accuracy and reduce the heuristic bias of common data mining algorithms. We demonstrated this concept on two greedy heuristics – the greedy algorithm underlying the standard solution for the Set Cover problem and the induction of decision tree stumps. We presented two approaches for this type of "algorithm tuning": *deepening* and *widening*, both based on relaxing the greedy criterion for hypothesis refinement and selection.

Again, we emphasize that this type of work is not all that novel – research in the parallel computing community has already focused quite extensively on optimizing all sorts of search strategies. However, in the context of data mining, such approaches have not yet been utilized. In contrast to ensemble or other set-of-model approaches, we believe it is often still desirable to find a single (interpretable) model. Utilizing parallel resources to find a better quality model in similar time to sequential approaches offers potential here, especially when the tuning approaches presented above are combined with diversity criteria to enforce a thorough exploration of the hypothesis space.

We believe that this type of approach can also be used to estimate the quality of a solution, i.e. how close to the optimum the current solution actually is. This may enable us to build systems that, after a certain time, can provide hints as to how much more time (and/or computing resources) would be needed to achieve a certain model quality.

References

1. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
2. Johnson, D.S.: Approximation algorithms for combinatorial problems. In: Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, STOC 1973, pp. 38–49. ACM, New York (1973)

3. Beasley, J.E.: Or-library: Distributing test problems by electronic mail. The Journal of the Operational Research Society 41(11), 1069–1072 (1990)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
5. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (1986)
6. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
7. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
8. Quinlan, J.R., Cameron-Jones, R.M.: Oversearching and layered search in empirical learning. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1019–1024 (1995)
9. Murthy, S., Salzberg, S.: Lookahead and pathology in decision tree induction. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1025–1031 (1995)
10. Iba, W., Langley, P.: Induction of one-level decision trees. In: Proceedings of the Ninth International Workshop on Machine Learning, pp. 233–240 (1992)
11. Sarkar, U., Chakrabarti, P., Ghose, S., Desarkar, S.: Improving Greedy Algorithms by Lookahead-Search. Journal of Algorithms 16(1), 1–23 (1994)
12. Elomaa, T., Malinen, T.: On Lookahead Heuristics in Decision Tree Learning. In: Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E. (eds.) ISMIS 2003. LNCS (LNAI), vol. 2871, pp. 445–453. Springer, Heidelberg (2003)
13. Esmeir, S., Markovitch, S.: Lookahead-based algorithms for anytime induction of decision trees. In: Twenty-First International Conference on Machine Learning, ICML 2004, pp. 33–40. ACM Press, New York (2004)
14. Brönnimann, H., Goodrich, M.T.: Almost optimal set covers in finite vc-dimension (preliminary version). In: Proceedings of the Tenth Annual Symposium on Computational Geometry, SCG 1994, pp. 293–302. ACM, New York (1994)
15. Berger, B., Rompel, J., Shor, P.W.: Efficient nc algorithms for set cover with applications to learning and geometry. Journal of Computer and System Sciences 49(3), 454–477 (1994)
16. Blelloch, G.E., Peng, R., Tangwongsan, K.: Linear-work greedy parallel approximate set cover and variants. In: Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2011, pp. 23–32. ACM, New York (2011)

Weighting Features for Partition around Medoids Using the Minkowski Metric

Renato Cordeiro de Amorim and Trevor Fenner

Department of Computer Science and Information Systems
Birkbeck University of London, Malet Street WC1E 7HX, UK
{renato,trevor}@dcs.bbk.ac.uk

Abstract. In this paper we introduce the Minkowski weighted partition around medoids algorithm (MW-PAM). This extends the popular partition around medoids algorithm (PAM) by automatically assigning K weights to each feature in a dataset, where K is the number of clusters. Our approach utilizes the within-cluster variance of features to calculate the weights and uses the Minkowski metric.

We show through many experiments that MW-PAM, particularly when initialized with the Build algorithm (also using the Minkowski metric), is superior to other medoid-based algorithms in terms of both accuracy and identification of irrelevant features.

Keywords: PAM, medoids, Minkowski metric, feature weighting, Build, L-p space.

1 Introduction

Clustering algorithms follow a data-driven approach to partition a dataset into K homogeneous groups $S = \{S_1, S_2, ..., S_k\}$. These algorithms can be very useful when creating taxonomies and have been used in various different scenarios [1–5].

The heavy research effort in this field has generated a number of clustering algorithms, K-Means [6, 7] arguably being the most popular of them. K-Means iteratively partitions a dataset I around K synthetic centroids $c_1, c_2, ..., c_k$, each being the centre of gravity of its corresponding cluster.

Although popular, there are scenarios in which K-Means is not the best option. For instance, the clustering of malware [8], countries or physical objects would require the use of realistic centroids representing each cluster, while K-Means centroids are synthetic. With this is mind, Kaufman and Rousseeuw [4] introduced the partition around medoids algorithm (PAM). PAM can provide realistic representations of clusters because it uses medoids rather than centroids. A medoid m_k is the entity with the lowest sum of dissimilarities from all other entities in the same cluster S_k; so it does represent an actual entity in the cluster.

Because of its widespread use, the weaknesses of PAM are also well-known, some shared with K-Means. For instance, PAM treats all features equally, irrespective of their actual relevance, while it is intuitive that different features may have different degrees of relevance in the clustering. Another weakness is

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 35–44, 2012.
© Springer-Verlag Berlin Heidelberg 2012

that the outcome of PAM depends heavily on the initial medoids selected. These weaknesses have been addressed in relation to K-Means by Huang et al. [9–11], who introduced the weighted K-Means algorithm (WK-Means), which automatically assigns lower weights to less relevant features; and Mirkin [12] with his intelligent K-Means (iK-Means), a heuristic algorithm used to find the number of clusters in a dataset, as well as the initial centroids. We have extended both of these by introducing intelligent Minkowski weighted K-Means (iMWK-Means), which we have shown to be superior to both WK-Means and iK-Means [13].

In this paper we propose a new algorithm, called Minkowski weighted partition around medoids (MW-PAM), which produces realistic cluster representations by using medoids. This also avoids possible difficulties in finding the centre of gravity of a cluster under the Minkowski metric, as finding this centroid is no longer required. We initialize MW-PAM with medoids obtained with Build [4], a popular algorithm used to initialize PAM, and present versions using the Minkowski and Euclidean metrics for comparison. We experimentally demonstrate that our proposed methods are able to discern between relevant and irrelevant features, as well as produce better accuracy than other medoid-based algorithms.

2 Related Work

For a dataset I to be partitioned into K clusters $S = \{S_1, S_2, ..., S_K\}$, the partition around medoids algorithm attempts to minimise the K-Means criterion:

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(y_i, c_k), \tag{1}$$

where $d(y_i, c_k)$ is a dissimilarity measure, normally the square of the Euclidean distance, between entity y_i and c_k, the centroid of cluster S_k. K-Means defines a centroid $c_k \in C$ as the centre of gravity of the cluster S_k. This is a synthetic value and not normally a member of S_k. The partition around medoids algorithm (PAM) takes a different approach by using medoids rather than centroids. The medoid is defined to be the entity $m_k \in S_k$ that has the lowest sum of dissimilarities from all other entities $y_i \in S_k$.

PAM minimises (1), with c_k replaced by m_k, partitioning the dataset I into K clusters, $S = \{S_1, S_2, ..., S_K\}$, as follows:

1. Select K entities at random as initial medoids $m_1, m_2, ..., m_K$; $S \leftarrow \emptyset$.
2. Assign each of the entities $y_i \in I$ to the cluster S_k of the closest medoid m_k, creating a new partition $S' = \{S'_1, S'_2, ..., S'_K\}$.
3. If $S' = S$, terminate with the clustering $S = \{S_1, S_2, ..., S_K\}$, and corresponding medoids $\{m_1, m_2, ..., m_K\}$. Otherwise, $S \leftarrow S'$.
4. For each of the clusters in $S = \{S_1, S_2, ..., S_K\}$, set its medoid m_k to be the entity $y'_j \in S_k$ with the lowest sum of dissimilarities from all other entities $y_i \in S_k$. Return to step 2.

The PAM algorithm inherits some of the weaknesses of K-means, including the necessity of knowing K beforehand, the high dependence on the initial medoids, and its uniform treatment of all features of the entities in the dataset I. In this paper we assume K to be known, but deal with the other weaknesses.

We intuitively believe that PAM should benefit from feature weighting, so that less relevant features have a lower impact on the final set of clusters. In order to apply feature weights in K-Means and generalize to the Minkowski metric, we introduced a weighted dissimilarity measure [13]. Equation (2) shows the weighted dissimilarity between the V-dimensional entities y_i and m_k.

$$d_p(y_i, m_k) = \sum_{v=1}^{V} w_{kv}^p |y_{iv} - m_{kv}|^p, \tag{2}$$

where p is the exponent of the Minkowski metric and w_{kv} is a cluster and feature specific weight, accommodating the fact that a feature v may have different degrees of relevance for different clusters S_k. Our dissimilarity measure in (2) allowed us to generalise the work of Huang et al. [9–11], which used the Euclidean metric (i.e. $p = 2$), and use instead the following Minkowski-metric weighted K-Means criterion:

$$W_p(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} w_{kv}^p |y_{iv} - c_{kv}|^p, \tag{3}$$

subject to $\sum_{v=1}^{V} w_{kv} = 1$ for each cluster. We use crisp clustering in which an entity belongs solely to one cluster.

We followed [9–11] and [13] in the calculation of the weights w_{kv}. This takes into account the within-cluster variance of features. Features with a smaller within-cluster variance are given a higher weight according to the following equation:

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv}/D_{ku}]^{1/(p-1)}}, \tag{4}$$

where $D_{kv} = \sum_{i \in S_k} |y_{iv} - c_{kv}|^p$. There are indeed other possibilities, even when using the Euclidean dissimilarity $d(y, m) = (y - m)^T (y - m) = \sum_{v=1}^{V} (y_v - m_v)^2$ (where T denotes the transpose vector). Alternatively one could use one of the following.

(i) Linear combinations of the original features, extending the inner product to include a weight matrix W, so that $d(y, m) = (y - m)^T W (y - m) = \sum_{ij} w_{ij} (y_i - m_i)(y_j - m_j)$. There have been different approaches to determining suitable weights in W, for instance: by making major changes in the K-Means criterion (1) dividing it by an index representing the 'between cluster dispersion' [14]; by adjusting the feature weights using a heuristic procedure [15]; or by using extra information regarding pairwise constraints [16, 17].

(ii) Non-linear weights, setting the dissimilarity as below.

$$d(y, m) = (y - m)^T W^\beta (y - m) = \sum_{v=1}^{V} w_v^\beta (y_v - m_v)^2, \tag{5}$$

where W is now a *diagonal* feature weight matrix. This approach seems to have been introduced by Makarenkov and Legendre [18], and later extended by Frigui and Nasraoui [19] to allow cluster-specific weights, both just for with $\beta = 2$ (we note that this is the same as (3) for $p = 2$). The possibility of β being a user-defined parameter was introduced by Huang et al. [9–11], still using the Euclidean metric. It is worth noting that only for $\beta = 2$ are the weights feature-rescaling factors, allowing their use in the data-preprocessing stage regardless of the clustering criterion in use.

For a more comprehensive review, see [13, 11].

3 Minkowski Weighted PAM

The MWK-Means algorithm partitions entities around synthetic centroids in L_p-space. This presents us with two problems. Firstly, MWK-Means requires the calculation of the centre of gravity of each cluster. This centre is easy to calculate only in the cases $p = 1$ and $p = 2$, when it is equivalent to the median or mean, respectively. For other values of p, one can use a steepest descent method [13], but this only gives an approximation to the real centre.

The second problem we deal with here is that synthetic centroids cannot always be used. A clustering problem may require the centroids to have realistic values or even represent real entities in the dataset.

In order to deal with these two problems, we have designed the Minkowski weighted PAM algorithm (MW-PAM). This partitions entities around medoids using the weighted dissimilarity defined in (2), adding the constraint of using medoids rather than centroids in the minimisation of the criterion shown in (3). The steps of MW-PAM in partitioning the data into K clusters are as follows:

1. Select K entities $\in I$ at random for the initial set of medoids $M = \{m_1, m_2, ..., m_K\}$; set a value for p; $S \leftarrow \emptyset$.
2. Assign each of the entities $y_i \in I$ to the cluster S_k of its closest medoid m_k, using the dissimilarity in (2), generating the partition $S' = \{S'_1, S'_2, ..., S'_k\}$.
3. If $S = S'$, terminate with the clustering $S = \{S_1, S_2, ..., S_K\}$ and medoids $\{m_1, m_2, ..., m_k\}$. Otherwise, $S \leftarrow S'$.
4. For each of the clusters in $S = \{S_1, S_2, ..., S_K\}$, set its medoid m_k to be entity $y'_i \in S_k$ with the smallest weighted sum of the Minkowski dissimilarities from all other entities in the same cluster S_k. Return to step 2.

Because of the non-deterministic nature of MW-PAM, we ran it 50 times in each of our experiments (discussed in Section 4). Of course, if we were to provide MW-PAM with good initial medoids rather than use random entities, we could make MW-PAM deterministic. In an attempt to solve the same problem for PAM, Kaufman and Rousseeuw [4] presented the Build algorithm:

1. Pick the entity $m_1 \in I$ with the lowest sum of dissimilarities from all other entities in I as the first medoid. Set $M = \{m_1\}$.

2. The next medoid is the farthest entity from M. Select an entity m for which E_y is the maximum over $y \in I - M$, where $E_y = \sum_{m \in M} d(y, m)$. Add m to M.
3. If the cardinality of M equals K stop, otherwise go back to Step 2.

It is intuitively consistent to use the Minkowski dissimilarities in Build with the same exponent p as we use in MW-PAM.

4 Setting of the Experiments and Results

Our experiments aim to investigate whether MW-PAM is able to discern between relevant and irrelevant features, as well as produce better accuracy than other popular algorithms.

It is very difficult to quantify the relevance of features in real-world datasets, so we have added extra features as uniformly distributed random noise to each dataset we use. These features will clearly be irrelevant for clustering purposes.

We have performed experiments on 6 real-world-based datasets and 25 Gaussian mixture models (GM). The 6 real-world-based datasets were derived from two popular real-world datasets downloaded from the UCI repository [20]. These datasets were:

(i) Iris dataset
This dataset contains 150 flower specimens described over four features, partitioned into three clusters. From this dataset, we generated two others by adding two and four extra noise features.
(ii) Hepatitis dataset
This dataset is comprised of 155 entities representing subjects with hepatitis over 19 features, partitioned into two clusters. From this standardized dataset, we generated two others by adding 10 and 20 noise features.

The GMs were generated with the help of Netlab [21]. Each mixture contains K hyperspherical clusters of variance 0.1. Their centres were independently generated from a Gaussian distribution with zero mean and variance unity.

We generated 20 GMs, five in each of the following four groups: (i) 500 entities over 6 features, partitioned into 5 clusters, to which we added two noise features, making a total of 8 features (500x6-5 +2); (ii) 500 entities over 15 features, partitioned into 5 clusters, to which we added 10 noise features, totalling 25 features (500x15-5 + 10); (iii) 1000 entities over 25 features partitioned into 12 clusters, to which we added 25 noise features, totalling 50 features (1000x25-12 +25); (iv) 1000 entities over 50 features, partitioned into 12 clusters, to which we added 25 noise features, totalling 75 features (1000x50-12 +25).

After generating all datasets, we standardized them. The hepatitis dataset, unlike all the others, contains features with categorical data. We transformed these features into numerical features by creating a binary dummy feature for each of the existing categories. A dummy feature would be set to one if the original categorical feature was in the category the dummy feature represents,

and zero otherwise. At the end, the original categorical features were removed and each dummy features had its values reduced by their average. All other features were standardized by Equation (6).

$$z_{iv} = \frac{y_{iv} - \bar{I}_v}{0.5 * range(I_v)}, \tag{6}$$

where $\bar{I}_v$ represents the average of feature v over the whole dataset I. This process is described in more details in [12]. We chose it because of our previous success in utilizing it for K-Means [8, 22, 13].

Regarding the accuracy, we acquired all labels for each dataset and computed a confusion matrix after each experiment. With this matrix we mapped the clustering generated by the algorithms to the given clustering and used the Rand index as the proportion of pairs of entities in I for which the two clusterings agree on whether or not the pairs are in the same cluster. Formally:

$$R = \frac{a + b}{a + b + c}, \tag{7}$$

where a is the number of pairs of entities belonging to the same cluster in both clusterings, b is the number of pairs belonging to different clusters in both clusterings, c is the number of remaining pairs of entities.

Here we compare a total of 8 algorithms: (i)*PAM:* partition around medoids; (ii)*WPAM:* a version of PAM using the feature weighting method of Huang et al. [9–11] in which, in (2), p is used as an exponent of the weights, but the distance is Euclidean; (iii) *MPAM:* PAM using the Minkowski metric with an arbitrary p; (iv) *MW-PAM:* our Minkowski weighted PAM; (v)*Build + PAM:* PAM initialized with the medoids generated by Build; (vi)*Build + WPAM:* WPAM also initialized with Build; (vii)*M Build + MPAM:* MPAM initialized using the Minkowski dissimilarity-based Build; (viii)*M Build + MW-PAM:* MW-PAM also initialized using the Minkowski dissimilarity-based Build.

We discarded results for any of the algorithms in two cases: (i) if the number of clusters generated by the algorithm was smaller than the specified number K; (ii) if the algorithm failed to converge within 1,000 iterations. We ran each of the non-deterministic algorithms, PAM, WPAM, MPAM and MW-PAM, 50 times.

4.1 Results Given a Good p

In this first set of experiments we would like to determine the best possible accuracies for each algorithm. With this in mind, we provide each algorithm that needs a value for p with the best we could find. We found this by testing all values of p between 1 and 6 in steps of 0.1, and chose the one with the highest accuracy, or highest average accuracy for the non-deterministic algorithms.

Table 1 show the results for each of the three iris-based and hepatitis-based datasets. Regarding the former dataset, these are the original, and the ones with two and four extra noise features. We observe that the highest accuracy of 97.3% was achieved by Build + WPAM and M Build + MW-PAM for the original

dataset and the version with four extra noise features. We are not aware of any unsupervised algorithm having attained such high accuracy for this dataset. The set of medoids obtained from the Build algorithm was not the same in the two cases.

Although the accuracy of both Build + WPAM and M Build + MW-PAM was the same for all three datasets, Table 2 shows that M Build + MW-PAM generated lower weights than Build + WPAM for the irrelevant features in the version of iris with four extra noise features.

Our experiments with the hepatitis dataset show that, this time, M Build + MW-PAM had higher accuracy than Build + WPAM for two out of the three datasets and the same accuracy for the third. This was again higher that of the other deterministic algorithms. Even though the same or slightly better maximum accuracy was obtained by the non-deterministic algorithms, the average accuracy was always less than that of M Build + MW-PAM.

The results of the experiments with the GMs in Table 3 were more variable. They show that PAM had the lowest average performance of 32.1% over the 20 GMs, while M Build + MW-PAM had the best with 65.4% accuracy, closely followed by Build + WPAM with 64.7%.

Our tables show that increasing the number of noise features has resulted in slightly better results for a few datasets. As the increase appeared in only some experiments, this may not have statistical significance. We intend to address this in future research.

The processing time presented in all tables relates to a single run. The non-deterministic algorithms tended to have lower processing times; however, these were run 50 times.

Table 1. Experiments with the iris and hepatitis datasets. The results are shown per row for the original iris dataset, with +2 and +4 noise features, and the original hepatitis dataset, with +10 and + 20 noise features. The algorithms marked with a * use the value of p solely as the exponent of the weights, the distance is Euclidean.

	Iris					Hepatitis				
	Accuracy					Accuracy				
	avg	sd	Max	p	Time (s)	avg	sd	Max	p	Time (s)
PAM	78.8	16.7	90.7	-	0.05±0.01	68.5	5.0	79.3	-	0.08±0.02
	70.0	10.3	90.0	-	0.04±0.01	66.1	8.0	74.8	-	0.06±0.01
	66.6	8.5	77.3	-	0.04±0.01	62.7	9.2	81.3	-	0.06±0.01
WPAM*	91.0	10.4	96.0	2.1	0.10±0.04	78.8	0.9	80.0	1.0	0.06±0.002
	81.8	13.8	96.0	3.1	0.10±0.04	78.7	0.7	80.0	1.0	0.07±0.002
	82.6	10.9	96.0	4.8	0.10±0.05	78.4	0.8	80.0	1.0	0.07±0.003
MPAM	85.4	12.2	92.0	1.2	0.09±0.03	70.2	4.9	81.3	1.1	0.14±0.05
	75.4	11.8	93.3	1.2	0.08±0.03	69.0	6.9	76.1	1.1	0.18±0.04
	73.0	10.6	88.0	1.3	0.08±0.02	67.6	9.5	81.3	1.4	0.24±0.04
MW-PAM	90.6	9.9	96.0	4.2	0.12±0.05	78.7	0.8	80.0	1.0	0.08±0.0043
	83.0	13.8	96.0	2.7	0.20±0.08	78.8	0.7	80.0	1.0	0.08±0.004
	81.8	15.3	**97.3**	3.0	0.25±0.08	78.8	0.9	80.0	1.0	0.08±0.005
Build + PAM	-	-	90.0	-	0.56	-	-	69.7	-	0.45
	-	-	78.0	-	0.51	-	-	72.9	-	0.37
	-	-	77.3	-	0.51	-	-	52.9	-	0.38
Build + WPAM*	-	-	**97.3**	1.1	0.56	-	-	78.7	1.0	0.41
	-	-	**96.7**	1.2	0.56	-	-	**80.0**	1.0	0.42
	-	-	**97.3**	1.7	0.56	-	-	76.8	1.0	0.42
M Build + MPAM	-	-	92.0	1.0	0.52	-	-	70.3	1.0	0.41
	-	-	90.7	1.1	0.54	-	-	76.8	1.0	0.39
	-	-	82.0	1.0	0.51	-	-	75.5	1.5	0.56
M Build + MW-PAM	-	-	**97.3**	1.1	0.60	-	-	**80.0**	1.0	0.42
	-	-	**96.7**	1.2	0.65	-	-	**80.0**	1.0	0.42
	-	-	**97.3**	1.1	0.64	-	-	**80.0**	1.0	0.46

Table 2. Final weights given by the experiments with the iris dataset. It shows one row per cluster for each dataset and algorithm. NF stands for noise feature.

	Features							
Original iris	Sepal length	Sepal width	Petal length	Petal width	NF 1	NF 2	NF 3	NF 4
Build+WPAM	0.000	0.000	0.544	0.456	-	-	-	-
	0.000	0.000	1.000	0.000	-	-	-	-
	0.000	0.001	0.995	0.004	-	-	-	-
M Build+MWPAM	0.000	0.001	0.525	0.474	-	-	-	-
	0.000	0.000	0.967	0.033	-	-	-	-
	0.000	0.045	0.922	0.032	-	-	-	-
+2 Noise features								
Build+WPAM	0.000	0.002	0.832	0.166	0.000	0.000	-	-
	0.000	0.000	0.990	0.010	0.000	0.000	-	-
	0.002	0.018	0.932	0.049	0.000	0.000	-	-
M Build+MWPAM	0.004	0.011	0.499	0.486	0.000	0.000	-	-
	0.001	0.000	0.884	0.116	0.000	0.000	-	-
	0.013	0.164	0.690	0.133	0.000	0.000	-	-
+4 Noise features								
Build+WPAM	0.044	0.071	0.435	0.425	0.008	0.009	0.007	0.001
	0.028	0.007	0.759	0.204	0.000	0.000	0.001	0.001
	0.099	0.093	0.468	0.250	0.025	0.027	0.022	0.016
M Build+MWPAM	0.000	0.001	0.525	0.474	0.000	0.000	0.000	0.000
	0.000	0.000	0.967	0.033	0.000	0.000	0.000	0.000
	0.000	0.045	0.922	0.032	0.000	0.000	0.000	0.000

4.2 Learning p

We have shown in the previous section, that, given a good p, the M Build + MW-PAM algorithm provides accuracy and detection of irrelevant features that is competitive or superior to that given by other algorithms. Of course this raises the question of how one can obtain a good value for p.

Table 3. Results of the experiments with the Gaussian mixtures dataset. The results are shown per row for the 500x6-5 (+2), 500x10-5 (+15), 1000x25-12 (+25) and 1000x50-12 (+25), respectively. The algorithms marked with a * use the value of p solely as the exponent of the weights, the distance is Euclidean.

Algorithms	Accuracy			p	Time (s)
	avg	sd	Max		
PAM	38.8	6.3	**62.6**	-	0.14±0.04
	35.7	5.5	51.4	-	0.15±0.02
	19.6	1.9	26.3	-	0.37±0.08
	34.5	5.0	47.5	-	0.46±0.09
WPAM*	**50.3**	5.0	58.1	4.96±0.79	12.05±0.98
	62.0	5.0	67.4	4.86±0.83	17.08±2.96
	57.3	3.1	61.4	3.58±0.13	122.70±7.71
	76.7	1.1	77.9	3.30±0.43	114.60±4.17
MPAM	43.3	3.8	50.8	1.10±0.15	11.37±1.06
	40.8	2.7	44.6	1.06±0.13	11.65±2.0822
	25.8	1.0	27.0	1.00±0.00	103.06±1.04
	45.0	1.9	47.1	1.08±0.08	113.06±10.7
MW-PAM	44.0	4.2	51.5	4.71±0.81	11.93±1.28
	52.6	4.8	58.3	3.86±0.25	14.43±1.96
	50.8	3.3	55.3	3.98±0.19	118.70±3.11
	74.1	1.7	76.0	3.14±0.30	110.92±3.84
Build + PAM	39.0	8.6	54.4	-	9.90±0.15
	37.5	3.2	42.4	-	9.86±0.02
	20.3	1.2	22.1	-	99.21±0.32
	47.1	3.8	51.9	-	99.61±0.47
Build + WPAM*	43.5	8.2	61.4	4.48±0.97	12.19±0.88
	65.8	5.3	71.2	4.46±1.09	15.16±1.40
	54.9	6.0	64.0	3.80±0.69	118.92±7.95
	94.6	3.4	97.5	2.60±0.43	112.42±6.41
M Build + MPAM	48.5	7.8	**62.6**	1.12±0.25	11.07±0.76
	48.5	7.7	62.0	1.00±0.0	10.86±0.39
	26.2	2.7	30.9	1.02±0.04	105.78±6.02
	68.4	6.5	75.9	1.08±0.13	114.80±14.36
M Build + MW-PAM	43.3	7.7	60.2	2.89±1.36	12.20±0.71
	67.0	8.5	**76.2**	4.06±1.07	15.42±2.18
	55.1	5.4	63.0	3.88±0.63	118.70±3.69
	96.4	1.8	**98.2**	1.84±0.48	121.07±15.46

From our experiments, it is clear that the best value depends on the dataset. We have solved the same issue for MWK-Means using a semi-supervised approach [13] and suggest here a similar solution for the medoid-based algorithms. We repeated the following procedure 50 times.

1. Select 20% of the entities in the dataset I, at random, keeping a record of their cluster indices.
2. Run M Build + MW-PAM on the whole dataset I.
3. Choose as the optimal p that with the highest accuracy among the selected entities.

Table 4 shows the results we obtained using the above algorithm, as well as the optimal results of Tables 1 and 3. These clearly show that it is possible to closely approximate the optimal accuracy of M Build + MW-PAM when p is unknown by using semi-supervised learning.

Table 4. Results of the experiments using the semi-supervised learning to find p for the M Build + MW-PAM algorithm

Datasets	Learnt			Optimal		
	p	avg	Max	p	avg	Max
Iris	1.16±0.34	96.67±0.40	97.33	1.1	-	97.33
	1.25±0.18	95.25±1.84	96.67	1.2	-	96.67
	1.05±0.05	96.33±1.01	97.33	1.1	-	97.33
Hepatitis	1.44±0.85	77.57±2.90	80.00	1.0	-	80.00
	1.09±0.46	79.08±2.89	80.00	1.0	-	80.00
	1.10±0.37	79.12±2.04	80.00	1.0	-	80.00
500x6-5 +2	2.81±1.22	41.19±2.81	60.20	2.89±1.36	43.3±7.7	60.20
500x15-5 + 10	3.78±0.81	65.26±8.37	76.20	4.06±1.07	67.0±8.5	76.20
1000x25-12 + 25	4.00±0.60	54.09±5.10	63.00	3.88±0.63	55.1±5.4	63.00
1000x50-12 + 25	1.84±0.45	96.18±1.77	98.20	1.84±0.48	96.4±1.8	98.20

5 Conclusion

In this paper we have presented a new medoid-based clustering algorithm that introduces the use of feature weighting and the Minkowski metric.

We have performed extensive experiments using both real-world and synthetic datasets, which showed that the Minkowski weighted partition around medoids algorithm (MW-PAM) was generally superior or competitive in terms of both accuracy and detection of irrelevant features to the 7 other PAM algorithms considered. In terms of future research, we intend to replace the semi-supervised algorithm by finding a method for generating a suitable value for p from the data, as well as perform experiments with datasets with many more noise features.

References

1. Brohee, S., Van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 7(1), 488–501 (2006)
2. Hartigan, J.A.: Clustering algorithms. John Willey & Sons (1975)
3. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters 31(8), 651–666 (2010)

4. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. Wiley Online Library (1990)
5. Mirkin, B.: Core concepts in data analysis: summarization, correlation and visualization. Springer, New York (2011)
6. Ball, G.H., Hall, D.J.: A clustering technique for summarizing multivariate data. Behavioral Science 12(2), 153–155 (1967)
7. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, California, USA, pp. 281–297 (1967)
8. de Amorim, R.C., Komisarczuk, P.: On partitional clustering of malware. In: CyberPatterns, pp. 47–51. Abingdon, Oxfordshire (2012)
9. Chan, E.Y., Ching, W.K., Ng, M.K., Huang, J.Z.: An optimization algorithm for clustering using weighted dissimilarity measures. Pattern Recognition 37(5), 943–952 (2004)
10. Huang, J.Z., Ng, M.K., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5), 657–668 (2005)
11. Huang, J.Z., Xu, J., Ng, M., Ye, Y.: Weighting Method for Feature Selection in K-Means. In: Computational Methods of Feature Selection, pp. 193–209. Chapman and Hall (2008)
12. Mirkin, B.G.: Clustering for data mining: a data recovery approach. CRC Press (2005)
13. de Amorim, R.C., Mirkin, B.: Minkowski Metric, Feature Weighting and Anomalous Cluster Initializing in K-Means Clustering. Pattern Recognition 45(3), 1061–1075 (2011)
14. Modha, D.S., Spangler, W.S.: Feature weighting in k-means clustering. Machine Learning 52(3), 217–237 (2003)
15. Tsai, C.Y., Chiu, C.C.: Developing a feature weight self-adjustment mechanism for a K-means clustering algorithm. Computational Statistics & Data Analysis 52(10), 4658–4672 (2008)
16. Bilenko, M., Basu, S., Mooney, R.J.: Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In: Proceedings of 21st International Conference on Machine Learning, Banff, Canada, pp. 81–88 (2004)
17. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems 16, pp. 521–528 (2002)
18. Makarenkov, V., Legendre, P.: Optimal variable weighting for ultrametric and additive trees and K-means partitioning: Methods and software. Journal of Classification 18(2), 245–271 (2001)
19. Frigui, H., Nasraoui, O.: Unsupervised learning of prototypes and attribute weights. Pattern Recognition 37(3), 567–581 (2004)
20. Irvine UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/`
21. Nabney, I., Bishop, C.: Netlab neural network software. Matlab Toolbox
22. de Amorim, R.C.: Constrained Intelligent K-Means: Improving Results with Limited Previous Knowledge. In: ADVCOMP, pp. 176–180 (2008)

On Initializations for the Minkowski Weighted K-Means

Renato Cordeiro de Amorim[1,2] and Peter Komisarczuk[2]

[1] Department of Computer Science and Information Systems
Birkbeck University of London, Malet Street WC1E 7HX, UK
[2] School of Computing and Technology
University of West London, St Mary's Road W5 5RF, UK
renato@dcs.bbk.ac.uk, peter.komisarczuk@uwl.ac.uk

Abstract. Minkowski Weighted K-Means is a variant of K-Means set in the Minkowski space, automatically computing weights for features at each cluster. As a variant of K-Means, its accuracy heavily depends on the initial centroids fed to it. In this paper we discuss our experiments comparing six initializations, random and five other initializations in the Minkowski space, in terms of their accuracy, processing time, and the recovery of the Minkowski exponent p.

We have found that the Ward method in the Minkowski space tends to outperform other initializations, with the exception of low-dimensional Gaussian Models with noise features. In these, a modified version of intelligent K-Means excels.

Keywords: Minkowski K-Means, K-Means Initializations, L_p Space, Minkowski Space, Feature Weighting, Noise Features, intelligent K-Means, Ward Method.

1 Introduction

The aim of any taxonomy is the difficult task of accurately grouping N entities into K homogeneous clusters. There are a number of algorithms seeking to cluster data; of these, perhaps the best-known is K-Means [1, 2]. K-Means partitions a dataset into K non-overlapping clusters, so that an entity $y_i \in Y = \{y_1, y_2, ..., y_N\}$ is assigned to a single cluster $S_k \in S = \{S_1, S_2, ..., S_K\}$ through the iterative minimisation of the criterion in Equation (1).

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(y_i, c_k) \tag{1}$$

where $d(y_i, c_k)$ is the dissimilarity between y_i and its respective centroid $c_k \in C = \{c_1, c_2, ..., c_K\}$, the centre of gravity of cluster S_k. The K-Means criterion allows the use of any distance function. In this paper, we focus on the Minkowski metric, which between the V-Dimensional entities y_i and c_k is defined by $d_p(y_i, c_k) = (\sum_{v=1}^{V} |y_{iv} - c_{kv}|^p)^{1/p}$. The Minkowski metric is at $p = 1$ and 2 equivalent to the Manhattan and Euclidean metrics, respectively.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 45–55, 2012.
© Springer-Verlag Berlin Heidelberg 2012

The K-Means algorithm treats all features in a dataset in equal terms, being particularly inaccurate when clustering datasets contaminated by noise features. Taking this into account, Huang et al. [3–5] have devised Weighted K-Means (WK-Means) with the aim of assigning lower weights to less relevant features. We further improved their algorithm creating the Minkowski Weighted K-Means method (MWK-Means) [6], which iteratively minimises the criterion shown in Equation (2).

$$W_p(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} w_{kv}^p |y_{iv} - c_{kv}|^p \tag{2}$$

where w_{kv} is the weight of feature v in cluster k, and p is a user-defined parameter that may be found through semi-supervised learning (see Section 6). MWK-Means inherites some of the weaknesses of K-Means, among them: (i) its accuracy depends heavily on the initial centroids it is fed; (ii) K, has to be known beforehand; (iii) there is no guarantee the criterion will reach a global minimum.

Here we focus solely on the weakness (i). Previously we presented a partial solution by initializing MWK-Means with a modified version of Intelligent K-Means (iK-Means) [7], also in the Minkowski space. We showed Intelligent Minkowski Weighted K-Means (iMWK-Means), together with MWK-Means, to be superior to the Euclidean-based WK-Means [6]. The decision to use iK-Means as the foundation rather than another initialization algorithm in the previous publication [6] was mainly based on research by Chiang and Mirkin [8] in the Euclidean space. IMWK-Means performed well in datasets with a small-to-modest number of features, but not as well in datasets with a large number of features.

In this paper, we compare six initializations, including random, for MWK-Means in terms of accuracy, processing time and recovery of the Minkowski exponent p. There has been a considerable amount of research comparing the K-Means algorithm under different initializations [9, 8, 10, 11], but to our knowledge this is the first study to compare five of these in Minkowski space.

2 The Minkowski Metric in WK-Means

MWK-Means [6] extends Huang et al.'s [3–5] work of applying feature weights to K-Means in two ways: (i) transforming these feature weights into feature-rescaling factors and (ii) introducing the Minkowski metric. In order to accomplish this, we adjusted the Minkowski distance metric to take into account feature weights. Equation (3) computes the distance between an entity y_i and a centroid c_k, both with V features v, where p is a given parameter.

$$d_p(y_i, c_k) = \sum_{v=1}^{V} w_{kv}^p |y_{iv} - c_{kv}|^p \tag{3}$$

From Equation (3) we derived the new criterion shown in Equation (2), subject to $\sum_{v=1}^{V} w_v = 1$ with no partial membership. The weight w_{kv} was set in terms of

features and clusters, allowing a feature v to have different weights at different clusters. The actual calculation of a weight followed Equation (4), where $D_{kv} = \sum_{i \in S_k} |y_{iv} - c_{kv}|^p$ was also cluster-specific.

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv}/D_{ku}]^{1/(p-1)}} \tag{4}$$

The algorithm used to iteratively minimise Equation (2) is very similar to the original K-Means algorithm and we formalised as follows:

1. Acquire an initial set of K centroids and V weights per centroid (or set $v_{ik} = 1/V$).
2. Assign all entities to the closest centroid using the minimum distance rule and Equation (3).
3. Update all centroids to the Minkowski centres of their respective clusters. Should the centroids not move, stop.
4. Update the feature weights using Equation (4); Go back to Step 2.

The Minkowski centre stated in the above algorithm can be found with a steepest descent algorithm, as described in our previous work [6]. In the next section, we describe seven popular algorithms normally used to find the initial centroids for K-Means, four expanded to Minkowski space.

3 Initializations

The output of the MWK-Means criterion shown in Equation (2) varies according to p. This restricted us in terms of the initializations on which we could experiment. Nevertheless, we performed a series of experiments with six initializations that did not need the criterion output to find the centroids.

We consider it intuitive that initializations performing their search for centroids in Minkowski space may outperform those that work only in Euclidean space, when these centroids are used in MWK-Means. Taking this into account, we have applied the Minkowski metric to all initialization but the random, since the later is not distance based. We present these six initializations in their original form below.

Random. In this experiment we ran MWK-Means 100 times with random initial centroids taken from the entities in the dataset. Cluster areas are likely to have a higher density; therefore, there is a higher likelihood that a random initial centroid will come from an actual cluster.

Hartigan [12] introduced an algorithm hopping it would avoid the formation of empty clusters after the initial assignment of entities to centroids:

1. Sort all N entities in relation to their centre of gravity.
2. For each cluster $k = \{1, 2, ..., K\}$ set its centroid as the $1 + (k-1) * [N/K]^{th}$ entity.

Ward. Hierarchical algorithms take a different approach than K-Means not generating a single set of labels, but a hierarchy of clusters also known as a dendogram. Ward [13] introduced the hierarchical agglomerative Ward algorithm that was later used by Milligan and Cooper [14] to find the initial centroids for K-Means:

1. Set each entity as a singleton.
2. Calculate the Ward distances between all clusters and merge clusters S_{w1} and S_{w2}, which are the closest.
3. Substitute the references of each of the merged clusters S_{w1} and S_{w2}, with that representing a newly created cluster $S_{w1 \cup w2}$.
4. Should $K > K^*$, K^* being the desired number of clusters, return to step 3.

The Build algorithm. Centroids are not entities belonging to the dataset, but synthetic elements. The Partition Around Medoids (PAM) algorithm takes a different approach; instead of using synthetic centroids, it uses medoids, which are the entities themselves. Of course, such an algorithm still needs an initialization, and the Build algorithm [15] is commonly used for this.

1. Pick the entity c_1, which should be closest to the centre of gravity of the dataset and put it in C
2. Set the following centroid as the farthest from c_1. An entity c, for which E_y is the maximum over $y \in Y - C, E_y = \sum_{y \in S_y} d(j, c_1) - d(y, j) > 0$ where j is an entity that has not been selected. Add c to C.
3. If the cardinality of $C = K$ terminate, otherwise go to Step 2.

Astrahan [16] picks centroids taking into account the density of the areas in the dataset:

1. Set $d_1 = \frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} ||y_i - y_j||$.
2. Pick the $y_i \in Y$ entity with the largest density within the d_1 radius; there should be at least a distance of d_1 between y_i and all the other centroids.
3. If the number of centroids is smaller than K, return to Step 2.

iMWK-Means. IK-Means [7] is a heuristic modification of K-Means that can be used to obtain the number of clusters in a dataset, as well as initial centroids. This is a successful algorithm [8, 17, 18] that we modified to work in the Minkowski space and perform feature weighting, introducing the iMWK-Means [6].

1. Sort all entities in relation to the Minkowski centre of the dataset, c_c, using Equation 3 and $1/V$ for each weight.
2. Select the farthest entity from the Minkowski centre, c_t, as a tentative centroid.
3. Cluster all entities to one of the centroids, c_c or c_t, using the minimum distance rule and Equation (3).
4. Update c_t to the Minkowski centre of its cluster.
5. Update the weights, using Equation 4. If c_t moved in step 4, return to step 3.
6. Remove all entities assigned to c_t from the dataset. If there are still entities to be clustered, return to step 2.
7. Output the K centroids with the largest cardinality.

Having performed a number of experiments [6], we have found that the iMWK-Means is considerably superior in terms of cluster recovery than the original WK-Means.

The Minkowski versions of the above algorithms are equivalent to their originals when $p = 2$. The iMWK-Means method already uses the Minkowski metric to find the initial centroids and does not require modification.

4 Setting Up the Experiments

The purpose of our experiments was to compare the behaviour of MWK-Means under six different initializations, five in Minkowski space and random. In order to accomplish this, we performed a number of experiments on synthetic and real-world datasets. The real-world datasets were originally obtained from the UCI machine learning repository[1]. The synthetic datasets are Gaussian Models generated with Netlab software[2].

We calculate accuracies by mapping the clusters generated by MWK-Means to the original labels of the datasets using a confusion matrix. We ran the algorithms on a 64 bits Intel dual core computer of 2.4GHz using Matlab R2010a and measured the processing time in seconds.

When using real-world datasets, it is difficult to be certain about the quantity of the features affected by noise and the degree of this noise. To facilitate our experiments, we opted to add a different amount of noise features in all datasets.

We chose to utilise the real-world datasets, described below, because of their wide use in research:

Iris contains 150 entities over four numerical features, partitioned into three clusters of equal cardinality. We derived two datasets from it by adding two and four noise features to each.

Wine contains 178 entities over 13 features, partitioned into three clusters of cardinalities 59, 71 and 48. We generated two datasets from it by adding seven and 13 noise features to each.

Hepatitis contains 155 entities over 19 features, partitioned into two clusters with cardinalities 32 and 123. We derived other two datasets, by adding 10 and 20 noise features to each.

Pima Indians diabetes contains 768 entities over eight features, partitioned into two clusters of cardinalities 500 and 268. We derived one dataset from it by adding four noise features.

Our synthetic datasets had spherical clusters of variance 0.1, with mixture coefficients equal to $1/K$. Their centres' components were generated independently from a Gaussian distribution $N(0, 1)$ of zero mean and unity variance. We have referenced these GMs as $NxV - K(+NF)$, hence 500x6-5 (+2) is a GM with 500 entities originally over six features partitioned into five clusters to which we added two noise features, making a total of eight features. We generated a

[1] `http://archive.ics.uci.edu/ml/`
[2] `http://www.ncrg.aston.ac.uk/netlab/`

total of 25 GMs, 10 500x6-5 (+2) GMs and five for each of the configurations: 500x15-5 (+10), 1000x25-12 (+25), 1000x50-12 (+25).

All datasets were standardised using $z_{iv} = \frac{y_{iv} - \bar{Y}_v}{0.5 * range(Y_v)}$, where $\bar{Y}_v$ is the average of a feature v over all entities, and y_{iv} the value of feature v in a given entity $y_i \in Y$. The standardisation process did not use the standard deviation as a scaling factor. Such a standardisation would have biased the data towards unimodal distributions: in clustering, bimodal distributions should make a higher contribution [7]. The use of the range rather than standard deviation in clustering has considerable empirical support [14, 19].

In our first set of experiments, we analysed how MWK-Means performed under the discussed initializations given a good p; as p was unknown, we ran experiments with ps from 1.0 to 5.0 in steps of 0.1. In our second set of experiments (Section 6), we analysed how the different initializations impacted MWK-Means when we tried to learn the best p under a semi-supervised approach using only 20% of the data.

5 Results for Accuracy and Processing Time

We performed experiments with 36 datasets, 11 real-world-based and 25 solely synthetic. The details of the results obtained using the real-world and synthetic datasets can be found in Sections 5.1 and 5.2 respectively.

All of the initialization algorithms, but the random initialization were set in the Minkowski space and provided centroids to MWK-Means. Under the random initialization we considered the optimal p to be the one with the highest average accuracy.

5.1 Experiments with Real-World Data

The results of our experiments with the iris, wine, hepatitis and Pima Indian Diabetes datasets are shown in Tables 1 and 2. These two tables show the results of our experiments for a total of 11 datasets.

Taking first the perspective of the accuracy, the Minkowski Ward method presented clearly the best results. Its accuracy was the highest for 8 of the 11 datasets and it was competitive to the best result in the other three cases.

The processing time highly depends on the value of p. Experiments in which the optimal found p was equal to one or two were much faster than those with other values, for the simple reason that the Minkowski centre in these cases were equivalent to the median and mean respectively. When an algorithm reached its maximum accuracy with more than one p, we used the lowest p to calculate its processing time. This dependence on p does not invalidate our experiments. We wanted to know whether an initialization could reach an optimal result at a fast p.

For instance, the random initialization is intuitively the fastest initialization to generate the initial K centroids for MWK-Means, but in our experiments it was the fastest in only three cases as we also take into account the time that MWK-Means takes to converge. The Minkowski Ward method achieved good results providing the lowest processing time in 5 of the 11 datasets. It was followed by

Table 1. Experiments with the iris dataset to which we added none, two and four extra noise features, and the wine dataset to which we added none, 10 and 20 extra noise features (top, middle and bottom, respectively)

	Iris					Wine				
	Accuracy					Accuracy				
	μ	σ	Max	p	Time (s)	μ	σ	Max	p	Time (s)
Random	93.3	8.3	96.7	1.2	0.40±0.2	92.6	1.3	96.1	2.3	2.40±1.0
	88.0	16.9	96.0	1.2	0.89±0.5	92.2	6.8	95.5	1.6	5.54±2.4
	90.0	12.8	96.0	1.2	1.32±0.7	88.3	10.6	94.4	1.4	6.79±3.3
Hartigan and Wong	-	-	96.7	1.1, 1.2	0.12	-	-	94.9	1.9, 2.2	4.74
	-	-	96.0	1.1, 1.2	0.72	-	-	94.9	1.6, 2.1	4.67
	-	-	94.7	1.3	2.32	-	-	94.4	1.2	6.34
Minkowski Ward	-	-	96.7	1.1	0.43	-	-	95.5	1.9, 2.0	1.19
	-	-	95.3	1.4	0.69	-	-	95.5	1.8, 2.1	2.58
	-	-	96.7	1.1	0.35	-	-	93.8	1.6, 1.8, 2.2	2.56
Build	-	-	96.7	1.1, 1.2	0.70	-	-	94.9	2.6	2.0
	-	-	96.0	1.1-1.3	1.05	-	-	95.5	2.3, 2.4	6.67
	-	-	96.0	1.1, 1.2	1.19	-	-	94.4	1.1	5.76
Astrahan*	-	-	96.7	1.2	0.99	-	-	95.5	2.5	3.29
	-	-	96.7	1.1	1.17	-	-	94.4	1.6	4.28
	-	-	96.0	1.1, 1.2	2.28	-	-	93.8	1.6, 1.7	5.73
iMWK-Means	-	-	96.7	1.2	0.51	-	-	94.9	1.2	2.16
	-	-	96.0	1.1-1.3	0.94	-	-	95.5	2.2	3.77
	-	-	96.0	1.1, 1.3	1.40	-	-	94.9	1.1	6.94

Table 2. Experiments with the hepatitis dataset to which we added none, 10 and 20 extra noise features (top, middle and bottom, respectively). And Pima Indian dataset to which we added none and four extra noise features (top and bottom, respectively).

	Hepatitis					Pima Indians				
	Accuracy					Accuracy				
	μ	σ	Max	p	Time (s)	μ	σ	Max	p	Time (s)
Random	79.0	0.8	80.0	1.0	0.02±0.001	68.2	2.8	71.3	3.9	11.05±4.7
	78.8	0.8	80.0	1.0	0.02±0.002	63.6	5.4	68.4	2.0	0.10±0.1
	80.2	3.6	85.2	1.9	4.2±1.9	-	-	-	-	-
Hartigan and Wong	-	-	82.6	2.0	0.13	-	-	71.2	3.5	10.31
	-	-	85.2	2.3, 2.8	4.17	-	-	68.5	1.8, 2.9	21.40
	-	-	85.8	2.8, 4.1	3.10	-	-	-	-	-
Minkowski Ward	-	-	85.2	1.6	1.65	-	-	70.3	4.2	13.51
	-	-	85.2	2.0	0.17	-	-	68.7	1.8	25.45
	-	-	86.6	3.7	3.24	-	-	-	-	-
Build	-	-	83.9	1.5-1.8	1.51	-	-	72.0	4.8	16.92
	-	-	85.2	1.4, 1.8, 1.9, 2.1-2.3	3.91	-	-	57.9	1.3	19.55
	-	-	84.5	1.6	6.57	-	-	-	-	-
Astrahan	-	-	78.7	1.0	0.34	-	-	71.0	3.7, 3.8	20.84
	-	-	84.5	1.9	7.86	-	-	66.8	1.6, 1.7	34.37
	-	-	85.2	2.4, 2.5	4.92	-	-	-	-	-
iMWK-Means	-	-	84.5	2.3	3.23	-	-	69.4	4.9	18.22
	-	-	83.2	4.5	5.24	-	-	67.2	4.1	38.77
	-	-	79.3	1.2, 4.1, 4.5	9.72	-	-	-	-	-

the random initialization with the best performance in three datasets and the Hartigan and Wong method also with the best performance in three datasets.

5.2 Experiments with Synthetic Data

We present the results for our experiments with 25 synthetic datasets in Table 3.

In our experiments, we observed that iMWK-Means had the highest accuracy in the experiments with the low dimensional GMs (total of 8 features) and it was competitive in the dataset with a total of 25 features. In the other hand, iMWK-Means had low accuracy in the high-dimensional datasets (50 and 75 features). This is aligned with our previous findings [6]. The Minkowski version of the Ward method performed very well in the high-dimensional GMs (with totals of 25, 50 and 75 features), being the sole achiever of 100% accuracy in the dataset with the largest number of features.

We find interesting that iMWK-Means and the Minkowski Ward seem to complement each other in terms of accuracy. It appears that while the former works better at a small number of features, the latter works better in GMs with a high number of features.

The Minkowski Ward was the fastest algorithm in the three GM groups with the highest number of features (totals of 25, 50 and 75). We acknowledge that hierarchical algorithms such as the Minkowski Ward are not known to scale well, and perhaps further experiments would be needed to set its performance in larger datasets.

We find our results to be very promising, particularly those obtained with the iMWK-Means and the Minkowski Ward algorithms. In order to quantify the algorithms' ability to recover a good p, we experimented with semi-supervised in the next section.

Table 3. Experiments with 10 GMs 500x6-5 (+2), five GMs 500x8-5 (+2), five GMs 1000x25-12 (+25) and five GMs 1000x50-12 (+25)

Algorithms	Accuracy			p		Processing Time		
	μ	σ	Max	μ	σ	μ	σ	Min/Max
Random	60.3	6.4	90.2	1.5	0.1	11.9	7.5	3.1/55.0
	78.8	6.6	97.8	1.4	0.1	21.3	8.2	8.9/50.0
	68.4	3.1	86.0	1.3	0.05	115.5	41.2	51.4/253.6
	85.2	8.2	100.0	1.2	0.05	102.1	43.2	43.0/275.3
Hartigan and Wong	69.5	8.5	78.4	1.9	0.3	10.5	5.4	0.2/17.6
	68.5	39.3	98.0	1.4	0.7	22.6	9.6	6.7/31.8
	68.9	6.7	76.2	1.4	0.1	114.6	25.2	83.3/147.3
	89.8	3.0	92.7	1.5	0.2	137.4	61.0	89.2/242.0
Mikowski Ward	58.3	7.1	67.6	1.6	0.3	10.6	7.6	2.4/27.7
	91.2	8.7	98.6	1.8	0.3	21.2	13.2	0.5/35.9
	89.2	2.7	90.9	1.3	0.04	78.1	11.5	68.6/97.9
	100.0	0	100.0	1.3	0.1	26.6	5.7	20.3/31.6
Build	58.3	8.2	70.8	1.4	0.2	22.3	4.0	17.0/30.3
	70.0	40.3	97.8	1.3	0.7	29.1	16.3	11.2/52.3
	83.1	11.6	97.2	1.3	0.1	215.7	13.9	201.0/238.1
	98.5	3.2	100.0	1.2	0.1	163.4	7.7	154.4/173.6
Astrahan	65.3	12.4	88.4	1.6	0.2	12.3	7.3	1.7/26.2
	67.0	38.5	97.8	1.2	0.6	28.9	14.9	11.0/45.6
	60.0	9.8	75.7	1.4	0.05	151.4	43.0	119.2/220.0
	82.4	10.6	93.2	1.4	0.2	124.1	36.5	89.9/182.0
iMWK-Means	70.3	11.9	87.6	1.8	0.5	10.9	4.7	0.3/17.8
	88.8	9.2	97.6	1.6	0.3	23.0	13.1	0.4/33.1
	54.9	3.3	59.5	1.5	1.8	203.9	54.7	143.0/278.3
	70.8	9.8	84.6	1.9	0.5	121.5	76.8	1.7/187.3

6 Learning p

In this section, we discuss our aim of verifying whether the initializations that performed well in our previous experiments would be capable of recovering a good value for p. To do so, we ran the algorithm below on all datasets 50 times with ps from 1.0 to 5.0, in steps of 0.1.

1. Randomly pick 20% of the data and respective labels.
2. Run experiments on the whole of the dataset.
3. Pick the p with the highest accuracy in the known labels.

The above algorithm closely approximates the optimal p and/or the maximum accuracy in all of the algorithms and datasets in the experiments. The results for Minkowski Ward and iMWK-Means are in Tables 4 and 5, respectively.

We did not apply this semi-supervised algorithm to the random initialization because its maximum accuracy, obtained in our experiments in section 5 showed

that even if we could have recovered a good p for it, its accuracy would have been less than what we could achieve with other initializations.

The pattern in the result is rather similar to our previous experiments. Minkowski Ward has performed quite well in the real-world-based datasets, with iMWK-Means being rather competitive. The experiments with the GMs again showed a much clearer picture, with iMWK-Means being the best performer in the GMs with a total of eight features, and the Minkowski Ward having better results in datasets with a higher number of features (25, 50 and 75 in total).

Table 4. Experiments using semi-supervised learning to find p using the agglomerative Ward algorithm and the Minkowski distance

Datasets		Accuracy		Optimals		
	p	μ	Max	p	μ	Max
Iris	1.2±0.4	94.53±3.2	96.7	1.1	-	96.7
Iris +2	1.3±0.1	94.87±0.3	95.3	1.4	-	95.3
Iris +4	1.1±0.05	95.33±1.7	96.7	1.1	-	96.7
Wine	1.6±0.5	94.07±1.7	95.5	1.9, 2.0	-	95.5
Wine +7	1.6±0.6	94.35±0.9	95.5	1.8,2.1	-	95.5
Wine +13	1.5±0.5	92.58±1.4	93.8	1.6,1.8,2.2	-	93.8
Hepatitis	1.4±0.4	82.52±2.5	85.2	1.6	-	85.2
Hepatitis +10	1.8±1.0	82.10±3.0	85.2	2.0	-	85.2
Hepatitis +20	2.5±1.2	82.64±3.2	86.4	3.7	-	86.6
Pima	3.8±0.8	69.13±1.7	70.3	4.2	-	70.3
Pima +4	1.8±0.2	66.91±2.2	68.7	1.8	-	68.7
500x6-5 +2	1.5±0.2	56.34±7.4	67.6	1.6±0.3	58.3±7.1	67.6
500x15-5 +10	1.7±0.3	90.80±7.8	98.6	1.8±0.3	91.2±8.7	98.6
1000x25-12 +25	1.3±0.05	89.18±2.4	90.9	1.3±0.04	89.2±2.7	90.9
1000x50-12 +25	1.2±0.03	99.99±0.03	100.00	1.3±0.1	100.0±0.0	100.0

Table 5. Experiments with semi-supervised learning to find p using the iMWK-Means algorithm

Datasets		Accuracy		Optimals		
	p	μ	Max	p	μ	Max
Iris	1.3±0.6	95.2±1.8	96.7	1.2	-	96.7
Iris +2	1.3±0.5	93.7±3.4	96.0	1.1-1.3	-	96.0
Iris +4	1.1±0.3	94.5±1.6	96.0	1.1, 1.3	-	96.0
Wine	2.2±1.2	93.6±0.9	94.9	1.2	-	94.9
Wine +7	1.5±0.6	93.0±1.6	95.5	2.2	-	95.5
Wine +13	1.7±1.4	93.6±2.0	94.9	1.1	-	94.9
Hepatitis	1.8±0.6	81.8±3.0	84.5	2.3	-	84.5
Hepatitis +10	3.1±1.6	80.2±4.1	83.2	4.5	-	83.2
Hepatitis +20	2.2±1.4	76.7±8.1	79.3	1.2, 4.1, 4.5	-	79.3
Pima	3.5±1.2	68.1±0.9	69.4	4.9	-	69.4
Pima +4	2.9±1.0	66.6±0.6	67.2	4.1	-	67.2
500x6-5 +2	1.7±0.5	69.3±12.1	87.6	1.8±0.5	70.3±11.9	87.6
500x15-5 +10	1.5±3.2	88.2±8.6	97.6	1.6±0.3	88.8±9.2	97.6
1000x25-12 +25	1.5±0.2	54.5±3.3	59.5	1.5±1.8	54.9±3.3	59.5
1000x50-12 +25	1.9±0.2	68.9±9.8	84.6	1.9±0.5	70.8±9.8	84.6

7 Conclusion and Future Work

We previously introduced MWK-Means [6] and initialized it with a version of iK-Means [7] in the Minkowski space, the iMWK-Means. IMWK-Means was in most cases superior to MWK-Means initialized with random centroids, but had a poor accuracy in high-dimensional GMs.

In this paper, we have discussed our extensive experiments comparing six initializations for MWK-Means, five in the Minkowski space plus a random initialization. We have used 36 datasets, real-world-based and synthetic to which we have added different amounts of noise features. We compared the initializations in terms of accuracy, processing time and the recovery of p.

Perhaps the most interesting outcome is the complementary accuracies of iMWK-Means with Minkowski Ward. While the former finds the optimal accuracy in GMs with eight features and is quite competitive in the group of datasets with 25 features, the latter tends to excel only in high-dimensional GMs, with 25, 50 and 75 features.

Our future research will address the development of a Minkowski version of a hierarchical algorithm as well as an application in malware clustering, following our previous publication [18].

References

1. Ball, G.H., Hall, D.J.: A clustering technique for summarizing multivariate data. Behavioral Science 12(2), 153–155 (1967)
2. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, California, USA, pp. 281–297 (1967)
3. Chan, E.Y., Ching, W.K., Ng, M.K., Huang, J.Z.: An optimization algorithm for clustering using weighted dissimilarity measures. Pattern Recognition 37(5), 943–952 (2004)
4. Huang, J.Z., Ng, M.K., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5), 657–668 (2005)
5. Huang, J.Z., Xu, J., Ng, M., Ye, Y.: Weighting Method for Feature Selection in K-Means. In: Computational Methods of feature selection, pp. 193–209. Chapman & Hall (2008)
6. de Amorim, R.C., Mirkin, B.: Minkowski Metric, Feature Weighting and Anomalous Cluster Initializing in K-Means Clustering. Pattern Recognition 45(3), 1061–1075 (2011)
7. Mirkin, B.G.: Clustering for data mining: a data recovery approach. CRC Press (2005)
8. Chiang, M.M.T., Mirkin, B.: Intelligent choice of the number of clusters in K-Means clustering: an experimental study with different cluster spreads. Journal of Classification 27(1), 3–40 (2010)
9. Pena, J.M., Lozano, J.A., Larranaga, P.: An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters 20(10), 1027–1040 (1999)
10. Steinley, D., Brusco, M.J.: Initializing K-Means batch clustering: A critical evaluation of several techniques. Journal of Classification 24(1), 99–121 (2007)
11. Maitra, R., Peterson, A.D., Ghosh, A.P.: A systematic evaluation of different methods for initializing the K-Means clustering algorithm. TKDE (2010)
12. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C. 28(1), 100–108 (1979)
13. Ward Jr., J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 236–244 (1963)
14. Milligan, G.W., Cooper, M.C.: A study of standardization of variables in cluster analysis. Journal of Classification 5(2), 181–204 (1988)
15. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. Wiley Online Library (1990)

16. Astrahan, M.M.: Speech analysis by clustering, or the hyperphoneme method. Issue 124 of Memo (Stanford Artificial Intelligence Project) (1970)
17. de Amorim, R.C.: Constrained Intelligent K-Means: Improving Results with Limited Previous Knowledge. In: ADVCOMP, pp. 176–180 (2008)
18. de Amorim, R.C., Komisarczuk, P.: On partitional clustering of malware. In: CyberPatterns, pp. 47–51. Abingdon, Oxfordshire (2012)
19. Steinley, D.: Standardizing variables in K-means clustering. Classification, Clustering, and Data Mining Applications, 53–60 (2004)

A Skew-t-Normal Multi-level Reduced-Rank Functional PCA Model for the Analysis of Replicated Genomics Time Course Data

Maurice Berk[1] and Giovanni Montana[2]

[1] Department of Medicine, Imperial College London, UK
maurice.berk01@imperial.ac.uk
[2] Department of Mathematics, Imperial College London, UK
g.montana@imperial.ac.uk

Abstract. Modelling replicated genomics time series data sets is challenging for two key reasons. Firstly, they exhibit two distinct levels of variation — the between-transcript and, nested within that, the between-replicate. Secondly, the typical assumption of normality rarely holds. Standard practice in light of these issues is to simply treat each transcript independently which greatly simplifies the modelling approach, reduces the computational burden and nevertheless appears to yield good results. We have set out to improve upon this, and in this article we present a multi-level reduced-rank functional PCA model that more accurately reflects the biological reality of these replicated genomics data sets, retains a degree of computational efficiency and enables us to carry out dimensionality reduction.

1 Introduction

The analysis of replicated genomics time series data sets is greatly complicated by the two nested levels of variation that they exhibit. On the one hand, for a given gene transcript, the replicates (e.g. human patients or laboratory mice, whatever the fundamental biological unit of the experiment is) display heterogeneous responses to the experimental conditions due to their unique genetic makeups. On the other hand, the mean expression levels across all replicates for a given gene transcript can evolve in markedly different ways depending upon the biological function that their corresponding gene fulfills. Furthermore, a great many of these transcripts will, in fact, be unaffected by the experimental conditions under study and therefore present with unchanging expression levels over the range of the time course, with observations varying due to measurement error alone. This characteristic leads to genomics time series data being highly non-normal which introduces significant additional complexity to the modelling process.

In this article we present a flexible multi-level reduced-rank functional PCA model that is capable of simultaneously modelling both levels of variation present in replicated genomics time series data sets while accounting for the departure

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 56–66, 2012.
© Springer-Verlag Berlin Heidelberg 2012

from normality. Functional models such as these have established themselves as the single most popular modelling approach for 'omics time series data analysis (Bar-Joseph et al., 2003; Luan and Li, 2003; Storey et al., 2005; Ma et al., 2006; Berk et al., 2011) as they are well suited to dealing with the few time points, few replicates, high degree of replicate heterogeneity, propensity for missing observations and high degree of noise that typify these experiments. Taking a functional PCA approach allows us to perform dimensionality reduction which is critical for visualising such high dimensional data sets and can aid with clustering. Furthermore, the reduced-rank approach to functional PCA, first introduced by James et al. (2000), greatly improves the computational efficiency of this process.

The model that we present here offers significant advantages over existing methods, as the 'state-of-the-art' within replicated genomics time series data analysis is to deal with the multi-level aspect by simply ignoring it and fitting each transcript independently with a single-level functional model (Storey et al., 2005; Berk et al., 2011). While these models have proven to be practically useful, this assumption is biologically implausible as it is well-known that genes interact in complex ways due to being coregulated. Furthermore, as these experiments generally involve very few replicates due to experimental costs, fitting each transcript in isolation greatly limits the available observations, allowing outlying measurements to have greater influence than if they were considered in the context of other transcripts. Taking a multi-level approach therefore enables better estimates of each transcript's temporal behaviour to be obtained, which should lead to increased power to detect those transcripts with significant changes in expression levels over time.

Multi-level functional PCA models already exist in other domains (Di et al., 2009; Zhou et al., 2010) yet they cannot be directly applied to replicated genomics time series data. In the case of Di et al. (2009), they use the method of moments to estimate the covariance surfaces at each level, which is unlikely to yield good results with the limited number of time points (≤ 10) and replicates (≤ 10) in a typical genomics experiment. In comparison, the data set on which Di et al. (2009) illustrate their method has $3,000$ replicates and 960 time points.

The model of Zhou et al. (2010) is more closely related to ours; however, it differs in the following key ways. Firstly, due to the specific characteristics of their motivating data set, their model accounts for spatial correlations between the variables (gene transcripts in our setting) which are typically not present in genomics experiments. Secondly, they assume that the second-level variance (the between-replicate in our case) is the same for all variables, which lacks biological plausibility in the genomics setting. Thirdly, their model fits a common error variance for all variables which, again, is known not to be valid for microarray technology (Tusher et al., 2001; Bar-Joseph et al., 2003) due to, for example, the way in which the slides are printed. Fourthly, they assume that the principal component loadings are normally distributed at both levels — an assumption which is invalid for genomics data, as we noted above and will demonstrate in due course.

We have structured this article in the following way. In Section 2 we introduce our reduced-rank multi-level functional PCA model. In Section 3 we describe our distributional assumptions and the process of parameter estimation using a Monte Carlo EM (MCEM) algorithm. We conclude with a discussion in Section 4.

2 The Skew-*t*-Normal Multi-level Functional PCA Model

A suitable model that accounts for the unique characteristics of replicated genomics time series data is to treat the expression level for replicate j for gene transcript i at time t as the sum of four components:

$$y_{ij}(t) = \mu(t) + f_i(t) + g_{ij}(t) + \epsilon_i(t) \tag{1}$$

where $\mu(t)$ is the 'grand mean' function representing the average gene expression levels over time across all transcripts in the data set, $f_i(t)$ is gene transcript i's deviation from that grand mean, such that $\mu(t) + f_i(t)$ gives the mean expression levels over time for transcript i across all replicates, $g_{ij}(t)$ is replicate j's deviation from the transcript mean function, and $\epsilon_i(t)$ is an error process.

In order to achieve computational efficiency, adhere to the principal of parsimony, and carry out dimensionality reduction, we take a reduced-rank approach (James et al., 2000) by replacing $f_i(t)$ and $g_{ij}(t)$ in (1) with their truncated Karhunen-Loève decompositons yielding

$$y_{ij}(t) = \mu(t) + \sum_{k=1}^{K} \zeta_k(t)\alpha_{ik} + \sum_{l=1}^{L_i} \eta_{il}(t)\beta_{ijl} + \epsilon_{ij}(t)$$

where $\zeta_k(t)$ is the k-th principal component function at the transcript level, α_{ik} is transcript i's loading on the k-th principal component function, $\eta_{il}(t)$ is the l-th principal component function at the replicate level, specific to transcript i and β_{ijl} is replicate j's loading on the l-th principal component function specific to transcript i. The number of principal components retained at each level are given by K and L_i. Representing the functions $\mu(t)$, $\zeta_k(t)$ and $\eta_{il}(t)$ using an appropriately orthogonalised p-dimensional B-spline basis (Zhou et al., 2008) and collecting all N_{ij} observations on replicate j for transcript i in the vector $\boldsymbol{y}_{ij}$ yields

$$\boldsymbol{y}_{ij} = \boldsymbol{B}_{ij}\boldsymbol{\theta}_\mu + \sum_{k=1}^{K} \boldsymbol{B}_{ij}\boldsymbol{\theta}_{\alpha_k}\alpha_{ik} + \sum_{l=1}^{L_i} \boldsymbol{B}_{ij}\boldsymbol{\theta}_{\beta_{il}}\beta_{ijl} + \boldsymbol{\epsilon}_{ij}$$

where $\boldsymbol{B}_{ij}$ is the $N_{ij} \times p$ orthogonalised B-spline basis matrix, $\boldsymbol{\theta}_\mu$ is a p-length vector of fitted spline coefficients for the grand mean function $\mu(t)$, $\boldsymbol{\theta}_{\alpha_k}$ is a p-length vector of fitted spline coefficients for the k-th principal component function at the transcript level and $\boldsymbol{\theta}_{\beta_{il}}$ is a p-length vector of fitted spline coefficients for the l-th principal component function at the replicate level.

Defining the $p \times K$ matrix $\boldsymbol{\Theta}_\alpha = [\boldsymbol{\theta}_{\alpha_1} \cdots \boldsymbol{\theta}_{\alpha_K}]$ and the $p \times L_i$ matrix $\boldsymbol{\Theta}_{\beta_{ij}} = [\boldsymbol{\theta}_{\beta_{i1}} \cdots \boldsymbol{\theta}_{\beta_{iL_i}}]$ allows the summations to be simplified using matrix algebra as

$$\boldsymbol{y}_{ij} = \boldsymbol{B}_{ij}\boldsymbol{\theta}_\mu + \boldsymbol{B}_{ij}\boldsymbol{\Theta}_{\alpha_k}\boldsymbol{\alpha}_i + \boldsymbol{B}_{ij}\boldsymbol{\Theta}_{\beta_i}\boldsymbol{\beta}_{ij} + \boldsymbol{\epsilon}_{ij} \tag{2}$$

where $\boldsymbol{\alpha}_i = [\alpha_{i1} \cdots \alpha_{iK}]^T$ is the K-length vector formed by collecting all of the α_{ik} terms together and similarly for $\boldsymbol{\beta}_{ij}$. By collecting the observations on all replicates $j = 1, \cdots, n_i$ for transcript i, in the vector $\boldsymbol{y}_i = [\boldsymbol{y}_{i1} \cdots \boldsymbol{y}_{in_i}]^T$, we can write

$$\boldsymbol{y}_i = \boldsymbol{B}_i\boldsymbol{\theta}_\mu + \boldsymbol{B}_i\boldsymbol{\Theta}_\alpha\boldsymbol{\alpha}_i + \widetilde{\boldsymbol{B}_i}\widetilde{\boldsymbol{\Theta}_{\beta_i}}\boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i$$

where $\boldsymbol{B}_i = [\boldsymbol{B}_{i1}^T \cdots \boldsymbol{B}_{in_i}^T]^T$ is the $N_i \times p$ basis matrix formed by stacking each $\boldsymbol{B}_{ij}$ matrix on top of one another. There are n_i such matrices, each with N_{ij} rows and so the matrix $\boldsymbol{B}_i$ has N_i rows where $N_i = \sum_{i=1}^{n_i} N_{ij}$ is the total number of observations on transcript i across all replicates. In contrast, the matrix $\widetilde{\boldsymbol{B}_i} = \text{diag}(\boldsymbol{B}_{i1}, \cdots, \boldsymbol{B}_{in_i})$ is a block diagonal matrix of dimension $N_i \times (n_i p)$ where the blocks correspond to the $\boldsymbol{B}_{ij}$ matrices. Similarly, $\widetilde{\boldsymbol{\Theta}_{\beta_i}} = \text{diag}(\boldsymbol{\Theta}_{\beta_i}, \cdots, \boldsymbol{\Theta}_{\beta_i})$ is a block diagonal matrix of dimension $(n_i L_i) \times (n_i L_i)$ where each block is identical and equal to $\boldsymbol{\Theta}_{\beta_i}$. Finally, $\boldsymbol{\beta}_i = [\boldsymbol{\beta}_{i1} \cdots \boldsymbol{\beta}_{in_i}]^T$ and $\boldsymbol{\epsilon}_i = [\boldsymbol{\epsilon}_{i1} \cdots \boldsymbol{\epsilon}_{in_i}]^T$.

3 Parameter Estimation

In order to estimate the spline coefficients $\boldsymbol{\theta}_{\boldsymbol{\mu}}$, $\boldsymbol{\Theta}_\alpha$ and $\boldsymbol{\Theta}_{\beta_i}$, $i = 1, \cdots, M$ where M is the total number of transcripts in the data set, standard practice with existing methods is to first assume that the principal component loadings α_{ik} and β_{ijl} are multivariate normally distributed, and then to treat them as missing data and employ the EM algorithm (Zhou et al., 2010). In our experience of working with real genomics data sets however, the transcript level loadings α_{ik} are *not* normally distributed, as can clearly be seen in the histograms given in Figure 2, which display many outliers and varying degrees of skewness. The high levels of kurtosis arise because many thousands of transcripts in the data set are more or less flat over the range of the time course and hence have very small loadings. The outliers correspond to those transcripts which exhibit significant changes in expression levels.

In fact, this misspecification can have dramatic consequences for the plausibility of the resulting model fit. We illustrate this issue in Figure 1 where we have plotted model fits obtained under the assumption of normality and under our proposed skew-t-normal alternative for an example transcript, which is typical of many others in the data set.

To deal with this issue, we propose to instead adopt the assumption that the transcript level loadings follow a skew-t-normal distribution, which is flexible

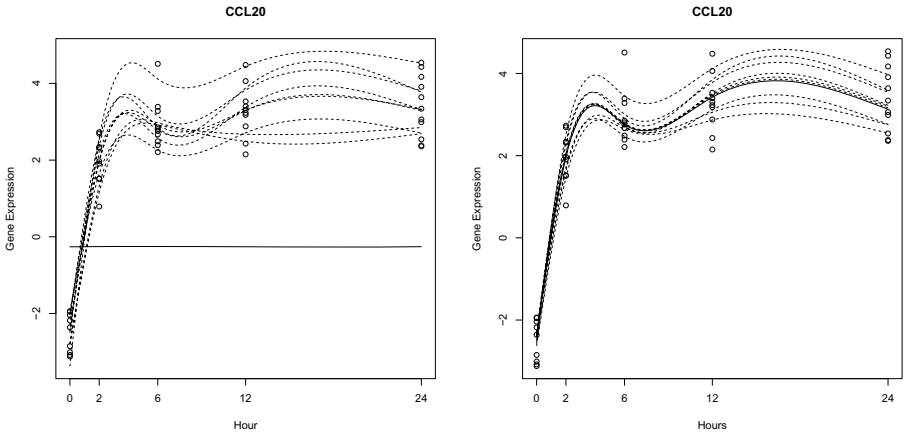


Fig. 1. Plots illustrating an unexpected phenomenon we observed when fitting a Gaussian multi-level reduced-rank functional PCA model to a real data set. Transcript mean curves are given as solid lines, replicate-level curves as dashed lines. The Gaussian model fit, on the left hand side, has produced an implausible transcript mean curve which is flat over the range of the time course and ignores the underlying temporal behaviour suggested by the single time point observations. On the other hand, the replicate-level curves fit the data well. This phenomenon is a result of the extreme departure from normality that genomics data sets exhibit as described in the main text. On the right hand side we show the model fit obtained under our proposed skew-t-normal model which produces a much more plausible transcript-level curve. This example is typical of other transcripts in the data set.

enough to account for the heterogenous departures from normality. Formally, we assume that:

$$\begin{gathered} \alpha_{ik} \overset{\text{i.i.d.}}{\sim} StN(\xi_{\alpha_k}, \sigma^2_{\alpha_k}, \lambda_{\alpha_k}, \nu_{\alpha_k}) \\ \alpha_{ik} \perp \alpha_{ik'}, k \neq k' \quad \alpha_{ik} \perp \alpha_{i'k}, i \neq i' \\ E[\alpha_{ik}] = 0 \end{gathered} \tag{3}$$

where $\perp$ denotes independence. We retain the assumption that $\boldsymbol{\beta}_{ij}$ and $\boldsymbol{\epsilon}_{ij}$ are multivariate normally distributed with zero means and covariance matrices $\boldsymbol{D}_{\beta_i}$ and $\sigma_i^2 \boldsymbol{I}$ respectively. $z \sim StN(\xi, \sigma^2, \lambda, \nu)$ denotes that the random variable z follows a skew-t-normal distribution (Gómez et al., 2007) where ξ is a location parameter, σ^2 is a scale parameter, λ is a skewness parameter and ν is the degrees of freedom controlling the kurtosis. The density of z is given by

$$f(z|\xi, \sigma^2, \lambda, \nu) = 2t_\nu(z; \xi, \sigma^2)\Phi\left(\frac{z-\xi}{\sigma}\lambda\right) \tag{4}$$

where $t_\nu(z; \xi, \sigma^2)$ denotes the Student-t density with ν degrees of freedom, location parameter ξ and scale parameter σ^2, and Φ denotes the normal cumulative distribution function.

It can be shown that under these distributional assumptions, the parameter estimates which maximise the complete log likelihood are

$$\widehat{\boldsymbol{\theta}_\mu} = (\sum_{i=1}^{M} \sigma_i^{-2} \boldsymbol{B}_i^T \boldsymbol{B}_i)^{-1} \sum_{i=1}^{M} \sigma_i^{-2} \boldsymbol{B}_i^T \left[\boldsymbol{y}_i - \boldsymbol{B}_i \boldsymbol{\Theta}_\alpha \boldsymbol{\alpha}_i - \widetilde{\boldsymbol{B}_i} \widetilde{\boldsymbol{\Theta}_{\beta_i}} \boldsymbol{\beta}_i \right]$$

$$\widehat{\boldsymbol{\theta}_{\alpha_k}} = (\sum_{i=1}^{M} \frac{\alpha_{ik}^2}{\sigma_i^2} \boldsymbol{B}_i^T \boldsymbol{B}_i)^{-1} \times \sum_{i=1}^{M} \frac{\alpha_{ik}}{\sigma_i^2} \boldsymbol{B}_i^T \left[\boldsymbol{y}_i - \boldsymbol{B}_i \boldsymbol{\theta}_\mu - \boldsymbol{B}_i \sum_{k' \neq k} [\boldsymbol{\theta}_{\alpha_k} \alpha_{ik'}] - \widetilde{\boldsymbol{B}_i} \widetilde{\boldsymbol{\Theta}_{\beta_i}} \boldsymbol{\beta}_i \right]$$

$$\widehat{\boldsymbol{\theta}_{\beta_{il}}} = (\sum_{j=1}^{n_i} \beta_{ijl}^2 \boldsymbol{B}_{ij}^T \boldsymbol{B}_{ij})^{-1} \times \sum_{j=1}^{n_i} \beta_{ijl} \boldsymbol{B}_{ij}^T \left[\boldsymbol{y}_{ij} - \boldsymbol{B}_{ij} \boldsymbol{\theta}_\mu - \boldsymbol{B}_{ij} \boldsymbol{\Theta}_\alpha \boldsymbol{\alpha}_i - \boldsymbol{B}_{ij} \sum_{l' \neq l} [\boldsymbol{\theta}_{\beta_i l'} \beta_{ijl'}] \right]$$

$$[\widehat{\boldsymbol{D}_{\beta_i}}]_{ll} = \frac{1}{n_i} \sum_{j=1}^{n_i} \beta_{ijl}^2$$

$$\widehat{\sigma_i^2} = \frac{1}{N_i} \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i$$

where $\widehat{\boldsymbol{\theta}_{\alpha_k}}$ is the kth column of the matrix $\widehat{\boldsymbol{\Theta}_\alpha}$, similarly for $\widehat{\boldsymbol{\theta}_{\beta_{il}}}$ and $[\widehat{\boldsymbol{D}_{\beta_i}}]_{ll}$ is the lth diagonal element of the matrix $\widehat{\boldsymbol{D}_{\beta_i}}$. For the parameters of the skew-t-normal distributions, these can be estimated using Newton-Raphson (NR) as in Gómez et al. (2007) or the EM algorithm as in Ho and Lin (2010). Alternatively, we have found that a simplex optimisation (Nelder and Mead, 1965) is simpler and works very well in practice.

Under the distributional assumptions given above, the observed likelihood, as the sum of skew-t-normal and multivariate normally distributed random variables, has no known form. As a result of this, the conditional expectations of the sufficient statistics of the maximum likelihood estimators required by the EM algorithm are challenging to obtain. Given that the skew-t-normal distribution admits a convenient hierarchical representation, we suggest the use of the Monte Carlo EM (MCEM) algorithm (Wei and Tanner, 1990) and the Gibbs sampler for drawing samples from the joint conditional distribution $f(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_{ij} | \boldsymbol{y}_i)$ in order to approximate the conditional expectations. Specifically, following Ho and Lin (2010), if $\tau_{ik} \sim \Gamma(\nu_{\alpha_k}/2, \nu_{\alpha_k}/2)$ then

$$\begin{aligned} \gamma_{ik} | \tau_{ik} &\sim TN\left(0, \frac{\tau_{ik} + \lambda_{\alpha_k}^2}{\tau_{ik}}; (0, \infty)\right) \\ \alpha_{ik} | \gamma_{ik}, \tau_{ik} &\sim N\left(\xi_{\alpha_k} + \frac{\sigma_{\alpha_k} \lambda_{\alpha_k}}{\tau_{ik} + \lambda_{\alpha_k}^2} \gamma_{\alpha_k}, \frac{\sigma_{\alpha_k}^2}{\tau_{ik} + \lambda_{\alpha_k}^2}\right) \end{aligned} \tag{5}$$

where $TN(\mu, \sigma^2; (a, b))$ denotes the truncated normal distribution lying within the interval (a, b). It can then be shown that

$$\gamma_{ik}|\alpha_{ik} \sim TN((\alpha_{ik} - \xi_{\alpha_k})\frac{\lambda_{\alpha_k}}{\sigma_{\alpha_k}}, 1; (0, \infty)) \tag{6}$$

$$\tau_{ik}|\alpha_{ik} \sim \Gamma\left(\frac{\nu_{\alpha_k} + 1}{2}, \frac{\nu_{\alpha_k} + (\alpha_{ik} - \xi_{\alpha_k})^2/\sigma^2_{\alpha_k}}{2}\right) \tag{7}$$

Therefore we introduce additional latent variables, $\boldsymbol{\tau}_i = [\tau_{i1} \cdots \tau_{iK}]^T$ and $\boldsymbol{\gamma}_i = [\gamma_{i1} \cdots \gamma_{iK}]^T$, and the target density for the Monte Carlo E-step becomes the joint conditional distribution $f(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\tau}_i, \boldsymbol{\gamma}_i|\boldsymbol{y}_i)$ whose form is still unknown. However, each of the individual conditional distributions follow known forms that are easy to sample from and hence the Gibbs sampler can be used to efficiently generate samples that are approximately distributed according to the target full joint conditional density.

4 Discussion

Another key way that our model differs from that of Zhou et al. (2010), aside from the skew-t-normal assumption for the distribution of the first-level principal component loadings, is that we treat the second-level variance as dependent on the first-level, by fitting a separate covariance matrix $\boldsymbol{D}_{\beta_i}$ for each transcript, as indicated by the i in the subscript. Furthermore, we choose a different number of principal components, L_i, to retain for each transcript. This is well motivated by real data sets, where it is clear to see that the between-replicate variation is transcript specific. We have provided an example of this in Figure 3 and 4 where we have plotted raw data for two transcripts taken from a real data set. Although both plots show data observed on the same replicates, the responses are much more heterogeneous for the transcript corresponding to IFNG than the transcript for TNF.

We highlight the fact that we retain the normality assumption at both the replicate and error levels, as we have found little evidence that it is necessary to change this. Some authors have proposed robust mixed-effects models in which both the random-effects and error terms are jointly skew-t-normally distributed with a common degrees of freedom paramter ν (Ho and Lin, 2010). However, this approach is motivated more by computational tractability than it is justified by the data characteristics. It is attractive because it allows exact analytical expressions for the conditional expectations required for the EM algorithm to be derived; however, there is no justification that the distribution of the errors should have any relation to that of the random-effects, and the assumption of a common degrees of freedom parameter seems overly restrictive, possibly countering any additional flexibility that might be offered by adopting a heavy tailed error distribution.

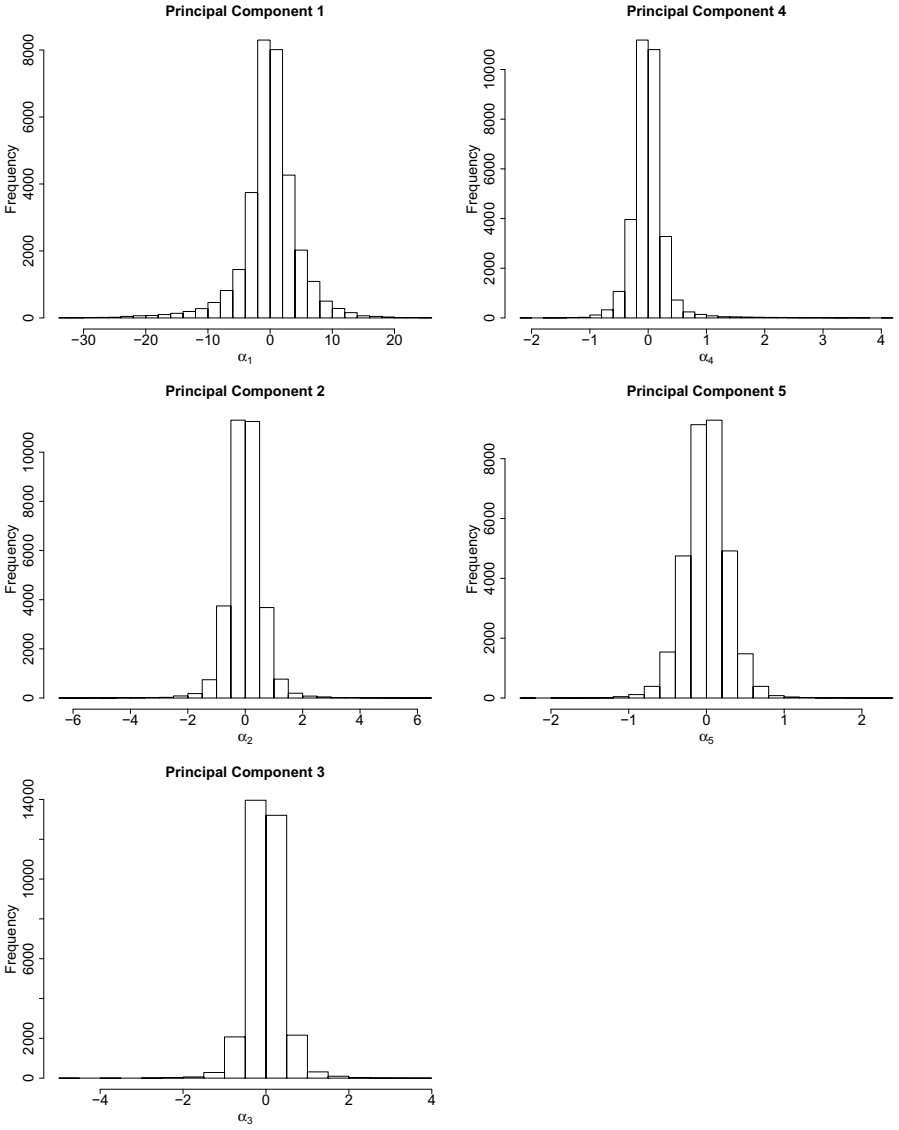


Fig. 2. Initialised transcript-level loadings for our example BCG genomics data set. Note the departure from normality in all instances with many outliers and varying levels of skewness.

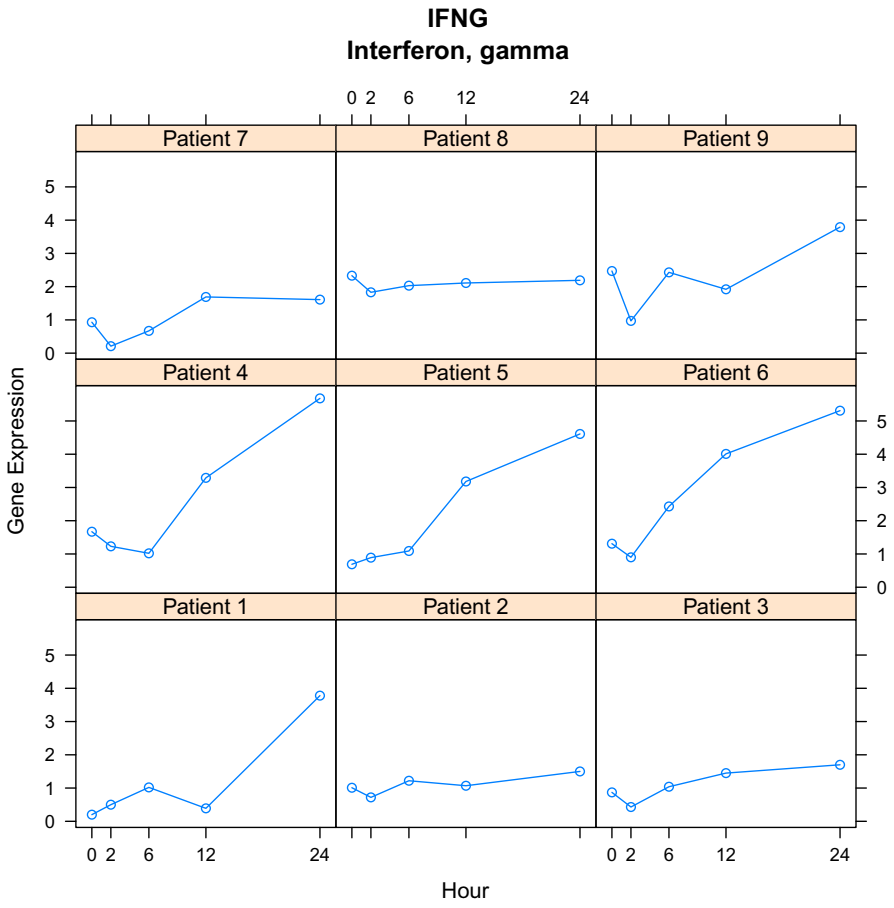


Fig. 3. Raw data for expression levels of interferon-gamma measured in blood samples from nine human patients to which the BCG vaccine for tuberculosis was added. Interferon-gamma is well-known to play an important role in the immune response to tuberculosis infection. Note the heterogeneous response with patients two, three, seven and eight exhibiting flat profiles, while the other patients end the time course with elevated levels of gene expression. Furthermore, these latter patients respond at varying rates, with patient one not responding until after twelve hours, patients four and five responding after six hours, and patient six responding early after only two hours. Compare this to the much more homogeneous profiles for tumour necrosis factor-alpha given in Figure 4.

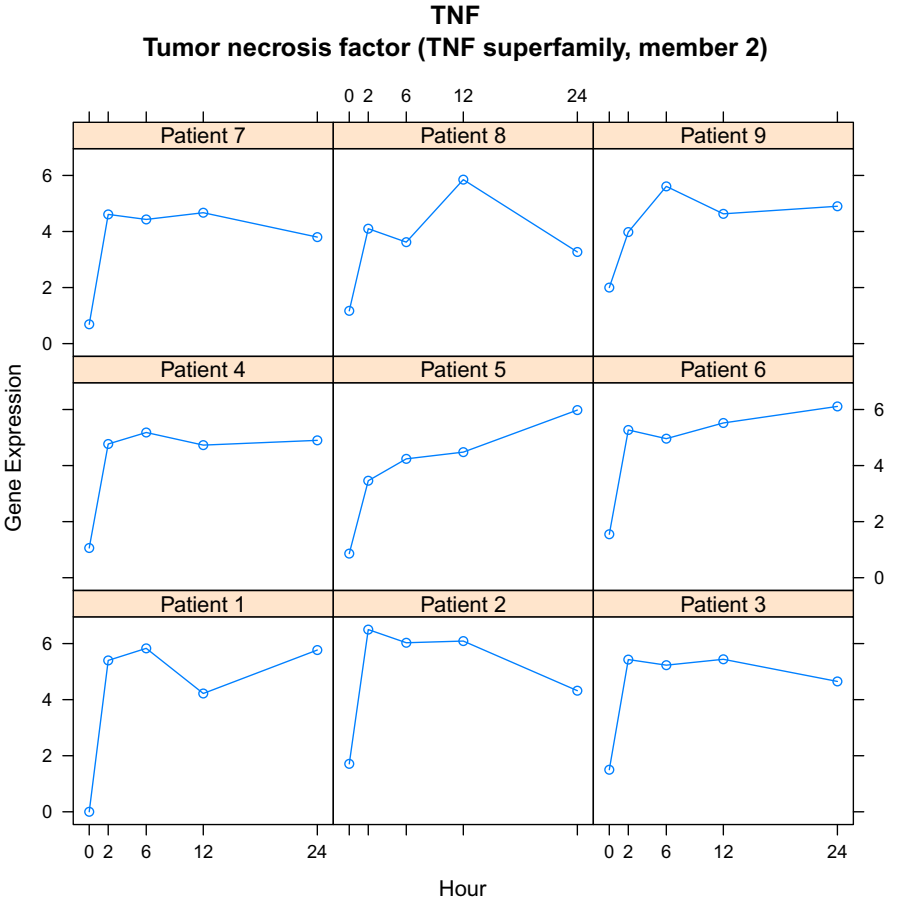


Fig. 4. Raw data for expression levels of tumour necrosis factor-alpha measured in blood samples from nine human patients to which the BCG vaccine for tuberculosis was added. Tumour necrosis factor-alpha is well-known to be associated with tuberculosis infection. Note the subtle differences between the profiles such as the expression level at which the time course begins, the level to which they peak, and whether they plateau, continue to rise or start to fall by the end of the time course. However, compared with the data given in Figure 3 for interferon-gamma, these profiles are much more homogeneous across the different patients.

References

Bar-Joseph, Z., Gerber, G., Simon, I., Gifford, D.K., Jaakkola, T.S.: Comparing the continuous representation of time-series expression profiles to identify differentially expressed genes. Proceedings of the National Academy of Sciences of the United States of America 100(18), 10146–10151 (2003)

Berk, M., Ebbels, T., Montana, G.: A statistical framework for metabolic profiling using longitudinal data. Bioinformatics 27, 1979–1985 (2011)

Di, C., Crainiceanu, C.M., Kuechenhoff, H., Peters, A.: Multilevel functional principal component analysis. Annals of Applied Statistics 3, 458–488 (2009)

Gómez, H.W., Venegas, O., Bolfarine, H.: Skew-symmetric distributions generated by the distribution function of the normal distribution. Environmetrics 18(4), 395–407 (2007)

Ho, H.J., Lin, T.-I.: Robust linear mixed models using the skew t distribution with application to schizophrenia data. Biometrical Journal 52(4), 449–469 (2010)

James, G., Hastie, T., Sugar, C.: Principal component models for sparse functional data. Biometrika 87(3), 587–602 (2000)

Luan, Y., Li, H.: Clustering of time-course gene expression data using a mixed-effects model with B-splines. Bioinformatics 19(4), 474–482 (2003)

Ma, P., Castillo-Davis, C.I., Zhong, W., Liu, J.S.: A data-driven clustering method for time course gene expression data. Nucleic Acids Research 34(4), 1261–1269 (2006)

Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal 7(4), 308–313 (1965)

Storey, J.D., Xiao, W., Leek, J.T., Tompkins, R.G., Davis, R.W.: Significance analysis of time course microarray experiments. Proceedings of the National Academy of Sciences of the United States of America 102(36), 12837–12842 (2005)

Tusher, V.G., Tibshirani, R., Chu, G.: Significance analysis of microarrays applied to the ionizing radiation response. Proceedings of the National Academy of Sciences of the United States of America 98(9), 5116–5121 (2001)

Wei, G.C.G., Tanner, M.A.: A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms. Journal of the American Statistical Association 85(411), 699–704 (1990)

Zhou, L., Huang, J.Z., Carroll, R.J.: Joint modelling of paired sparse functional data using principal components. Biometrika 95(3), 601–619 (2008)

Zhou, L., Huang, J.Z., Martinez, J.G., Maity, A., Baladandayuthapani, V., Carroll, R.J.: Reduced rank mixed effects models for spatially correlated hierarchical functional data. Journal of the American Statistical Association 105(489), 390–400 (2010)

Methodological Foundation for Sign Language 3D Motion Trajectory Analysis

Mehrez Boulares and Mohamed Jemni

Research Laboratory of Technologies of Information and Communication
& Electrical Ingineering (LaTICE)
Ecole Supérieure des Sciences et Techniques de Tunis,
5, Av. Taha Hussein, B.P. 56, Bab Mnara 1008, Tunis, Tunisia
mehrez.boulares@gmail.com, Mohamed.jemni@fst.rnu.tn

Abstract. Current researches in sign language computer recognition, aim to recognize signs from video content. The majority of existing studies of sign language recognition from video-based scenes use classical learning approach due to their acceptable results. HMM, Neural Network, Matching techniques or Fuzzy classifier; are very used in video recognition with large training data. Up to day, there is a considerable progress in animation generation field. These tools contribute to improve the accessibility to information and to services for deaf individuals with low literacy level. They rely mainly on 3D-based content standard (X3D) in their sign language animation. Therefore, signs animations are becoming common. However in this new field, there are few works that try to apply the classical learning techniques for sign language recognition from 3D-based content. The majority of studies rely on positions or rotations of virtual agent articulations as training data for classifiers or for matching techniques. Unfortunately, existing animation generation software use different 3D virtual agent content, therefore, articulation positions or rotations differ from system to other. Consequently this recognition method is not efficient.

In this paper, we propose a methodological foundation for future research to recognize signs from any sign language 3D content. Our new approach aims to provide an invariant to sign position changes method based on 3D motion trajectory analysis. Our recognition experiments were based on 900 ASL signs using Microsoft kinect sensor to manipulate our X3D virtual agent. We have successfully recognized 887 isolated signs with 98.5 recognition rate and 0.3 second as recognition response time.

Keywords: X3D, 3D content Recognition, 3D Motion Trajectory Analysis.

1 Introduction and Motivation

In the world, there are around 70 million people with hearing deficiencies (information from World Federation of the Deaf http://www.wfdeaf.org/). There are varying degrees of deafness: hard of hearing, "profoundly" deaf, and completely deaf. Deaf or profoundly deaf people may wear no hearing aid. Some will be able to lip read and understand you nearly perfectly. But some have problems with verb tenses,

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 67–77, 2012.
© Springer-Verlag Berlin Heidelberg 2012

concordances of gender and number, etc., and they have difficulties when creating a mental image of abstract concepts. Consequently, many communicate with sign language rather than with written and spoken language (Wheatley, M. and Annika P., 2010) [17]. For example, the American Sign Language (ASL) is the primary means of communication for about one-half million deaf people in the US (Charles A. and Rebecca S.) [2]. However, sign Language is a distinct language from spoken languages. In fact, the majority of hard of hearing people have many difficulties in reading written text (Charles A. and Rebecca S.)[2]; consequently, they find it difficult to use technology such as computers. There are some solutions to make accessible textual information by providing a generated animation of sign language based on virtual agent (Kennaway, J. and Glaubert)[9], (Fotinea, S. and al.)[5] and (Stein, D. and al.) [14] works. These systems are based on X3D standard to describe the generated 3D scene. Therefore, signs animations are becoming common.

However, hard of hearing person can create his sign animation using existing generating animation software (Annelies B. and al.]) [1]. Unfortunately, the majority of hearing persons cannot communicate with sign language. Consequently, there is a communication gap with hearing impairment people. Using the recognition process, we can generate automatically the equivalent text according to 3D content animation. Therefore, the animation will be understandable for hearing person. There are many researches in the recognition field. The majority of existing studies rely on sign language recognition from video-based scenes. The machine learning approach used in this domain is not adapted to 3D-based scene or requires a very big training data to provide an efficient recognition result. Moreover, this is due to the consideration of articulations positions as a training data. If we want to apply this solution, we must introduce all different 3D articulations positions to the system and this is a very painful task.

In our context, we rely on the 3D motion trajectory analysis of virtual agent articulations. Moreover, we propose an invariant to position changes solution that contributes on the improvement of the recognition rate from 3D-Based content. In Sign Language, there are many specificities related to sign creation such as hands shape, palms orientation, hands movement and no manual signal that includes Facial expression and body expression. We are based on Microsoft kinect motion capture data for sign creation that allows feature extraction of 20 joints per person. Unfortunately, this tool does not allow hand shape recognition. Moreover, we built an application that allows to user to record and to add sign language specificities according to the 3D recorded motion (as shown in Figure1). Our application enables a real time virtual agent animation from kinect data stream. Based on the 3D recorded scenes, we built an approach invariant to position changes to recognize isolated signs.

Our methodology relies on the 3D motion analysis to extract the signature of the sign. We are based on the 3D non linear regression and polygonal approximation to analyze the 3D motion trajectory. Furthermore, this study aims to learn the trajectory of both hand to our Support Vector Machine and this is more useful compared to hands positions learning.

The remainder of this paper is organized as follows. In section 2, we present some related works. Section 3 is devoted to describe our approach. Section 4 introduces our experimental results. Finally, we conclude by a conclusion and some perspectives.

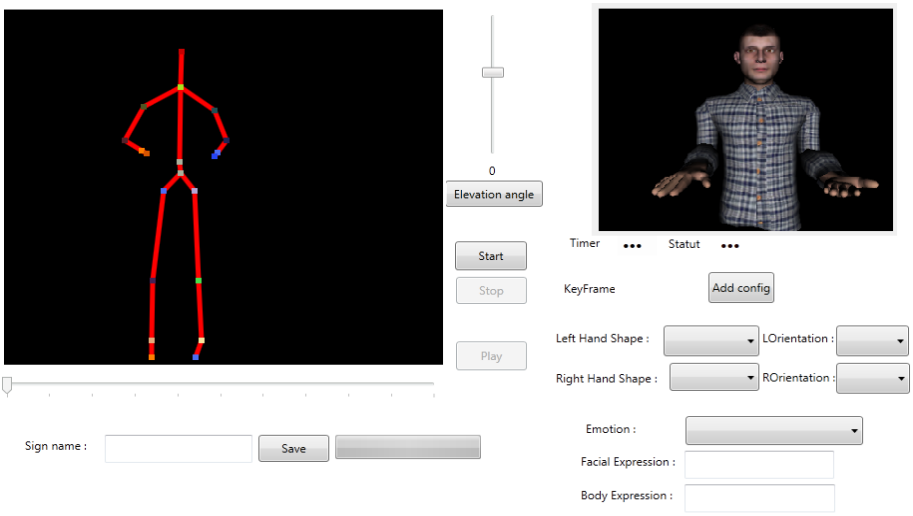


Fig. 1. Our tool to manipulate and to create virtual agent animation using Microsoft Kinect

2 Related Works

Up today, there are many researches in Sign Language recognition. However, these studies rely mainly on two fields. The first one is based on sign recognition from video support. In this domain, (Omer rashid & al.) [13] used Support Vector Machine to recognize static one hand finger spellings of American Sign Language ASL but only static alphabets and numbers in ASL are recognized. (Chung huang & al.) [3] have used SVM to recognize static signs in Taiwanese Sign Language with 89% as average correct recognition rate. Fuzzy classifier with colored gloves was used for the recognition of alphabets of PSL by (Sumaira kausar & al.) [15] . (Yang & al.) [18] use Dynamic programming to recognize ASL. (Kelly & al.) [8] used movement epenthesis and Hidden Markov Model to recognize Eight two handed gesture with 95,6% of detection accuracy and 93.2% of recognition accuracy. (Gao W. & al.) [7] used self-organizing feature maps (SOFM) as different signers feature extractor for continuous hidden Markov models (HMM) with 82.9% word recognition rate over a 5113-sign vocabulary. (Zahoor & al.) [19] performed real time ASL word verification using HMM with 73.72 % as word verification accuracy.

In other word, learning techniques of classifiers, give acceptable results using video support. For this reason, there are a few works which tried to apply these techniques on 3D content. That led to the second field which concerns sign language recognition from 3D content. However, we have used a Longest Common Subsequence method to recognize sign language from 3D based content. This method is based on

the alignment of articulations rotations and this is not efficient. In other meaning, if we change rotation angle, the same sign will be not recognized. In sign language, the sign can be signed by right hand, by left hand or by both hands. However, there are some signs that use both hands but there is no symmetry between right hand rotations and left hand rotations. For instance, in figure 2 the sign BELIEVE can be signed by right hand as dominant hand (Figure 2A) or left hand as dominant hand (Figure 2B). We deduce that the LCS applied to 500 different signs lexicon is not efficient and this is due to the non consideration of sign variation. The LCS applied to 500 lexicon sign cannot reach real time recognition; it depends a lot to alignment time. The recognition process takes quite a long time 20 second to recognize one sign, 42 second to recognize 2 signs to reach 120 second for 7 signs.

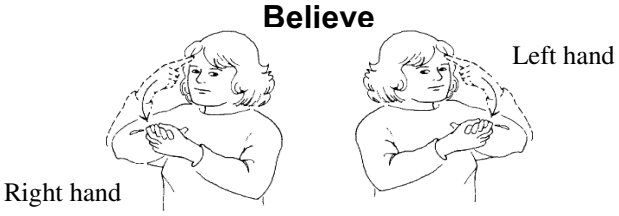


Fig. 2. Left hand and Right hand of ASL sign "BELIEVE"

In general, when only isolated motions are considered and multiple examples for each motion are available, classification can have higher recognition rate than template matching with certain similarity or distance measure (Vapnik 1998) [16] and (Li et al. 2005) [10]. The majority of animation generation systems such as, Vsigns [11] and eSign [4], use rotation angle to animate their avatars. Certainly, for the same sign, rotation angle differ from system to another. Therefore, alignment approach based on rotation will be not useful. However, (Kin Fun Li and al.) [6] work is based respectively on HMM training and matching techniques of 3D skeleton articulation position. These approaches depend on 3D position to create signs. Consequently, for the same sign, the 3D positions of articulations changes from animation to other. Therefore, this technique is very sensitive to 3D position variation and requires a large training data to reach a useful result.

3 Our Approach

Since, classifiers and matching techniques based on articulation positions or rotations are not efficient to recognize the sign motion signature. We propose to analyze the 3D motion trajectory by using approximation techniques of motion data matrices. In sign language, both hands are mainly used to describe the sign gesture. To create signs, the hand motion trajectory can be linear, zigzag, wave, circular or arc. For example, the ASL movement of LETTER is described as bring right hand down with linear movement ending with thumb on left palm. The ASL sign OPEN makes arc movement, ATTITUDE makes small circular movement, PIZZA makes zigzag movement hand

down, etc. Moreover, to analyze the different signs motions, we rely on 3D non linear regression and polygonal approximation to extract the major geometric structures of motion data matrices. Classification is then applied to recognize/reject the original test motion candidates. We choose support vector machine (SVM) classifiers for classification, considering that SVM is one of the most powerful classification techniques (Vapnik 1998; Platt 2000) [16] [12].

3.1 3D Non Linear Regression

In Sign language, motion trajectory of hand is related generally to complex movement. Therefore, we use the 3D non linear regression to extract the major geometric structures of motion data matrices. Our motion data can be represented by matrices with 3 columns (X, Y, Z) and multiple rows (as shown in Table1).

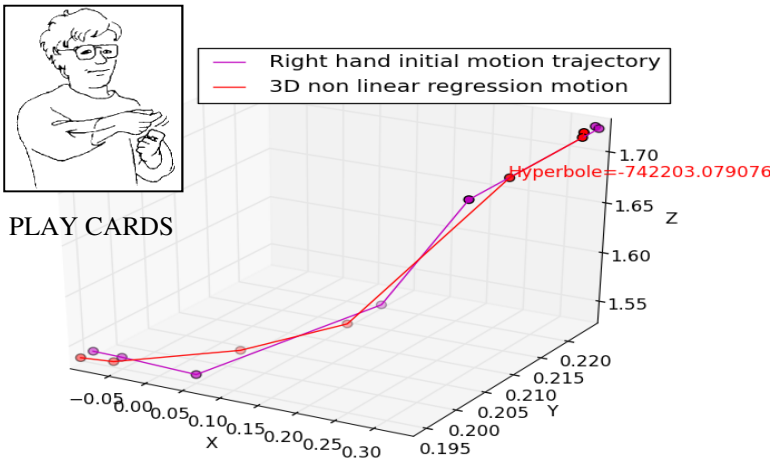


Fig. 3. Original and approximation curve of ASL sign "PLAY CARDS" using right hand motion

Table 1. "PLAY CARDS" initial data matrix of right hand 3D position

M	X	Y	Z
P1	0.34120460	0.22410810	1.7277089
P2	0.34011780	0.22498119	1.7229939
P3	0.27939429	0.21080360	1.6885829
P4	0.13962289	0.21463050	1.5534800
P5	0.04621779	0.19547529	1.5337940
P6	-0.06523611	0.19663230	1.531191
P7	-0.09348568	0.1954235	1.537533

In our approach, we choose conic as approximation curve of motion data. We tried to find the conic nature and the conic parametric equation. However, we start by calculating the Eigen values and the Eigen vectors of the M Matrix.

$[E] = [M]^T [M]$; Eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and Eigenvectors v_1, v_2, v_3

$$\vec{V_1} = \begin{bmatrix} v_{1,1} \\ v_{1,2} \\ v_{1,3} \end{bmatrix} ; \vec{V_2} = \begin{bmatrix} v_{2,1} \\ v_{2,2} \\ v_{2,3} \end{bmatrix} ; \vec{V_3} = \begin{bmatrix} v_{3,1} \\ v_{3,2} \\ v_{3,3} \end{bmatrix} \quad : \quad [V] = \begin{bmatrix} v_{1,1} v_{2,1} v_{3,1} \\ v_{1,2} v_{2,2} v_{3,2} \\ v_{1,3} v_{2,3} v_{3,3} \end{bmatrix}$$

The smallest Eigen value in our case is λ_3 therefore $\overrightarrow{v_3}$ is the smallest unit vector perpendicular to the plan P. The regression equation of the plan is:

$$Ax + By + Cz + 1 = 0 \quad ; \quad \begin{cases} A = -\dfrac{a}{aX_m + bY_m + Z_m} \\ B = -\dfrac{b}{aX_m + bY_m + Z_m} \\ C = -\dfrac{1}{aX_m + bY_m + Z_m} \end{cases} ; \quad \begin{cases} x = v_{1.1}\alpha + v_{2,1}\beta \\ y = v_{1.2}\alpha + v_{2,2}\beta \\ z = v_{1.3}\alpha + v_{2,3}\beta \end{cases}$$

with α and β are the coordinates of (x, y, z) point in axes system $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ .The distance between a given point $p_k(x_k, y_k, z_k)$ and the plane P, is the projection of $\overrightarrow{M_{p_k}}$ on $\overrightarrow{v_3}$. The sum of square distances $\sum (\overrightarrow{M_{p_k}} . \overrightarrow{V_3})^2 = \lambda_3$.

Optimizing the conic equation parameters in the plane p; the system V1, V2, V3 is the new basis on which a point (p) is identified by its coordinates (α, β, γ) with the relationship: $\overrightarrow{MP} \begin{cases} x = v_{1.1}\alpha + v_{2.1}\beta + v_{3.1}\gamma \\ y = v_{1.2}\alpha + v_{2.2}\beta + v_{3.2}\gamma = [v]^T . \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} ; \\ z = v_{1.3}\alpha + v_{2.3}\beta + v_{3.3}\gamma \end{cases}$

[$\gamma = 0$ If (p) is on the plan (P)]. In this axes system, the coordinates of the given points (P_k) are calculated: $\begin{bmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{bmatrix} = ([v]^T)^{-1} \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}$; the general equation of conic curve: $k_{0.2}\beta^2 + k_{0.2}\alpha^2 + k_{1.1}\alpha\beta + k_{0.1}\beta + k_{1.0}\alpha + 1 = 0$;

$$\begin{pmatrix} \sum \beta_k^4 ; \sum \alpha_k^2 \beta_k^2 ; \sum \alpha_k \beta_k^3 ; \sum \beta_k^3 ; \sum \alpha_k \beta_k^2 \\ \sum \alpha_k^2 \beta_k^2 ; \sum \alpha_k^4 ; \sum \alpha_k^3 \beta_k ; \sum \alpha_k^2 \beta_k ; \sum \alpha_k^3 \\ \sum \alpha_k \beta_k^3 ; \sum \alpha_k^3 \beta_k ; \sum \alpha_k^2 \beta_k^2 ; \sum \alpha_k \beta_k^2 ; \sum \alpha_k^2 \beta_k \\ \sum \beta_k^3 ; \sum \alpha_k^2 \beta_k ; \sum \alpha_k \beta_k^2 ; \sum \beta_k^2 ; \sum \alpha_k \beta_k \\ \sum \alpha_k \beta_k^2 ; \sum \alpha_k^3 ; \sum \alpha_k^2 \beta_k ; \sum \alpha_k \beta_k ; \sum \alpha_k^2 \end{pmatrix} \begin{pmatrix} k_{0.2} \\ k_{2.0} \\ k_{1.1} \\ k_{0.1} \\ k_{1.0} \end{pmatrix} = - \begin{pmatrix} \sum \beta_k^2 \\ \sum \alpha_k^2 \\ \sum \alpha_k \beta_k \\ \sum \beta_k \\ \sum \alpha_k \end{pmatrix}$$

The Symbolic form of the previous system: $[S][K] = -[\sigma]$; $[K] = -[S]^{-1}[\sigma]$ which gives the coefficients of the conic equation. The parametric equation of the conic in the initial axes system and the conic nature and characteristics:

$$\begin{cases} k_{0.2}\beta^2 + k_{0.2}\alpha^2 + k_{1.1}\alpha\beta + k_{0.1}\beta + k_{1.0}\alpha + 1 = 0 \\ X = X_m + V_{1.1}\alpha + V_{2.1}\beta \\ Y = Y_m + V_{1.2}\alpha + V_{2.2}\beta \\ Z = Z_m + V_{1.3}\alpha + V_{2.3}\beta \end{cases} ; k_{0.2}k_{2.0} - 4(k_{1.1})^2 \begin{cases} > 0 \rightarrow Ellipse \\ = 0 \rightarrow Parabola \\ < 0 \rightarrow Hyperbola \end{cases}$$

In other words, we can approximate motion data according to the conic nature. Therefore, hand movement can take the form of Ellipse, parabola or hyperbola. Complex movement can be formed by combining different conic forms. Figure 3 shows the right hand motion trajectory of the ASL sign "PLAY CARDS". The purple color describes the initial motion trajectory and the red color describes the 3D non-linear regression of the motion trajectory. In our case, the approximation curve of the right hand motion of the sign "PLAY CARDS", takes HYPERBOLA curve form with -742203,079076 as a conic value.

3.2 Multi-window 3D Non-linear Regression

Figure 4 shows a discrepancy between the original curve and the approximation curve of the ASL sign "PLAY". This discrepancy is due to that the motion trajectory of the sign "PLAY" is composed of several curves. However, our approach can be useful if we rely our multi-window 3D non-linear regression application. We are based on the polygonal approximation approach to decompose the motion trajectory to several curves. We apply then the 3D approximation to each sub curve.

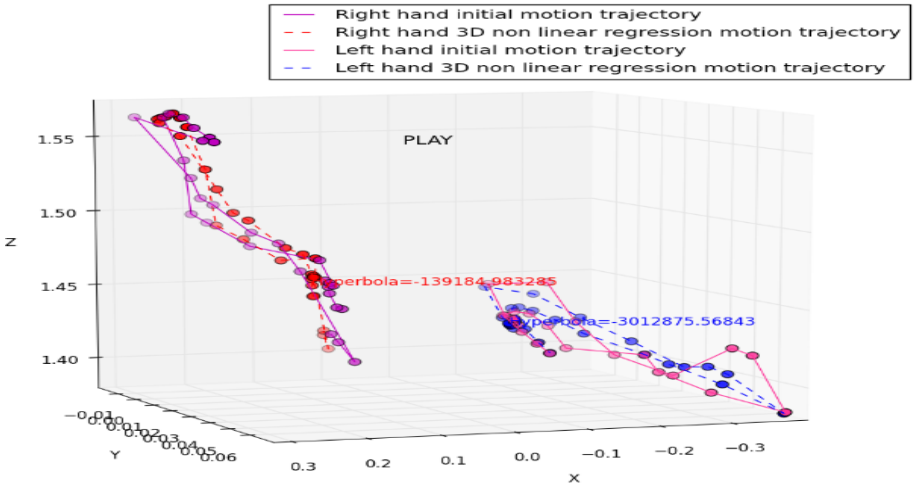


Fig. 4. Motion analysis applied to both right hand and left hand of ASL sign “PLAY”

The polygonal approximation aims to find the greatest distance between the curve and the straight line formed by the first and the last points of the curve. This point is the Max curve (in figure5) which serves to approximate the original curve. Unfortunately using this approach, we have no indication about the curve nature. However, we rely on this idea to decompose our motion trajectory. We propose a multi-window 3D non-linear approximation to analyze motion trajectory. This new approach relies on the application of the 3D curve approximation to the entire motion data matrix. We compare then the Euclidian distance between the original curve and our approximation curve. If this value is greater than the threshold, our algorithm calculates automatically the greatest peak related to this curve (Max curve) and decomposes the motion curve into two sub curves. Our system applies then the 3D non-linear approximation to the two sub curves and repeats this operation recursively until the threshold condition will be verified.

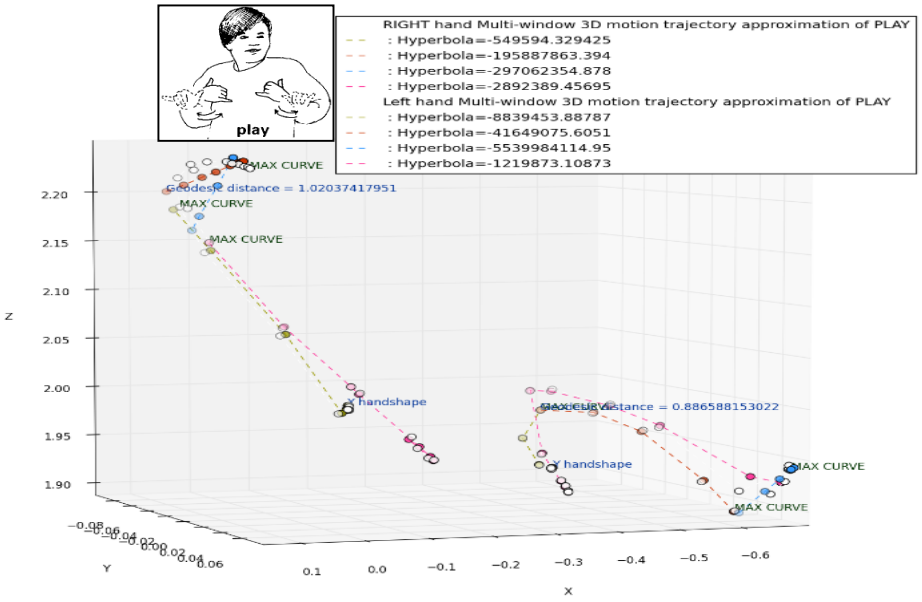


Fig. 5. Our multi-window approximation applied to "PLAY" sign

3.3 Motion Trajectory Classification

SVMs have demonstrated good classification performance and widespread successful uses in many pattern recognition problems. These classifiers rely mainly on the hyper plane optimization that maximizes the margin, or the distance between separating hyper plane and the training examples nearest to the hyper plane. In our context, we rely on SVM to classify signs through data motion. The training data used by SVM is the conic curves obtained through our 3D multi-window non-linear regression. Our algorithm prepares automatically the training data composed by hand shape coordinate (as shown in figure 5, Y hand shape at first 3D coordinates), curves values (shown in figure 5 four hyperbola values for right hand and four values for left hand) and geodesic distance of right and left hand motion.

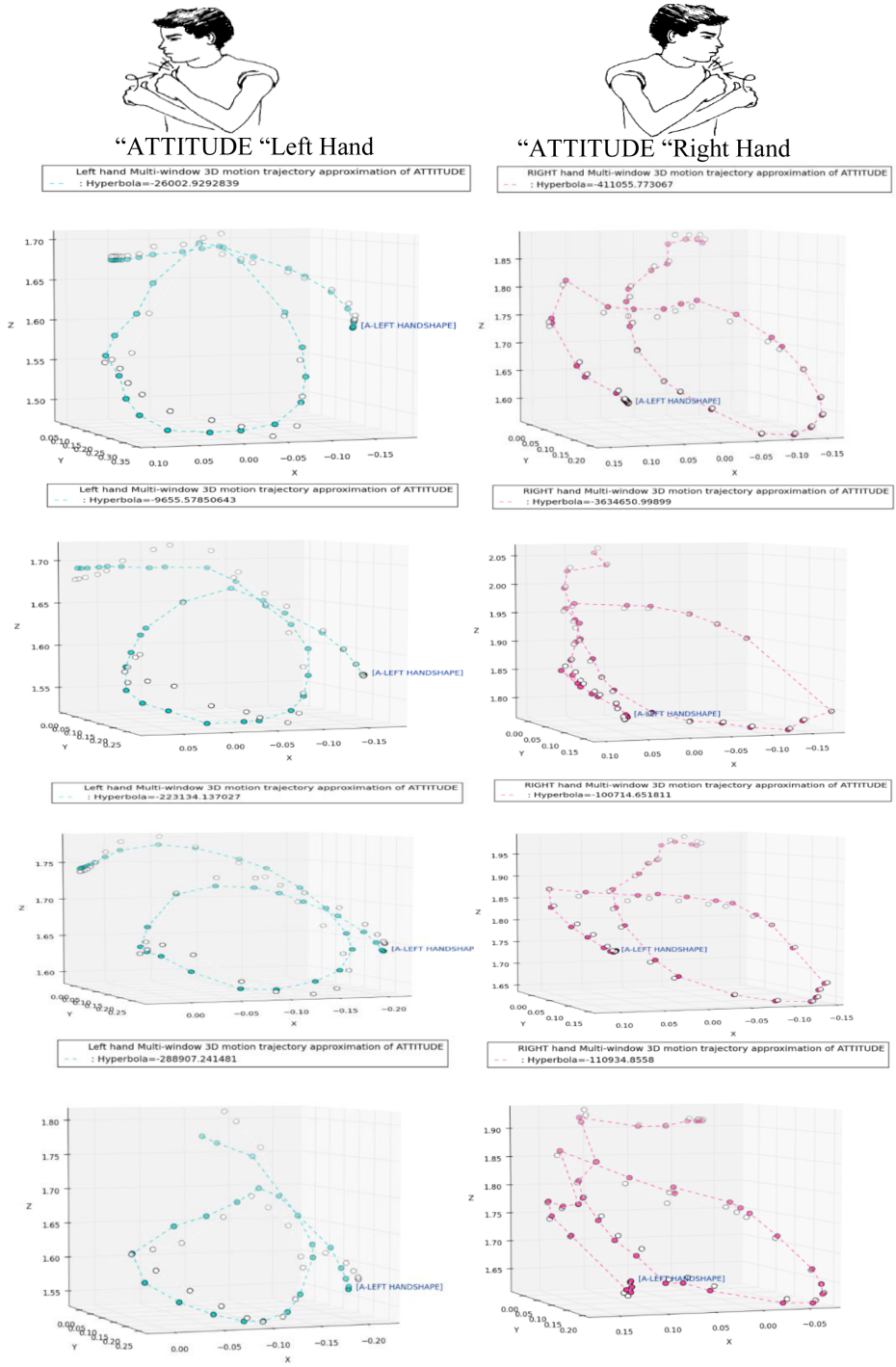


Fig. 6. Curves appearance according to different “ATTITUDE” positions

4 Experiment Evaluation

Figure 6 shows 4 different positions of the ASL sign "ATTITUDE" using the right hand and 4 different positions of the same sign using left hand. In figure 6, white points describe the original motion, blue and pink points are respectively the left hand approximation and the right hand approximation. Moreover, we can see that curves approximation have the same appearance for the same sign "ATTITUDE". Table 2, shows also the curves values variation that is considered as training data for our SVM classifier.

Table 2. Part of SVM training data (Curves values of motion analysis of sign "ATTITUDE")

	Position 1	**Position 2**	**Position 3**	**Position 4**
Right hand	-411055.7	-3634650.9	-100714.6	-110934.8
Left hand	-26002.9	-288907.2	-9655.5	-223134.1

Our SVM training data include motion direction, hand shape, palm orientation (as shown in figure 6 "A" hand shape and "Left" palm orientation at the first location of motion), geodesic distance (as shown in figure 5), motion velocity and motion curves (as shown in table 2). The proposed solution has proven very good results for motion signature extraction. We have tested this approach on 900 ASL signs generated from our tool (shown in figure 1) using Microsoft kinect sensor to manipulate our virtual agent. We have successfully recognized 890 isolated signs with 98.5% recognition rate and 0.3 second as recognition response time compared to our previous results that require 20 seconds to recognize one sign and compared to 73.72% as word recognition rate of (Zahoor zafroula and al.) [19] work.

5 Conclusion and Future Works

The research described in this paper is the first attempt to analyze the motion trajectory in sign language. In other word, this paper has laid a methodological foundation for future research in the 3D-based content recognition process based on data motion analysis. We proposed a 3D motion trajectory approximation approach inspired from 2D hand signature analysis to extract 3D sign signature. Our aim is to provide an efficient recognition which supports sign position changes and allows real time recognition. We managed to achieve 98.5 as a recognition rate using 900 ASL signs with a 0.3 second response time. The ultimate goal of our future research is to provide real time recognition of sign language phrases. Also, we plan to exploit this approach using different generated animation form existing software such as diva, e-sign, and sign smith studio using different sing language.

References

1. Annelies, B., Jean-Paul, S., Jean-Claude, M., Cyril, V.: Diva, une architecture pour le support des agents gestuels interactifs sur internet. Technique et Science Informatiques 29(7), 777–806 (2010)

2. Charles, A., Rebecca, S.: Reading optimally builds on spoken language implication for deaf readers. Learning research and development center University of Pittsburgh (2000)
3. Chung-Lin, H., Bo-Lin, T.: A vision-based Taiwanese sign language Recognition. In: Pattern Recognition (ICPR) Istanbul, Turkey, August 23-27, pp. 3683–3686 (2010)
4. Ehrhardt, U., Davies, B.: A good introduction to the work of the esign project. Esign Deliverable D7-2 (August 2004)
5. Fotinea, S.-E., Efthimiou, E., Caridakis, G., Karpouzis, K.: A knowledge-based sign synthesis architecture. Universal Access in the Information Society 6(4), 405–418 (2008)
6. Fun Li, K., Lothrop, K., Gill, E., Lau, S.: A Web-Based Sign Language Translator Using 3D Video Processing. In: Proceedings of the 2011 14th International Conference on Network-Based Information Systems, NBIS 2011 (2011)
7. Gao, W., Fang, G., Zhao, D., Chen, Y.: A Chinese sign language recognition system based on SOFM/SRN/HMM. Pattern Recognition 37, 2389–2402 (2004)
8. Kelly, D., McDonal, J., Markham, C.: Evaluation of threshold model HMMS and Conditional Random Fields for recognition of spatiotemporal gestures in sign language. In: IEEE 12th International Conference on Computer Vision Workshops, pp. 490–497 (2009)
9. Kennaway, J., Glaubert, J.: Providing signed content on the internet by synthesized animation. ACM Transaction (2007), doi: 10.1145/1279700.1279705
10. Li, C., Prabhakaran, B.: A similarity measure for motion stream segmentation and recognition. In: Proc. of the 6th Inter. Workshop on Multimedia Data Mining (2005)
11. Papadogiorgaki, M., Grammalidis, N., Makris, L., Sarris, N., Strintzis, M.G.: vSIGN project. Communication (September 20, 2002), http://vsign.nl/en/vsignEN.html
12. Platt, J.C.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large Margin Classifiers (2000)
13. Rashid, O., Al-Hamadi, A., Michaelis, B.: Utilizing Invariant Descriptors for Finger Spelling American Sign Language Using SVM. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Chung, R., Hammoud, R., Hussain, M., Kar-Han, T., Crawfis, R., Thalmann, D., Kao, D., Avila, L. (eds.) ISVC 2010, Part I. LNCS, vol. 6453, pp. 253–263. Springer, Heidelberg (2010)
14. Stein, D., Bungeroth, J., Ney, H.: Morpho-syntax based statistical methods for sign language translation. In: Proceedings of the European Association for Machine Translation, Allschwil, Switzerland, pp. 169–177 (2006)
15. Sumaira, K., Younus, M.: Recognition of gestures in Pakistani sign language using fuzzy classifier. In: 8th Conference on Signal Processing, Computational Geometry and Artificial Vision 2008, Rhodes, Greece, pp. 101–105 (2008)
16. Vapnik, V.N.: Statistical Learning theory. Wiley, New York (1998)
17. Wheatley, M., Pabsch, A.: Sign Language in Europe. In: 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies, pp. 251–255. LREC, Malta (2010)
18. Yang, R., Sarkar, S., Loeding, B.: Handling Movement Epenthesis and Hand Segmentation Ambiguities in Continuous Sign Language Recognition Using Nested Dynamic Programming. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(3), 462–477 (2012)
19. Zafrulla, Z., Brashear, H., Starner, T., Hamilton, H., Presti, P.: American Sign Language Recognition with the Kinect. In: ICMI 2011, Alicante, Spain, November 14–18. ACM (2011) 978-1-4503-0641-6
20. Jemni, M., Ghoul, O.E.: An avatar based approach for automatic interpretation of text to Sign language. In: 9th European Conference for the Advancement of the Assistive Technologies in Europe, San Sebastién, Spain, October 3-5 (2007)
21. Boulares, M., Jemni, M.: Mobile sign language translation system for deaf community. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, p. 37 (2012)

Assembly Detection in Continuous Neural Spike Train Data

Christian Braune[1], Christian Borgelt[2], and Sonja Grün[3,4]

[1] Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
[2] European Centre for Soft Computing
Calle Gonzalo Gutiérrez Quirós s/n, E-33600 Mieres (Asturias), Spain
[3] Institute of Neuroscience and Medicine (INM-6), Research Center Jülich, Germany
[4] Theoretical Systems Neurobiology, RWTH Aachen University, Aachen, Germany
christian.braune@ovgu.de, christian@borgelt.net, s.gruen@fz-juelich.de

Abstract. Since Hebb's work on the organization of the brain [16] finding cell assemblies in neural spike trains has become a vivid field of research. As modern multi-electrode techniques allow to record the electrical potentials of many neurons in parallel, there is an increasing need for efficient and reliable algorithms to identify assemblies as expressed by synchronous spiking activity. We present a method that is able to cope with two core challenges of this complex task: *temporal imprecision* (spikes are not perfectly aligned across the spike trains) and *selective participation* (neurons in an ensemble do not all contribute a spike to all synchronous spiking events). Our approach is based on modeling spikes by influence regions of a user-specified width around the exact spike times and a clustering-like grouping of similar spike trains.

Keywords: spike train, ensemble detection, Hebbian learning, continuous data, multidimensional scaling.

1 Introduction and Motivation

Modern multi-electrode arrays (MEAs) allow to record electrical potentials simultaneously at many positions in a brain area [5]. In the raw recordings *spikes* (i.e. brief, sharp increases in the extracellular electrical potentials recorded by the MEAs, also called *action potentials*) are detected and processed by so-called *spike sorting* [18] into the spikes of individual neurons according to their shapes and amplitudes. The result is formally a set of point processes (lists of spike times, one per neuron), which is called *parallel spike trains*.

Recordings of parallel spike trains are essential if one tries to understand how a (biological) neural network encodes and processes information. One of many competing theories is that information is encoded and processed by *temporal coordination* of spiking activity, in particular, *synchronous spiking activity*, a theory that was introduced by D.O. Hebb [16]. If it is correct, groups of neurons, called *cell assemblies*, can be expected to emit spikes in a fairly synchronized

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 78–89, 2012.
© Springer-Verlag Berlin Heidelberg 2012

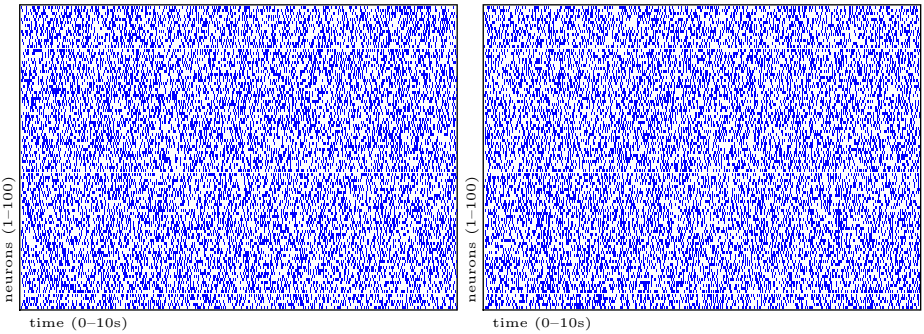


Fig. 1. Two sets of parallel spike trains, one of which shows only random noise (independent stationary Poisson processes, left), while the other (right) exhibits several occurrences of synchronous activity of an assembly of 20 (randomly selected) neurons (stationary Poisson processes with injected coincident spiking events, cf. [14,13,3]).

fashion. In order to be able to test this theory, efficient and reliable algorithms are needed that can detect such synchronous spiking activity in MEA recordings.

To demonstrate that this is a challenging task, Figure 1 shows *dot displays* of two sets of artificially generated parallel spike trains, one of which exhibits synchronous activity: by pure visual inspection it is essentially impossible to discover the assembly, because the 20 assembly neurons are randomly selected from the total of 100 neurons. However, detection algorithms face two even harder challenges, which are not present in the data shown in Figure 1: *temporal imprecision*, that is, spikes cannot be expected to be perfectly aligned across the spike trains, and *selective participation*, that is, the neurons in an assembly cannot all be expected to contribute a spike to all synchronous spiking events of the assembly. The most common approach to deal with the former problem is to discretize the originally continuous signal into time bins of equal length and to consider spikes as synchronous if they lie in the same time bin [12]. This allows further analysis by means of, for example, cross-correlograms [21], with which the correlation of two (discretized) time series is measured and graphically displayed.

However, time binning has severe disadvantages: depending on how the time bin boundaries fall between the spikes, two spikes that are actually very close together in time may be considered as not synchronous (because they end up in different time bins) [12]. On the other hand, spikes that are almost as far apart as the time bin length may still be considered as synchronous if they happen to fall into the same time bin. This can lead to a severe loss and distortion of synchrony information [12]. In this paper, we try to overcome this problem by avoiding time binning. We rather model each spike by an influence region in which other spikes (of other neurons) may be considered as synchronous to it. Based on this model, we then extend the approach presented in [4], which is based on time-binning the data, to a continuous time treatment of spike trains, thus making much better use of the time resolution of the recordings.

2 Related Work

Using pairwise comparisons of neurons or their corresponding spike trains with each other has been shown to yield promising results in several publications over the last years (such as in [10,8,25]).

A classical approach to detect joint spiking patterns in parallel spike trains is the so-called *accretion method* [11], which works with time-binned data. Starting from single neurons, this method iteratively accretes neurons into sequences as long as another neuron shows significantly correlated activity (χ^2 test) with the already accreted neurons. However, accretion suffers from several drawbacks: it works on sequences instead of sets, thus incurring high costs from redundant detections (memory consumption, speed), but at the same time may miss assemblies due to a restriction of the branching factor of the search. Alternatives consist in improved approaches that exploit ideas from frequent item set mining (such as [2]) and enumerate all subsets of neurons as long as a certain quality criterion (e.g. minimum support, significant statistical test result) is met. However, the need to enumerate large numbers of sets of neurons (exponential in the number of neurons in the worst case) still makes them computationally costly.

In order to overcome this problem, we proposed a method that interprets time-binned spike trains as binary vectors in a high-dimensional space (one dimension per time bin) [4]. Dimensionality reduction techniques such as Sammon's Mapping [24] or multi-dimensional scaling [6] onto a single dimension (placing new data points onto the real line such that the pairwise distances of the new points resemble the original distances as closely as possible) yield a sorting criterion for the spike trains. Thus, instead of testing all pairs and subsets of neurons, a single traversal of the sorted set of spike trains suffices to identify assemblies.

Still, this method still suffers from a problem called temporal jitter. The spikes produced by the neurons may neither be perfectly timed nor does the recording process ensure that they are recorded at the same time. The previously mentioned methods can cope with this problem only as long as two originally coincident events fall into the same time bin. In [12] a method is proposed that shifts the spike trains against each other to cope with that problem. Several shifts of integer multiples of the sampling resolution are performed and each time the number of (exact) coincidences is counted. The sum over all these coincidences is used as a measure of synchronicity between two trains. Yet, this approach still requires an the spike train to be binned, making this method prone to effects such as clipping. Also we may argue that although a coincidence may have been found in this way, its significance may be less when the offset between the two original spikes is larger.

A different approach considering the likelihood that a spike has been generated as the result of spikes in different spike trains can be found in [19]. Here a maximum-likelihood approach is used to calculate the influence strengths between several spike trains at once. This comes at the cost of an increased computational need, that we want to avoid.

Table 1. Contingency table for two binary vectors A and B

	B = 0	B = 1	
A = 0	n_{00}	n_{01}	n_{0*}
A = 1	n_{10}	n_{11}	n_{1*}
	n_{*0}	n_{*1}	$N = n_{**}$

Table 2. Different distance measures used for binary vector comparison

Jaccard [17]	$d_{\text{Jaccard}} = \frac{n_{10}+n_{01}}{n_{11}+n_{10}+n_{01}}$
Tanimoto [23,26]	$d_{\text{Tanimoto}} = \frac{2(n_{10}+n_{01})}{n_{11}+n_{00}+2(n_{10}+n_{01})}$
Dice [7]	$d_{\text{Dice}} = \frac{n_{10}+n_{01}}{2n_{11}+n_{10}+n_{01}}$
Correlation [9]	$d_{\text{Correlation}} = \frac{1}{2} - \frac{n_{11}n_{00}-n_{01}n_{10}}{2\sqrt{(n_{10}+n_{11})(n_{01}+n_{00})(n_{11}+n_{01})(n_{00}+n_{10})}}$
Yule [27]	$d_{\text{Yule}} = \frac{n_{01}n_{10}}{n_{11}n_{00}-n_{01}n_{10}}$
Hamming [15]	$d_{\text{Hamming}} = \frac{n_{01}+n_{10}}{n**}$

3 Finding Assemblies in Continuous Spike Train Data

Most of the previously mentioned methods rely on time-binning the spike trains and thus suffer from the disadvantages (loss of synchrony information) pointed out above. To cope with this problem, [22] introduces the notion of so-called *influence maps*: intervals in which a spike may have influence. This approach is similar to convolving spike trains with a kernel and using the overlap of such convolved spike trains as a synchrony measure (see, e.g. [20]), but relies on different kernels and can be generalized to more than two spike trains. The overlap of two influence maps measures the amount or degree of synchrony of the two spikes the influence maps were generated from. The highest degree of synchrony can obviously be achieved by two perfectly aligned spikes. The sum of the degrees of synchrony of individual spikes may be used as an indicator for the correlation between two spike trains. We exploit this idea to improve our approach [4].

In the discrete, time-binned case, the (dis-)similarity between two given spike trains, represented as binary vectors with one element per time bin, is computed from the elements of a 2×2 contingency table (see Table 1). This table records how often both, neither, or only one of two considered neurons produce a spike in a time bin. There is an abundance of (dis-)similarity measures that can be defined on such a table, from which we selected the ones listed in Table 2. They cover some of the most interesting features of common binary distance measures.

3.1 Generalized Contingency Tables

The measures listed in Table 2 are designed for binary vectors. However, none of them actually requires the implied restriction to integer entries in the contingency table. Therefore we may generalize them by computing contingency tables

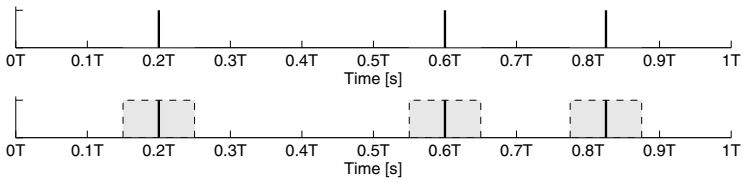


Fig. 2. Continuous spike train with three spikes at $0.2T$, $0.6T$ and $0.825T$, without and with influence maps ($r = 0.05T$)

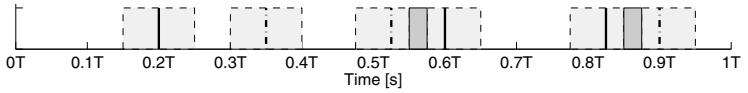


Fig. 3. Two continuous spike trains, one with three spikes at $0.2T$, $0.6T$ and $0.825T$ (solid) and one with spikes at $0.35T$, $0.525T$ and $0.9T$ (dash-dotted). Overlap of influence maps in darker gray.

with real-valued entries by exploiting the influence maps introduced in [22]. The fundamental idea may also be viewed as using binary vectors with a supercountably infinite number of elements, one for each time in the recording period.

Formally, let $\boldsymbol{t} = \{t_1, t_2, \ldots, t_n\}, 0 \leq t_i \leq T$, be a sequence of spike times for a given neuron, where T is the recording period. A user now chooses an influence map, that is, a function $f : \mathbb{R} \to \mathbb{R}_0^+$, which describes an influence region in which spikes of other neurons may be considered as synchronous (possibly only to some degree). In principle, these maps may have any shape, but for reasons of simplicity we confine ourselves here to symmetric rectangular functions of a user-specified width r. That is, we consider $\forall t \in \boldsymbol{t}$:

$$f_t : \mathbb{R} \to \{0, 1\}, \quad x \mapsto f_t(x) = \begin{cases} 1, \text{ if } |x - t| \leq \frac{r}{2}, \\ 0, \text{ otherwise,} \end{cases}$$

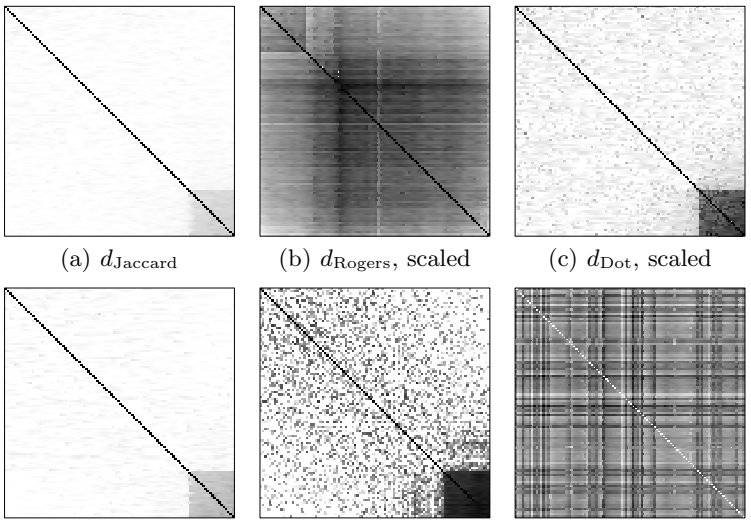
as illustrated in Figure 2. This effectively turns the sequence of spike times into a set of intervals $M_T = \{[a, b] \mid \exists t \in \boldsymbol{t} : a = \max(0, t - \frac{r}{2}) \wedge b = \min(t + \frac{r}{2}, T)\}$. However, we have to take care of the fact that these intervals may overlap, which makes the function view easier to handle. Instead of merging intervals, we combine the functions f_t for all $t \in \boldsymbol{t}$, which yields the function

$$f_{\boldsymbol{t}} : \mathbb{R} \to \{0, 1\}, \quad x \mapsto f_{\boldsymbol{t}}(x) = \max_{t \in \boldsymbol{t}} f_t(x) = \begin{cases} 1, \text{ if } \exists t \in \boldsymbol{t} : |x - t| \leq \frac{r}{2}, \\ 0, \text{ otherwise.} \end{cases}$$

This function describes where spikes of the considered neuron have influence and can be considered synchronous with spikes of other neurons.

Consider now two neurons a and b with their sets $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ of spike times, which give rise to functions $f_{\boldsymbol{t}_a}$ and $f_{\boldsymbol{t}_b}$. From these we derive the four functions

$$f_{ab}^{(i,j)} : \mathbb{R} \to \{0, 1\}, \quad x \mapsto f_{ab}^{(i,j)}(x) = \begin{cases} 1, \text{ if } f_{\boldsymbol{t}_a} = i \text{ and } f_{\boldsymbol{t}_b} = j, \\ 0, \text{ otherwise,} \end{cases}$$



(a) d_{Jaccard} (b) d_{Rogers}, scaled (c) d_{Dot}, scaled

(d) $d_{\text{Correlation}}$ (e) d_{Yule} (f) d_{Hamming}, scaled

Fig. 4. Distance matrices obtained from the respective distance measures. Distances are encoded on a gray scale, where items have a lower distance the blacker the corresponding square. 100 neurons with an injected assembly of size 20. Data points have been pre-sorted such that the assembly appears in the lower right corner of the plot. Distance measures marked as *scaled* have $d_{\min} = 0$ and $d_{\max} = 1.0$.

for all $(i, j) \in \{0, 1\}^2$, which describe whether the argument x is inside the influence region of a spike for both (if $(i, j) = (1, 1)$), neither (if $(i, j) = (0, 0)$) or only one of the neurons (if $(i, j) = (0, 1)$ or $(i, j) = (1, 0)$). An example for $(i, j) = (1, 1)$ is shown in Figure 3. Intuitively, these four functions indicate where in a (infinitely dimensional) binary vector with one element for each $x \in [0, T]$ we have a situation that corresponds to an element of the 2×2 contingency table shown in Table 1. Note that for all $x \in [0, T]$ exactly one of the four functions $f_{ab}^{(i,j)}(x)$, $(i, j) \in \{0, 1\}^2$, has the value 1, while the other three are 0. Therefore we can easily derive the elements of a generalized contingency table as

$$n_{ij} = \frac{1}{r} \int_0^T f_{ab}^{(i,j)}(x) dx$$

for all $(i, j) \in \{0, 1\}^2$. Note that it is $n_{00} + n_{01} + n_{10} + n_{11} = n_{**} = \frac{T}{r}$. This shows how one can solve the counting problem, which consists in the fact that the entries of a standard contingency table are counters of vector elements, but we cannot count elements of a binary vector with super-countably many elements. Note that the solution chosen here is in line with a time-binned approach, where the width of the time bins, which corresponds to the width of the influence maps in our approach, determines the total count n_{**} in exactly the same way.

3.2 Evaluating Distance Measures

Being able to efficiently calculate distances between continuous spike trains, we can evaluate the distance measures. As we want to discriminate between neurons belonging to an assembly and those that exhibit only random noise, we take a first look at plots of the distance matrices. In these plots large distances correspond to whiter cells, while darker cells indicate a high similarity between two data points. Figure 4 shows example plots for different distance measures.

As can be seen, only Jaccard, Dot, Correlation and Yule allow for a (visually) clear discrimination of the assembly from the rest of the neurons. In addition, Yule's distance suffers from many small distances between neurons exhibiting random noise, which can impede the detection. The remaining two distance measures show no significant differences between neurons belonging to the injected assembly and other neurons. Obviously the possible values for the distance measure is in both cases not fully exhausted such that a visual inspection may not be sufficient to reject these two distance measure at this stage. But even normalizing the values in the distance matrix so that they lie in the range $[0, 1]$ does not lead to better result. Hence we will only consider the four distance measures Jaccard, Dot, Correlation and Yule in the following.

3.3 Sorting by Non-linear Mapping and Assembly Detection

To find a suitable ordering of the neurons for further testing it is necessary to choose a sorting criterion. In [4] we already proposed to use Sammon's non-linear mapping [24] to reduce the dimensionality of the data set to only one dimension. The resulting scalar can then be used to sort the neurons. In a linear traversal of the sorted list the final tests are performed and the assemblies are detected.

Originally, in the discrete case, we performed χ^2 tests for independence to distinguish between correlated and non-correlated neurons. However, this test is only properly applicable if we have actually countable events like coincidences, which in the discrete case is the number of time bins. Only in this case the χ^2 value is properly interpretable. In our current setup, however, even though we can achieve an analogous sum of the contingency table entries (see Section 3.1), applying the test in the same way appears to be a somewhat dubious procedure.

Similarly, choosing a time resolution (other than the width r of the influence maps) to give a unit to the entries in the contingency table does not appear to be appropriate, since the result of the χ^2 test (whether it is significant or not) depends directly on this choice. Suppose, for instance, that we have a recording of one second length and choose one second as the time unit. As a consequence, no entry in the contingency table can be higher than 1 (if measured in this unit). Thus, for a χ^2 distribution with one degree of freedom, we would have to lower the level of significance to 0.68 in order to be able to reject the null hypothesis of independence. On the other hand, using a millisecond resolution for the same data may enable us to reject the null hypothesis, because of the wider range of possible χ^2 values. Hence we refrain from such an approach.

However, as the calculation of the χ^2 value and the corresponding p-value is only used to accept or reject whether a neurons is correlated with its neighbor

(in the sorted list of neurons), we can also argue that the distance measure itself provides all relevant information. Neurons belonging to the same assembly have lower distances between each other than to the rest of the neurons. This leads to the following simple idea: we sort the neurons w.r.t. the value that is assigned to them by Sammon's non-linear mapping (or some other multi-dimensional scaling method) to one dimension. Then we find the index k of the neuron such that

$$k = \underset{1 \leq i \leq n}{\operatorname{argmax}} |x_i - x_{i+1}|,$$

which indicates the largest gap between two consecutive neurons (all indices refer to the sorted list of neurons). We set the decision boundary in this gap and thus obtain the two sets of neuron indices

$$A = \{i \mid i \geq 1 \wedge i \leq k\} \quad \text{and} \quad B = \{i \mid i > k \wedge i \leq n\}.$$

We then report the set as assembly that has the least average inner-set distance:

$$d_A = \frac{1}{k-1}|x_k - x_1| \quad \text{and} \quad d_B = \frac{1}{n-k-1}|x_n - x_{k+1}|$$

4 Evaluation

To evaluate our method we artificially generate a set of $n = 50$ spike trains (independent stationary Poisson processes with a firing rate of $\lambda = 20$Hz, which is similar to the frequency with which cortical neurons fire) of $T = 10$s length, into which synchronous spiking activity of 10 neurons was injected (coincidence firing rate $\lambda_c = 5$Hz, background rate appropriately adapted to obtain equal overall firing rates of 20Hz). Coincident spikes are copied from a *mother process* into the spike trains with different copy probabilities p. In addition, the spike times are uniformly dithered by ± 5ms (independently for each neuron) to simulate temporal imprecision as it may be present in a real life data set. The width of the influence maps was chosen to be 15ms to be able to capture a jittered spike at least partially if the dithering moved the coincidences far away from each other. The possible outcomes that may occur in our test setup are:

1. The assembly has been correctly detected. No neuron is missing, no additional neuron has been found (depicted as *perfect*). These are true positives.
2. The whole assembly has been found, but at least one additional neuron was found (depicted as *too many*).
3. Only a subset of the assembly's neurons has been found. At least one neuron was missing (depicted as *too few*).
4. Only a subset of the assembly's neurons has been found. At least one neuron was missing. In addition at least one neuron has been found to belong to the assembly that is actually independent (depicted as *too few/too many*).
5. Every neuron that has been reported belongs to the group of independent neurons and not a single neuron of the true assembly has been reported (depicted as *wrong*). These are false positives.

For each test 100 trials were run and the number of each type of error was recorded. The results of these tests are shown in Figure 5.

We obtained no results for the dot product distance measure because during the first run the test was canceled due to the surprisingly bad quality of the results (see Figure 5, bottom) even if the copy probability is one. Further investigation showed that the quality is only bad if the dimensionality reduction is performed to produce only one dimension. Two dimensional plots reveal, that an algorithm which calculates decision boundaries at arbitrary angles would very well be able to distinguish between assembly and noise neurons in case of a dot product distance measure (see Figure 7).

To compare the results obtained with the binned approach we conducted exactly the same experiments for half of the copy probability settings as well. The distance measures used are exactly the same, only that the spike trains are treated as binary vectors. The numbers $n_{ij}, i, j \in \{0, 1\}$ are calculated in the usual way. What we can see is that the binning method is slightly more prone to not finding the whole assembly or considering too many neurons while never reporting any assembly that has to be labelled as *wrong*. The later may be attributed to the fact that spikes that fall into the same time bin are counted as if they were perfectly aligned, thus leading to a higher density of the assembly as such, while the continuous calculation uses a more differentiated view on such coincidences.

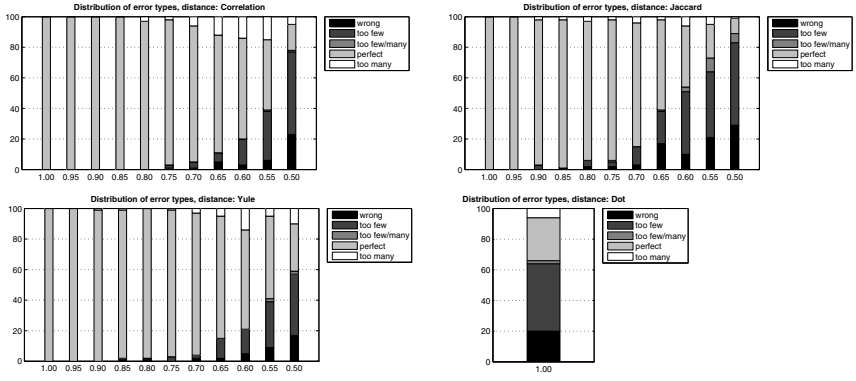


Fig. 5. Assembly detection quality under varying conditions for the selective participation of neurons in synchronous spiking activity. The horizontal axis shows the probability with which a spike is copied from the coincidence mother process. For the dot product as the distance measure, experiments for lower copy probabilities were not conducted, because of the disappointing results for a copy probability of 1.

In contrast the method proposed by used is able to detect the assembly as a whole even when the copy probability is decreasing. Perfect findings are much more common but only at the cost of reports of totally *wrong* assemblies. Having a closer look at those false positives revealed that these were mostly single noise

neurons that were so different from all other neurons that the multidimensional scaling placed them farther apart from the rest of all neurons than the gap between assembly and noise neurons was.

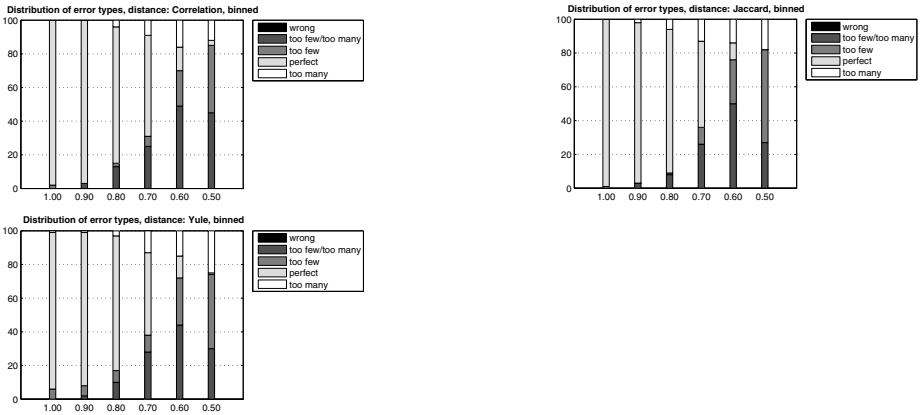


Fig. 6. Assembly detection quality under varying conditions for the selective participation of neurons in synchronous spiking activity. The horizontal axis shows the probability with which a spike is copied from the coincidence mother process. The spike trains have undergone a binning before being analysed with the same procedure to compare results.

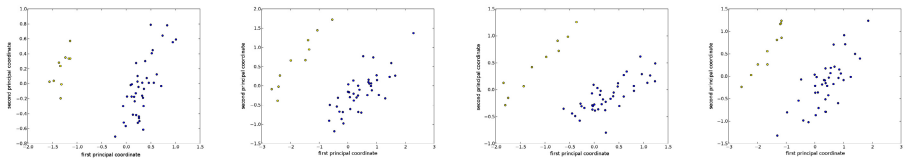


Fig. 7. Two dimensional plots of 50 neurons, assembly of size 10 (yellow) and dot product as distance measure. Classification has been performed with only the x-axis available. The upper left image shows the case where an assembly was completely found. The upper right image shows the case, where only a single (noise) neuron was reported. The lower left image shows the case, where some neurons were missed. The lower right image shows the case, where too many neurons were found. On the right hand side the distribution of error types for a copy probability of 1.0 is shown.

5 Conclusion and Future Work

In this paper we presented a simple and efficient method for detecting assemblies of neurons in continuous spike train data, which avoids the otherwise almost exclusively applied time binning. As a consequence, the loss and distortion of synchrony information resulting from binning the data can be avoided. The additional challenges of *temporal imprecision* (imperfect spike alignment) and *selective participation* (copy probabilities less than 1) can be handled with this

method while still recovering most of the assemblies at least partially. Although the performance of our algorithm is fairly good, there are still many open problems that need to be addressed. In the first place, additional tests on different types of data need to be performed. Compared to algorithms that work on discretized data the rate of false positives is significantly higher. This is partly due to the multi-dimensional scaling that increases the likelihood of neighboring data points having a small distance to each other. On the other hand a lot of information about the structure of the data is lost, if multi-dimensional scaling to only one dimension is performed. First tests have shown that this one dimension contains about three times as much information as the next best dimensions, but keeping two or more dimensions and finding assemblies with common clustering methods or support vector machines may be worth looking into.

Acknowledgments. This work was partly supported by Helmholtz Alliance on Systems Biology, European Union (FP7-ICT-2009-6, BrainScales).

References

1. Allen, J.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
2. Berger, D., Borgelt, C., Diesmann, M., Gerstein, G., Grün, S.: An accretion based data mining algorithm for identification of sets of correlated neurons. In: 18th Annual Computational Neuroscience Meeting: CNS 2009, vol. 10(suppl. 1), pp. 18–23 (2009)
3. Berger, D., Borgelt, C., Louis, S., Morrison, A., Grün, S.: Efficient identification of assembly neurons within massively parallel spike trains. Computational Intelligence and Neuroscience, Article ID 439648 (2010), doi: 10.1155/2010/439648
4. Braune, C., Borgelt, C., Grün, S.: Finding Ensembles of Neurons in Spike Trains by Non-linear Mapping and Statistical Testing. In: Gama, J., Bradley, E., Hollmén, J. (eds.) IDA 2011. LNCS, vol. 7014, pp. 55–66. Springer, Heidelberg (2011)
5. Buzsáki, G.: Large-scale Recording of Neuronal Ensembles. Nature Neuroscience 7, 446–461 (2004)
6. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling, 2nd edn. Chapman and Hall, London (2000)
7. Dice, L.R.: Measures of the amount of ecologic association between species. Ecology 26, 297–302 (1945)
8. Diekman, C.O., Sastry, P.S., Unnikrishnan, K.P.: Statistical significance of sequential firing patterns in multi-neuronal spike trains. Journal of Neuroscience Methods 182(2), 279–284 (2009)
9. Edwards, A.: An introduction to linear regression and correlation. WH Freeman, New York (1984)
10. Feldt, S., Waddell, J., Hetrick, V.L., Berke, J.D., Zochowski, M.: Functional clustering algorithm for the analysis of dynamic network data. Physical Review E 79(5), 056104-1–056104-9 (2009)
11. Gerstein, G., Perkel, D., Subramanian, K.: Identification of functionally related neural assemblies. Brain Research 140(1), 43–62 (1978)
12. Grün, S., Diesmann, M., Grammont, F., Riehle, A., Aertsen, A.: Detecting unitary events without discretization of time. Journal of Neuroscience Methods 94, 67–79 (1999)

13. Grün, S., Abeles, M., Diesmann, M.: Impact of Higher-Order Correlations on Coincidence Distributions of Massively Parallel Data. In: Marinaro, M., Scarpetta, S., Yamaguchi, Y. (eds.) Dynamic Brain - from Neural Spikes to Behaviors. LNCS, vol. 5286, pp. 96–114. Springer, Heidelberg (2008)
14. Grün, S., Diesmann, M., Aertsen, A.: Unitary events in multiple single-neuron activity. I. Detection and significance. Neural Computation 14(1), 43–80 (2002)
15. Hamming, R.: Error detecting and error correcting codes. Bell Systems Tech. Journal 29, 147–160 (1950)
16. Hebb, D.: The organization of behavior. J. Wiley & Sons, New York (1949)
17. Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin de la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
18. Lewicki, M.: A review of methods for spike sorting: The detection and classification of neural action potentials. Network 9(4), R53–R78 (1998)
19. Masud, M.S., Borisyuk, R.: Statistical technique for analysing functional connectivity of multiple spike trains. Journal of Neuroscience Methods 196(1), 201–219 (2011)
20. Pazienti, A., Maldonado, P., Diesmann, M., Grün, S.: Effectiveness of systematic spike dithering depends on the precision of cortical synchronization. Brain Research 1225, 39–46 (2008)
21. Perkel, D.H., Gerstein, G.L., Moore, G.P.: Neuronal spike trains and stochastic point processes: II. Simultaneous spike trains. Biophysical Journal 7(4), 419–440 (1967)
22. Picado-Muiño, D., Castro-León, I., Borgelt, C.: Fuzzy frequent pattern mining to identify frequent neuronal patterns in parallel spike trains. Submitted to IDA 2012
23. Rogers, D., Tanimoto, T.: A computer program for classifying plants. Science 132, 1115–1118 (1960)
24. Sammon, J.: A nonlinear mapping for data structure analysis. IEEE Trans. Comput. 18(5), 401–409 (1969)
25. Shmiel, T., Drori, R., Shmiel, O., Ben-Shaul, Y., Nadasdy, Z., Shemesh, M., Teicher, M., Abeles, M.: Temporally precise cortical firing patterns are associated with distinct action segments. Journal of Neurophysiology 96(5), 2645–2652 (2006)
26. Tanimoto, T.: IBM Internal Report (November 17, 1957)
27. Yule, G.: On the association of attributes in statistics. Philosophical Transactions of the Royal Society of London, Series A 194, 257–319 (1900)

Online Techniques for Dealing with Concept Drift in Process Mining

Josep Carmona and Ricard Gavaldà

Universitat Politècnica de Catalunya
Barcelona, Spain

Abstract. *Concept drift* is an important concern for any data analysis scenario involving temporally ordered data. In the last decade *Process mining* arose as a discipline that uses the *logs* of *information systems* in order to mine, analyze and enhance the process dimension. There is very little work dealing with concept drift in process mining. In this paper we present the first online mechanism for detecting and managing concept drift, which is based on *abstract interpretation* and *sequential sampling*, together with recent learning techniques on data streams.

1 Introduction

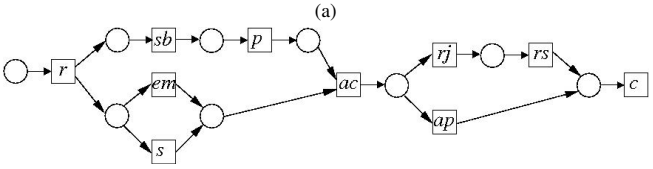
Process Mining is a relatively novel discipline which has received a lot of attention in the last decade [16]. Although it shares many features with Data Mining, it has originated from different concerns and communities, has a set of distinctive techniques, and produces slightly different outcomes. Historically, process mining arises from the observation that many organizations record their activities into *logs* which describe, among others, the real ordering of activities of a given process, in a particular implementation. *Software engineering* techniques have mainly focused on the specification part of the processes within an *information system*. In reality, this may cause a big gap between a system specification's and the final implementation, hampering the use of the models that specify the main processes of an information system. As another example, designers of hardware or embedded, concurrent systems, need to compare behavior and specifications; typically they can passively or actively generate large amounts of logs from their target system and/or their prototypes, so a logical approach is to use these logs for the verification task.

By using the logs as source of information, process mining techniques are meant to *discover*, *analyze,* and *enhance* formal process models of an information system [17]. Process discovery is probably the main and most challenging discipline within process mining: to discover a formal process model (a Petri net [15], an automaton, etc.) that adequately represents the traces in the log. Several process discovery algorithms exist in the literature (the reader can find a good summary in [17]). In this paper, we concentrate on the *control-flow* part, i.e., the causal relations between the events of a process. Let us use an example to illustrate control-flow discovery. The example shown in Figure 1 is taken from [18] and considers

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 90–102, 2012.
© Springer-Verlag Berlin Heidelberg 2012

1	r,s,sb,p,ac,ap,c	8	r,em,s,sb,p,ac,ap,c
2	r,sb,em,p,ac,ap,c	9	r,sb,em,s,p,ac,ap,c
3	r,sb,p,em,ac,rj,rs,c	10	r,sb,em,s,p,ac,rj,rs,c
4	r,em,sb,p,ac,ap,c	11	r,em,sb,p,s,ac,ap,c
5	r,sb,s,p,ac,rj,rs,c	12	r,em,sb,s,p,ac,rj,rs,c
6	r,sb,p,s,ac,ap,c	13	r,em,sb,p,s,ac,ap,c
7	r,sb,p,em,ac,ap,c	14	r,sb,p,em,s,ac,ap,c

(a)



(b)

Fig. 1. Control-flow process discovery: (a) log containing a drift from trace 8 on, (b) Petri net discovered from part of the log (traces 1 to 7)

the process of handling customer orders. In the example, the log contains the following activities: *r=register*, *s=ship*, *sb=send_bill*, *p=payment*, *ac=accounting*, *ap=approved*, *c=close*, *em=express_mail*, *rj=rejected*, and *rs=resolve*. The goal of process discovery is to obtain a formal model such as the Petri net shown in Figure 1(b)[1].

The problem of *concept drift* is well known and well studied in the data mining and machine learning communities, but hardly addressed so far in process mining, where it has important particularities. Three main problems regarding concept drift can be identified in the context of process mining [5]:

1. *Change Detection*: detect when a process change happens. This is the most fundamental problem to solve.
2. *Change Localization and Characterization*: characterize the nature of one particular change, and identify the region(s) of change in a process.
3. *Unraveling Process Evolution*: discover the evolution of process change overall, and how change affects the model over time.

Current process discovery algorithms behave poorly when a log incorporates a drift: causal relations between events may appear and disappear, or even reverse, and therefore cannot be resolved. For instance, in the example of Figure 1, in the first part of the log (traces 1–7) activities *em* and *s* are in conflict, i.e., only one of them can be observed in a process execution. However, from trace 8 on, both activities occur with *em* always preceding *s*. Thus, the whole log contains both behaviors and therefore process discovery techniques fail at determining the causal relationship between *em* and *s*. In that case, no arcs connecting activities *s* and *em* with the rest of activities in the model are discovered.

[1] For the reader not familiar with Petri nets: a transition (box) is enabled if every input place (circle) holds a token (black dot). If enabled, the transition can fire, removing tokens from its input places and adding tokens to its output places.

This paper presents an online technique to detect concept drift by sequential monitoring of the logs of a system. It is a multi-stage technique that uses the theory of *abstract interpretation* [10] to learn an internal representation (in terms of a *polyhedron* or an *octagon*), that is afterwards used to estimate the faithfulness of the representation in including the traces in the log. For the estimation and concept drift detection, we use an adaptive window technique [1] which has been proved to be very effective for similar purposes [2–4]. Remarkably, the techniques presented in this paper are automatic by nature, e.g., no user-intervention is required.

To our knowledge, the only work in the literature that addresses concept drift in process mining is [5], where *statistical hypothesis testing* is applied to detect *post-mortem* the drifts present in a log. Our approach, by contrast, is intended detect and react to changes in an online, almost real-time way. The technique may be used in different scenarios, such as: (i) for an *a posteriori* analysis, as in [5]; ii) as a preprocessor in an online setting, to segment the log and apply process discovery techniques separately to drift-free segments; (iii) to monitor a system in order to detect deviations from the expected behavior (embodied in an existing model).

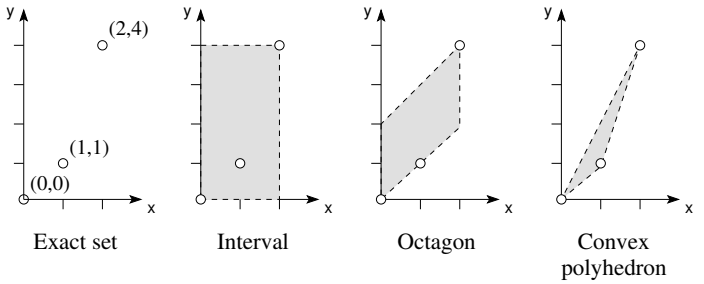
2 Background

Given a set of activities T, an event log over T is a multiset $L : T^* \to I\!N$. A sequence $\sigma \in T^*$ is a called *trace*. A trace σ is contained in a log if $L(\sigma) \geq 1$. Given a trace $\sigma = t_1, t_2, \ldots, t_n$, and a natural number $1 \leq k \leq n$, the sequence $t_1, t_2, \ldots, t_k$ is called the *prefix* of length k in σ. Given a log L, we denote by *Pref*(L) the set of all prefixes of traces in L. Finally, $\#(\sigma, e)$ is the number of times that activity e occurs in sequence σ.

2.1 Abstract Interpretation

Intuitively, abstract interpretation defines a procedure to compute an upper approximation for a given behavior of a system that still suffices for reasoning about the behavior itself. An important decision is the choice of the kind of upper approximation to be used, which is called the *abstract domain*. For a given problem, there are typically several abstract domains available. Each abstract domain provides a different trade-off between precision (closeness to the exact result) and computational efficiency.

There are many problems where abstract interpretation can be applied, several of them oriented towards the compile-time detection of run-time errors in software. For example, some analysis based on abstract interpretation can discover numeric invariants among the variables of a program. Several abstract domains can be used to describe the invariants: intervals [9], octagons [14], convex polyhedra [11], among others. These abstract domains provide different ways to approximate sets of values of numeric variables. For example, Figure 2 shows how these abstract domains can represent the set of values of a pair of variables x and y. For space reasons, we focus on the abstract domain of convex polyhedra. In the experiments, the domain of octagons is also used.

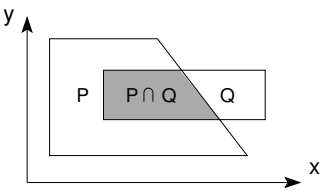


Interval	Octagon	Convex polyhedron
$0 \leq x \leq 2$	$0 \leq x \leq 2$	$y - x \geq 0$
$0 \leq y \leq 4$	$0 \leq y - x \leq 2$	$x - 2y \geq 0$
		$3x - y \leq 2$

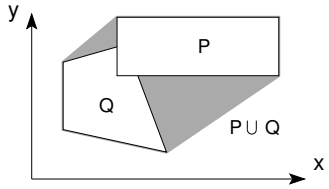
Fig. 2. Approximating a set of values (left) with several abstract domains

Convex Polyhedra. This domain can be described as the sets of solutions of a set of *linear inequality constraints* with rational ($\mathbb{Q}$) coefficients. Let P be a polyhedron over $\mathbb{Q}^n$; then it can be represented as the solution to some system of m inequalities $P = \{X | AX \leq B\}$ where $A \in \mathbb{Q}^{m \times n}$ and $B \in \mathbb{Q}^m$.

The domain of convex polyhedra provides the operations required in abstract interpretation. In this paper, we will mainly use the following two operations:



Meet ($\cap$): Given convex polyhedra P and Q, their meet is the intersection $P \cap Q$. Notice that this operation is exact, e.g., the meet or intersection of two convex polyhedra is always a convex polyhedron.



Join ($\cup$): Given convex polyhedra P and Q, we would like to compute the union of P and Q. Unfortunately the union of convex polyhedra is not necessarily a convex polyhedron. Therefore, the union of two convex polyhedra is approximated by the *convex hull*, the smallest convex polyhedron that includes both operands, denoted by $P \cup Q$. The example on the left shows $P \cup Q$ in gray.

2.2 Estimation, Drift, and Change Detection

In a stream setting one receives a potentially infinite stream of objects X_1, X_2, $\ldots X_t \ldots$ to be analyzed, where object X_t becomes available only at time step t. The underlying assumption is that there is some distribution D on the set of all objects generating the X_i's, often together with the assumption that some

degree of independence among the draws exist. The *concept drift* problem occurs when the distribution D cannot be assumed to be stationary, but there is in fact a distribution D_t for every t, which change gradually or abruptly over time.

There are two common strategies for dealing with concept drift: in the *sliding window* approach, one keeps a window of the last W elements, and the *change detection* approach, where one *estimates* or monitors some statistics of the X_i's and when some large enough deviation from past behavior is detected, change is declared. These two strategies can, of course, be combined too.

In this paper we will use for estimation and change detection the ADWIN (ADaptive WINdowing) method proposed in [1]. This choice is not the essential to the paper, and other methods (such as CUSUM or Page-Hinkley). Roughly speaking, ADWIN reads a real number X_t at each time step t, outputs an estimation of the current average of $E[X_t]$ in D_t, and also outputs a bit indicating whether drift has been detected in the recent observations. Internally, it keeps a window of the most recent X_t's whose length varies adaptively. It requires no parameters such as window length, delivering the user from the difficult tradeoff in such choices, and is efficient in the sense that it simulates a window of length W using $O(\log W)$ memory (rather than the obvious $O(W)$) and amortized $O(1)$ time per item. Furthermore, unlike most methods which are heuristics, ADWIN has rigorous guarantees (theorems) on its change detection performance. See [1] and the extended version of this paper for details[2].

3 Concept Drift Detection via Abstract Interpretation

This section describes the technique for concept drift detection in process mining. It first learns a process model using abstract interpretation (Section 3.1), which will be the main actor for the concept drift detection method in Section 3.2.

3.1 Learning via Abstract Interpretation

We now introduce the element to link traces from a log and abstract interpretation, which was initially presented in [7]:

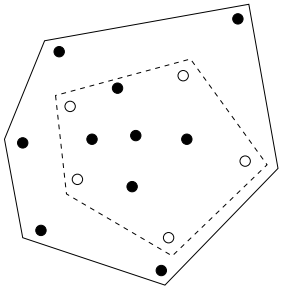
Definition 1 (Parikh vector). *Given a trace* $\sigma \in \{t_1, t_2, \ldots, t_n\}^*$*, the* Parikh vector *of* σ *is defined as* $\widehat{\sigma} = (\#(\sigma, t_1), \#(\sigma, t_2), \ldots, \#(\sigma, t_n))$.

Any component of a Parikh vector can be seen as a constraint for the n-dimensional point that it defines. Hence, the Parikh vector defined by $\widehat{\sigma} = (\#(\sigma, t_1), \#(\sigma, t_2), \ldots, \#(\sigma, t_n))$, a point, can be seen as the polyhedron $P_{\widehat{\sigma}} = \bigcap_{i=1}^{n}(x_i = \#(\sigma, t_i))$, where each variable x_i denotes the number of occurrences of activity t_i in σ, i.e., $x_i = \#(\sigma, t_i)$[3]. For each prefix σ of a trace in L, a polyhedron $P_{\widehat{\sigma}}$ can be obtained. Given all possible prefixes $\sigma_1, \sigma_2, \ldots, \sigma_m$ of traces

[2] Extended version of the paper:
`http://www.lsi.upc.edu/~jcarmona/ida2012ext.pdf`

[3] Hence a point $\widehat{\sigma}$ is represented as the polyhedron $P_{\widehat{\sigma}}$ that defines it.

in L, the polyhedra $P_{\widehat{\sigma_1}}, P_{\widehat{\sigma_2}}, \ldots, P_{\widehat{\sigma_k}}$ can be found[4]. Finally, the polyhedron $P = \bigcup_{i \in \{1 \ldots m\}} P_{\widehat{\sigma_i}}$ can be learned as the *convex-hull* of the points represented by the polyhedra $P_{\widehat{\sigma_1}}, P_{\widehat{\sigma_2}}, \ldots, P_{\widehat{\sigma_m}}$, and taken as the representation of the log.



Computing the convex hull of a large set of points may be expensive, so subsampling has been suggested [7] as a way to speed-up the process. The figure on the right shows an example: when the polyhedra of all the points are united, it may derive a polyhedron that covers large areas void of real points. If instead, a random sample of five points is used (denoted by points with white background), a smaller polyhedron is derived which, although not including the complete set of Parikh vectors from the log, represents a significant part of it and therefore the percentage of areas void of real points may be reduced. The *mass* of the polyhedron is the probability that a Parikh vector from the log belongs to the polyhedron; obviously, mass close to 1 is desired.

3.2 Online Algorithm for Concept Drift Detection

The technique is described as Algorithm 1. The input of the algorithm is the sequence of points or Parikh vectors produced from the traces received. The algorithm is divided into three stages: Learn (lines 2-8), Mean estimation (lines 9-15) and Mean monitoring (lines 16-25). Notice that the algorithm iterates over these three stages each time a drift is detected (Line 23).

In the learning stage, a set of m points is collected for training. The larger the m, the less biased the sampling is to a particular behavior, since the distribution of points will be more diverse. The outcome of the learning stage is a polyhedron $\widehat{P}$ that represents the concept underlying the set of points from the input.

In the mean estimation stage, the mass (fraction of points) belonging to $\widehat{P}$ is estimated using an ADWIN instance W. This process is iterated until a convergence criteria is met, e.g., the mean is stabilized to a given value. Notice that in this stage the algorithm, as written, may not converge to a stationary value if change keeps occurring. One could prevent this problem by using e.g. Chernoff bounds to bound on the number of iterations to reach a given approximation, assuming stationariness. In the monitoring stage, the same estimation is continued, but now with the possibility to detect a drift by detecting a change in the mass of $\widehat{P}$. In the simplest version, when a drift is detected, the three-stage technique is re-started from scratch; better strategies will be discussed in Section 5.

As explained in Section 2.2, an ADWIN instance W can be used to monitor a data sequence in order to determine drifts. In our setting, the data sequence will be produced by the outcome of the following question performed on each point from the traces in the log: does the learned polyhedron include the point? If it does, we will update W with a 1, and otherwise with a 0.

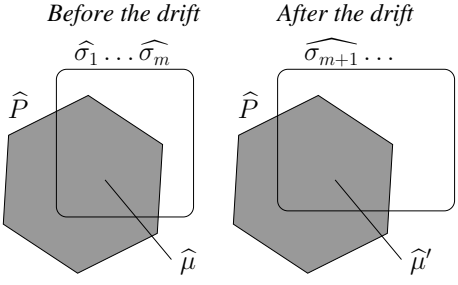
[4] Here k is in practice significantly smaller than $\sum_{\sigma \in L} |\sigma|$ since many prefixes of different traces in L share the same Parikh vector.

Algorithm 1. Concept Drift Detection algorithm

Input: Sequence of Parikh vectors $\widehat{\sigma_1}, \widehat{\sigma_2}, \ldots$

1 **begin**
2 Select appropriate training size n;
3 Collect a random sample of m vectors out of the first n;
// $m < n$ `if required for efficiency`
// `Stage 1: Learning the current concept` $\widehat{P}$
4 $\widehat{P}$ = "empty domain" ;
5 **for** $j \leftarrow 1$ **to** m **do**
6 let $\widehat{\sigma_j}$ be the jth randomly selected vector; compute $P_{\widehat{\sigma_j}}$;
7 $\widehat{P} = \widehat{P} \cup P_{\widehat{\sigma_j}}$;
8 **end**
// `Stage 2: Estimating the average of points included in` $\widehat{P}$
9 W =InitADWIN;
10 $i = m + 1$;
11 **repeat**
12 **if** $\widehat{\sigma_i} \in \widehat{P}$ **then** $W = W \cup \{1\}$;
13 **else** $W = W \cup \{0\}$;
14 $i = i + 1$;
15 **until** *convergence criteria on* W *estimation*;
// `Stage 3: Monitoring the average of points included in` $\widehat{P}$
16 **while** *true* **do**
17 **if** $\widehat{\sigma_i} \in \widehat{P}$ **then** $W = W \cup \{1\}$;
18 **else** $W = W \cup \{0\}$;
19 $i = i + 1$;
20 **if** *Drift detected on* W **then**
21 Declare "drift!";
22 Throw away the current set of points;
23 Jump to line 2;
24 **end**
25 **end**
26 **end**

This way, W will estimate the mass of current polyhedron, denoted $\widehat{\mu}$. The figure next illustrates the approach: a polyhedron $\widehat{P}$ which (maybe partially) represents the Parikh vectors of the traces in the log is learned, and we estimate $\widehat{\mu}$, the fraction of the points falling into the polygon. When a change occurs (in the figure, more points are added), chances are that this quantity changes to a new value $\widehat{\mu'}$ (this assumption will be discussed in Section 5).



3.3 Rigorous Guarantees

Using the rigorous guarantees of ADWIN given in [1], one can give a rigorous statement on the ability of the method above to detect drift in its input. The proof can be found in the extended version.

Theorem 1. *Suppose that for a sufficiently long time $T_0 \dots T_1$ the input distribution has remained stable, and that the learn phase has built a polyhedron $\widehat{P}$, with mass μ_1. Suppose that by time T_2 ($> T_1$) the input distribution has changed so that the mass of $\widehat{P}$ is μ_2 from then on. Then by time at most $T_2 + O(\ln(T_2 - T_0)/(\mu_2 - \mu_1)^2)$ the method will have detected change and restarted.*

Note that the case of abrupt change is when $T_2 = T_1 + 1$, and that larger changes in the mass of the current polyhedron imply shorter reaction times.

4 Experiments on Concept Drift Detection

To test the detection technique of Section 3, a set of models (in our case, Petri nets) have been used. For each model M, a log has been created by simulating M. The first six models in Table 1 are taken from [8], and represent typical behaviors in a concurrent system. Model CYCLES(X,Y) represents a workflow of overlayed cyclic processes. Finally, the models A12F0N00 ... T32F0N00 are originated from well-known benchmarks in the area of process mining.

To derive logs that contain a drift, each model has been slightly modified in four different dimensions:

- *Flip*: the ordering of two events of the model has been reversed.
- *Rem*: one event of the model has been removed
- *Conc*: two sequential events have been put in parallel
- *Conf*: two sequential or concurrent events have been put in conflict

These transformations represent a wide spectra of drifts for control-flow models. Each transformation leads to a new model, which can be simulated to derive the corresponding log[5]. An assumption of this work, which is inherited from the use of the techniques in [7], is that a drift causes a modification in the spectra of Parikh vectors representing a model. Although this assumption is very likely in most practical cases, it does not cover the drifts which neither incorporate nor remove Parikh vectors to the current spectra. However, using an alternative technique to [7] in Algorithm 1 which is sensitive to this type of drifts can be the cure for this particular problem. Notice that only a single transformation is applied to a model to introduce drift. In general, however, drift may arise from

[5] All the models and logs used in this paper can be obtained at the following url: `http://www.lsi.upc.edu/~jcarmona/benchsIDA2012.tar`

Table 1. Concept drift detection: number of points to detect the drift

benchmark	$\|\Sigma\|$	$\|Places\|$	$\|L1\|$	*Flip*	*Rem*	*Conc*	*Conf*
SharedRes(6)	24	25	4000	115	54	183	37
SharedRes(8)	32	33	4000	165	73	381	83
ProdCons(8)	41	42	4000	337	550	262	266
ProdCons(9)	46	47	4000	256	136	323	489
WeightMG(9)	9	16	4000	101	16	75	16
WeightMG(10)	10	18	4000	147	28	53	18
Cycles(4,2)	14	11	4000	563	23	664	22
Cycles(5,2)	20	16	4000	554	22	845	21
a12f0n00	12	11	620	83	76	117	15
a22f0n00	22	19	2132	340	56	99	198
a32f0n00	32	32	2483	67	79	258	162
a42f0n00	42	46	3308	178	41	185	37
t32f0n00	33	31	3766	143	28	394	36

several transformations, either simultaneous or spaced out in time. However, if the technique is able to identify drift from a single transformation, it should also be able to detect these more drastic drifts.

The experiment performed is to use the concatenation of two logs $L1$, $L2$ with different distribution, so that abrupt change occurs at the transition from $L1$ to $L2$. Using the Apron library [13], an octagon/polyhedra P is obtained from $L1$ by sampling a few points, and an ADWIN is then created to estimate the fraction of points in the input (still from $L1$) that are covered by P. When the estimation converges, the input is switched to points from $L2$. If the drift is not sporadic, the fraction of points covered by P will change, ADWIN will detect change in this quantity, and drift will be declared.

In Table 1 the results of the experiment are provided. The second column in the table reports the number of different events (which also represents the number of transitions in the Petri net), and the third column reports the number of places. Column $|L1|$ provides the number of points used in the stages 1 and 2 of the algorithm (we have set a maximum of four thousand points in the simulation)[6]. The following columns denote the logs corresponding to the models with each one of the aforementioned transformations. In each cell from column five on we provide the number of points needed to sample in order to detect a drift. For instance, for the SharedRes(8) benchmark, it only needs to sample 73 points in order to detect a drift when a single event is removed, and needs to sample 381 points to detect that two events became concurrent. In terms of CPU time, all drifts have been detected in few seconds[7].

[6] $|L2|$ is particular to each drift. In general it is of the same magnitude as $|L1|$.

[7] In the experiments the domain of octagons has been used when $|\Sigma| > 20$, to bound the time and memory requirements in the learning stage.

A second experiment was performed on a log with another kind of drift. The detailed description can be found in [6]. It is a real log containing the processing of copy/scan jobs on a digital copier. The log contains 1050 traces and 35 event classes, with a drift introduced after 750 traces, consisting in the addition of a new functionality to the copier (zooming function for image processing). For the first part of the log (traces 1 – 750), there are 34.427 points, whereas for the second part (traces 751 – 1050), 13.635 points. The technique of this paper identifies the drift by sampling only 504 points.

5 Change Location and Unraveling Process Evolution

So far, we have focused into explaining how to apply the theory of abstract interpretation together with estimation and change detection in order to detect concept drifts in the area of process mining. In this section we will briefly address the two other problems highlighted in [5]: change location and characterization, and unravel process evolution; this is ongoing work.

Change Location and Characterization. A polyhedron P is the solution to the system of m inequalities $P = \{X|AX \leq B\}$ where $A \in \mathbb{Q}^{m \times n}$ and $B \in \mathbb{Q}^m$ (see Section 2.1). A subset C of these inequalities called *causal constraints* can be used to derive the corresponding process model (see [7] for details). A causal constraint satisfies particular conditions that makes it possible to be converted into a process model element, e.g., a *place* and its corresponding arcs in a Petri net. Since the adaptive windowing technique described in Section 2.2 requires few resources, one can use one adaptive window *for monitoring each causal constraint in* C. Thus, after the learning stage of Algorithm 1, $|C|$ instances of ADWIN are used to estimate the average satisfaction for each constraint. Finally, in stage three these ADWIN's can detect drift at each of the constraints. When global or partial drift occurs, its location is exactly characterized by the causal constraints that have experienced drift, which can be mapped to the corresponding places in the process model. Remarkably, this provides a *fine-grain* concept drift detection version of the technique presented in Section 3.

Unraveling Process Evolution. After change has been localized and characterized as above, the new process model can be then produced. This is crucial to unravel the process evolution [5]. Two sets of causal constraints will be used to derive the new process model: i) the causal constraints which still are valid after the drift, and ii) the new set of causal constraints that may appear in the new polyhedron learned by revisiting stage one of Algorithm 1. For the former set, both drifting and non-drifting causal constraints detected in the previous iteration of Algorithm 1 will be considered. For drifting causal constraints, a threshold value may be defined to determine when drift is strong enough to

invalidate it. As for the complexity of the model revision, for example the method in [7] for deriving Petri nets from polyhedra is well-behaved in the sense that a change in some of the inequalities can be translated to a local change in the Petri net, with proportional computational cost.

The same idea can be used to alleviate a problem with the change detection strategy described in Section 3. Recall that there we were only detecting changes where new points appeared in regions outside the learned polyhedron. Drift may also mean that points previously in the log, or in its convex hull, do no longer appear (e.g., if some behaviors disappear and the polygon becomes larger than necessary). We can detect many such changes by monitoring many aspects of the stream of points, instead of just the mass of the learned polyhedron. For example, we could use an array of ADWIN instances to monitor the average distance among points, distance to their centroid or set of designated points, distance to each constraint, projection to a set of random hyperplanes, etc.

6 Conclusions and Future Work

Concept drift is an important concern for any data analysis scenario involving temporally ordered data. Surprisingly, there is very little work (in fact, possibly only [5]) in dealing with concept drift within process mining techniques.

In this paper we have presented the first online mechanism for detecting and managing concept drift, combined with the process mining approach in [7] based on abstract interpretation and Petri net models. Our experiments on process mining benchmark data twisted to incorporate drift show that our method detects abrupt changes quickly and accurately. We have also described how to apply the mechanism for a richer set of tasks: characterizing and locating change, unraveling the process change, and revising the mined models.

Future work includes experimenting with different forms of change, particularly, gradual, long term changes and those discussed at the end of Section 5; implementing the fine-grain detection mechanisms for change location and unraveling; and using our approach in a real scenario with a high volume of data, so sampling becomes essential, and with strong requirements on time and memory. Also, investigating tailored techniques that can deal with logs that contain *noise* is an interesting future research direction. A possibility will be to adapt Algorithm 1 to use some of the few process discovery techniques that can handle noise in the log [12, 19, 20].

Acknowledgements. We would like to thank R. P. Jagadeesh Chandra Bose and W. M. P. van der Aalst for the providing some of the logs of the experiments. This work is partially supported by BASMATI MICINN project (TIN2011-27479-C04-03), by the SGR2009-1428 (LARCA), by the EU PASCAL2 Network of Excellence (FP7-ICT-216886), by the FORMALISM project (TIN2007-66523) and by the project TIN2011-22484.

References

1. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SDM. SIAM (2007)
2. Bifet, A., Gavaldà, R.: Adaptive Learning from Evolving Data Streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009)
3. Bifet, A., Gavaldà, R.: Mining frequent closed trees in evolving data streams. Intell. Data Anal. 15(1), 29–48 (2011)
4. Bifet, A., Holmes, G., Pfahringer, B., Gavaldà, R.: Mining frequent closed graphs on evolving data streams. In: Apté, C., Ghosh, J., Smyth, P. (eds.) KDD, pp. 591–599. ACM (2011)
5. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P., Žliobaitė, I.e., Pechenizkiy, M.: Handling Concept Drift in Process Mining. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 391–405. Springer, Heidelberg (2011)
6. Jagadeesh Chandra Bose, R.P.: Process Mining in the Large: Preprocessing, Discovery, and Diagnostics. PhD thesis, Eindhoven University of Technology (2012)
7. Carmona, J., Cortadella, J.: Process Mining Meets Abstract Interpretation. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 184–199. Springer, Heidelberg (2010)
8. Carmona, J., Cortadella, J., Kishinevsky, M.: New region-based algorithms for deriving bounded Petri nets. IEEE Trans. on Computers 59(3), 371–384 (2009)
9. Cousot, P., Cousot, R.: Static determination of dynamic properties of programs. In: 2nd Int. Symposium on Programming, Paris, France, pp. 106–130 (1976)
10. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proc. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, pp. 238–252. ACM Press (1977)
11. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: Proc. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, pp. 84–97. ACM Press, New York (1978)
12. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
13. Jeannet, B., Miné, A.: Apron: A Library of Numerical Abstract Domains for Static Analysis. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 661–667. Springer, Heidelberg (2009)
14. Miné, A.: The octagon abstract domain. In: IEEE Analysis, Slicing and Tranformation, pp. 310–319. IEEE CS Press (October 2001)
15. Murata, T.: Petri nets: Properties, analysis and applications. Proc. of the IEEE 77(4) (1989)
16. van der Aalst, W., et al.: Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)

17. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
18. van der Aalst, W.M.P., Günther, C.W.: Finding structure in unstructured processes: The case for process mining. In: Basten, T., Juhás, G., Shukla, S.K. (eds.) ACSD, pp. 3–12. IEEE Computer Society (2007)
19. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Genetic Process Mining. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 48–69. Springer, Heidelberg (2005)
20. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: CIDM, pp. 310–317 (2011)

An Evolutionary Based Clustering Algorithm Applied to Dada Compression for Industrial Systems

Jun Chen[1], Mahdi Mahfouf [2], Chris Bingham[1], Yu Zhang[1], Zhijing Yang[1], and Michael Gallimore[1]

[1] School of Engineering, University of Lincoln, Lincoln, U.K.
[2] Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K.
juchen@lincoln.ac.uk

Abstract. In this paper, in order to address the well-known 'sensitivity' problems associated with *K*-means clustering, a real-coded Genetic Algorithms (GA) is incorporated into *K*-means clustering. The result of the hybridisation is an enhanced search algorithm obtained by incorporating the local search capability rendered by the hill-climbing optimisation with the global search ability provided by GAs. The proposed algorithm has been compared with other clustering algorithms under the same category using an artificial data set and a benchmark problem. Results show, in all cases, that the proposed algorithm outperforms its counterparts in terms of global search capability. Moreover, the scalability of the proposed algorithm to high-dimensional problems featuring a large number of data points has been validated using an application to compress field data sets from sub-15MW industry gas turbines, during commissioning. Such compressed field data is expected to result in more efficient and more accurate sensor fault detection.

Keywords: hybridised clustering algorithm, genetic algorithms, *K*-means algorithms, data compression, sensor fault detection.

1 Introduction

Knowledge in some sense can be defined as the ability that distinguishes the similar from the dissimilar. However, given the amount of information (in other words, data) encountered in the real life, such ability is sometimes limited, which in turn gives rise to limited knowledge. Without further abstraction, more information may simply lead to more difficulties for human beings or even for computational algorithms to uncover the underlying structure. The key to the success of handling ample data lies in the capability of retrieving representatives or prototypes from anfractuous information via a certain level of abstraction. Representing the data by fewer prototypes necessarily loses certain fine details, but achieves simplification and interpretability. One of the vital means which allows the abstraction of this kind is data clustering, which can be employed to compress a large data set into a few representative points that can be more easily handled in practice.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 103–113, 2012.
© Springer-Verlag Berlin Heidelberg 2012

In recent decades significant effort has been directed towards hybridizing GAs with conventional partition-based clustering algorithms (J. C. Bezdek, *et al.*, 1994; L. O. Hall, *et al.*, 1999; K. Krishna, et al., 1999; S. Bandyopadhyay, *et al.*, 2002; W. Sheng, *et al.*, 2006). The rationale behind hybridization is based on the fact that most optimization techniques used in partition-based clustering are inherently hill-climbing techniques which are very sensitive to the initial settings and may lead to convergence to local optima. One of the earliest GA-based clustering implementations was made by J. C. Bezdek, *et al.* (1994) by hybridizing a GA with Fuzzy *C*-Means (FCM), in which a binary coded chromosome was adopted to represent cluster centres so that a GA can be used to iteratively search for the optimal fuzzy partitions. L. O. Hall, *et al.* (1999) improved the efficiency of such a GA-based fuzzy clustering algorithm by coding the centres with a binary Gray code representation in which any two adjacent numbers are different by one bit. The authors of both works argued that coding centres in the chromosome is more efficient than coding the membership matrix. Similar efforts have been made to hybridize GAs with K-means clustering. Instead of coding centres in the chromosome, C. A. Murthy, *et al.* (1996) proposed to code the cluster identity number which is assigned to each data point. Hence, the length of the chromosome is the same as the number of data points, which makes the algorithm vulnerable to the large data set. K. Krishna, *et al.* (1999) proposed to code the hard membership matrix in a binary form so that a GA framework can be applied to find the optimal hard membership matrix which minimises the 'within-cluster variance'. S. Bandyopahyay, *et al.* (2002) acknowledged the comments made earlier by J. C. Bezdek (1994) and noticed that the clustering problem under a GA framework is actually a real-valued optimization problem. Hence, a real-valued GA was adopted in their work. Cluster centres are encoded in the chromosome with floating-point values. Recently, W. Sheng, *et al.* (2006) incorporated GAs into K-medoids clustering using an integer encoding scheme.

Despite of the great achievements reported in the aforementioned works, several related problems deserve more attention:

1. In most of the previous implementations, binary-coded GAs were widely adopted. However, applying binary-coded GAs to such a continuous optimisation problem will result in a so-called 'Hamming cliffs' difficulty associated with certain strings (K. Deb, 2001, p. 110).
2. Binary-coded GA-based clustering suffers from the problems of imprecision and the 'curse of dimensionality'. In both cases, a longer chromosome is needed which increases the search space. It also means that a large population size is required in order to have an effective search.
3. Although S. Bandyopahyay, *et al.* (2002) adopted a real-valued GA, a so-called Naïve crossover (single-point crossover) was used in their work, which as mentioned by K. Deb (2001, p. 112) does not have an adequate search power for experimental cases. Hence, it relinquishes it search responsibility to the mutation operators, which is less effective.

In light of the above problems, in this paper, a novel clustering algorithm, termed G3Kmeans (J. Chen, 2009), is proposed, which chooses a real-coded GA, namely G3PCX (Deb *et al.*, 2002), as its optimisation method. G3PCX is then incorporated into *K*-means clustering, which only encodes cluster centres in the chromosome. Hence, the length of the chromosome rests only with the number of the cluster centres. The search power of G3Kmeans is significantly enhanced by combining the Parent-Centric Recombination (PCX) operator with the hill-climbing operator. Such a combination takes full advantage of global search capability mainly attributed to the PCX recombination operator and local search ability rendered by the hill-climbing operator. As a result, G3Kmeans is more robust to the initialisation compared to other conventional partition-based clustering and is more efficient than other algorithms of the same class.

The paper is organised as follows: Section 2 details the implementation of G3Kmeans; in order to show the superiority of G3Kmeans, in Section 3, one synthetic data set and iris data set are utilised to validate the proposed algorithm; the results are then compared to those of FCM, *K*-means, Subtractive Clustering (Chiu, 1994), GA-clustering (C. A. Murthy, *et al.*, 1996) and KGA (S. Bandyopadhyay, *et al.*, 2002); in Section 4, the algorithm is applied to compress field data sets from sub-15MW industry gas turbines, during commissioning, which is envisaged to result in more efficient and more accurate sensor fault detection; finally, Section 5 draws conclusions and gives future research direction.

2 Description of G3Kmeans

The detailed steps involved in G3Kmeans are described as follows:

- Step1: **Initialisation:** the randomly generated '*k*' cluster centres are encoded in each chromosome in a concatenated form. *P* initial chromosomes are generated in the initial population.
- Step 2: **Assigning data points:** Each data point is assigned to one cluster with the centre of C_i using Eq. (1):

$$X_m \in C_i: if\{_{m = 1,2,\dots,N; i,l = 1,2,\dots,k; l \neq i}^{\|X_m - C_i\| < \|X_m - C_l\|} \tag{1}$$

  where, $\|\ \|$ is the Euclidean norm, X_m is the *m*th data point, C_i is the *i*th cluster centre, k is the pre-specified number of cluster centres, and *N* is the number of data samples. After the assignment, cluster centres encoded in the chromosome are updated by calculating the mean value of each cluster.

- Step 3: **Fitness computation:** the fitness of each individual is calculated using Eq. (2):

$$\varpi(C_1, C_2, \dots, C_k) = \sum_{l=1}^{k} \sum_{X_m \in C_l} \|X_m - C_l\|^2 \quad l = 1, \dots, k \tag{2}$$

  where, ϖ is a within-cluster-distance metric to be optimised (minimised) and $C_1, C_2, \dots, C_3$ are k cluster centres.

- Step 4: **Parent-Centric Crossover (PCX):** Generate λ offspring from the μ parents using the PCX recombination as described in Eq. (3):

$$X_{offspring} = X^{(P)} + \omega_\zeta \cdot \overrightarrow{d^{(P)}} + \sum_{i=1, i\neq p}^{\mu} \omega_\eta \cdot \overline{D} \cdot \vec{e}^{(i)} \quad (3)$$

  PCX operator first calculates the mean vector $\vec{g}$ of the chosen μ parents so that for each offspring, one parent $X^{(P)}$ is chosen with equal probability. The direction vector $\overrightarrow{d^{(P)}} = X^{(P)} - \vec{g}$ is then calculated. Afterwards, from each of the other $(\mu - 1)$ parents, perpendicular distances D_i to the vector $\overrightarrow{d^{(P)}}$ are computed and their average $\overline{D}$ is found. $\vec{e}^{(i)}$ are the $(\mu - 1)$ orthonormal bases that span the subspace perpendicular to the vector $\overrightarrow{d^{(P)}}$. The parameters ω_ζ and ω_η are zero-mean normally distributed variables with variance σ_ζ^2 and σ_η^2 respectively. Detailed description of PCX is referred to K. Deb, *et al.* (2002).
- Step 5: **Fitness computation:** the cluster centres and fitness values of the offspring are updated and calculated again as in the Step 2 and 3 accordingly.
- Step 6: **Parents to be replaced:** choose two parents at random from the population *P*.
- Step 7: **Replacement:** from the combined subpopulation of two chosen parents and λ created offspring, choose the best two solutions and replace the chosen two parents (in Step 6) with these solutions.
- Step 8: **Iteration:** the aforementioned steps from Step 2 are repeated for a specified generations or until the standard deviation of the fitness values of the last five iterations becomes less than a threshold *stable*, and the final solution is the one with the smallest fitness value at the end of the execution.

It is worth mentioning that in the following experiments all the user-specified parameters are set as those suggested by Deb, *et al.* (2002) unless otherwise stated. Hence, $\sigma_\zeta^2 = \sigma_\eta^2 = 0.1,\ P = 100,\ \lambda = 2, \mu = 3, stable = 0.001$.

3 Experimental Studies

3.1 An Artificial Data Set

The first test problem consists of 4 randomly generated Gaussian clusters around the nominal cluster centre[1]. Each cluster contains 100 data points. In order to test the robustness of the proposed algorithm to the data contaminated by random noise, these 100 data points are then mixed with 200 randomly distributed noisy data. Figure 1 shows an example data set.

In order to obtain a quantitative comparison with different clustering algorithms, objective values are calculated using Eq. (2) and are served as the measure to distinguish the algorithms' efficacy. The objective value of the original clusters is also computed using the nominal centres as the baseline to see if a specific clustering algorithm can approach the nominal centres as close as possible.

[1] Termed nominal centres since the objective value of the clusters generated around these centres may not represent the minimum objective value compared to those of the identified clusters (see Table 1 for more details).

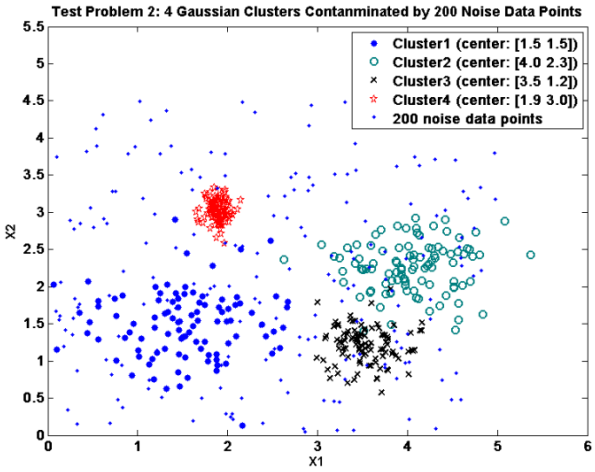


Fig. 1. 4 Gaussian clusters contaminated by 200 noise data points

The difficulties of this test problem lie in the facts that the classes are not well separated and the search space presents many local optima due to the presence of noise. Table 1 summarises the results of FCM, Subtractive clustering, K-means and G3Kmeans algorithms respectively. The results are the average values of 20 independent runs. Standard deviation of the results is also calculated to show if the algorithm is robust to different initialisation and runs.

Table 1. Comparisons Of The Objective Values Between Different Clustering Algorithms*

Methods	Max.	Min.	Mean	Standard Deviation	Time (second)
Original Clusters**	13.8697	13.8697	13.8697	0	-
FCM	13.1898	13.1895	13.1897	1.038e-004	0.0269
K-means	18.3056	13.0382	13.5173	1.5881	**0.0130**
Subtractive Clustering	16.2954	16.2954	16.2954	0	0.2000
G3Kmeans	**13.0382**	**13.0382**	**13.0382**	**0**	0.8008

* The objective values are calculated using the normalised data.
** The objective value of the original clusters is obtained using nominal centres.

For this problem, both FCM and *K*-means are sensitive to the initialisations. Due to the presence of noise, this problem contains several local optima, which corresponds to the non-zero standard deviations produced by these two algorithms. In fact, *K*-means algorithm misclassifies clusters 2 out of 20 runs. The larger the standard deviation, the more likely an algorithm depends on the initial condition. Figure 2 demonstrates the distribution of the identified cluster centres via different clustering algorithms and the nominal centres and the nominal centres. It can be seen from Table 1 and Figure 2 that the *K*-means algorithm not only depends on its initial condition but also gives higher objective values which implies that the centres found by *K*-means may be far from the nominal ones.

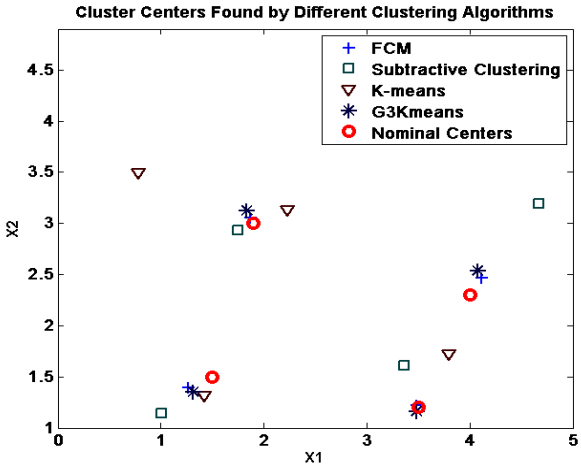


Fig. 2. The distribution of the identified cluster centres

Figure 3 shows the evolution curve of the G3Kmeans. G3Kmeans takes 11 generations to terminate. However, the algorithm has already converged to the minimum objective within 6 generations.

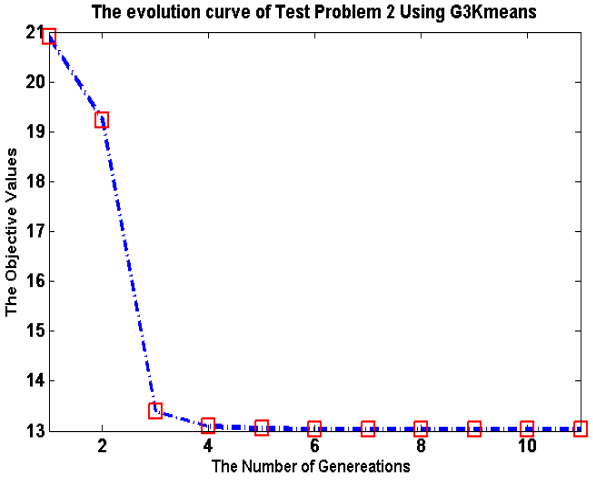


Fig. 3. The evolution curve of the problem using G3Kmeans

3.2 *Iris* Data

In order to compare with other GA-based *K*-means algorithms, e.g. GA-Clustering (C. A. Murthy, *et al.*, 1996) and KGA (S. Bandyopadhyay, *et al.*, 2002) and to justify the rationale of the proposed hybridisation, the Iris data set is employed (R. A. Fisher, 1936), which consists of 10 patterns belonging to three categories of Iris. Each of the

patterns is described by four real-valued features in centimetres, which are the sepal length, sepal width, petal length and width. Each of the categories consists of 50 patterns. The difficulties of the Iris data lie in the fact that the problem possesses two overlapped classes and presents many local optima. Figure 4 shows the evolution curve of G3Kmeans. It can be seen from tis graph that G3Kmeans takes 11 generations to terminate. However, the algorithm converged to the minimum objective value (6.9981) within 5 generations. It is noteworthy that the other GA-based *K*-means algorithms require longer computation times than the G3Kmeans.

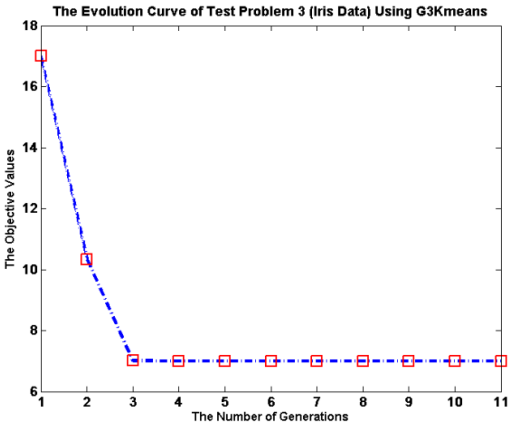


Fig. 4. The evolution curve of the Iris data using G3Kmeans

Table 2 summarises the results over 20 independent runs. The results of GA-clustering and KGA are extracted from S. Bandyopadhyay, *et al.* (2002). It is worth noting that the objective values included in Table 2 are calculated using the original data rather than the normalised one.

Table 2. Comparasions Of Different Clustering Algorithms On The Iris Data

Methods	Max.	Min.	Mean	Standard Deviation	Time (second)
FCM	79.4566	79.4516	79.4557	1.6000e-3	0.0252
K-means	142.9149	**79.0031**	95.0244	27.4598	**0.0052**
Subtractive Clustering	84.6800	84.6800	84.6800	0	0.0114
GA-clustering	139.7782	124.1274	135.4048	-	-
KGA	97.1008	97.1008	97.1008	0	-
G3Kmeans	**79.0031**	**79.0031**	**79.0031**	**0**	0.4623

For this problem, *K*-means algorithm misclassified clusters 4 out 20 runs, which corespond to its maximum objective value shown in Table 2. A large standard deviation associated with *K*-means algorithm indicates that the Iris data consists of many local optima and confirms that the *K*-means algorithm is vulnerable to such a

scenario. The results of G3Kmeans are far superior to those of GA-clustering and KGA for the reasons dicussed in Section 1. In fact, GA-clustering is unable to provide meaningful clusters within 1000 iterations (S. Bandyopadhyay, *et al.*, 2002). This is mainly due to the coding scheme adopted by GA-clustering, which needlessly increasses the search space and thus requires more computational effort to converge. Figure 5 shows the identified Iris classes and their centres via every two features.

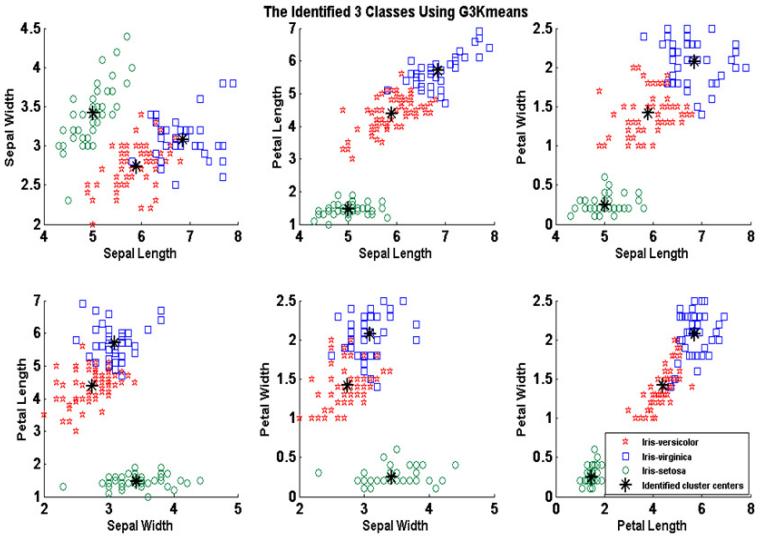


Fig. 5. The identified 3 Iris classes using G3Keans

4 Real-World Data from Sub-15MW Industry Gas Turbines

In order to test the scalability of the proposed G3Kmeans algorithm to high dimensional problems and its efficacy in the application of compressing large data sets, two real data sets from sub-15MW industry gas turbines are used as test problems.

4.1 Data Description

The data is formed via four groups of sensors, namely bearing vibration sensors, speed sensors, bearing temperature sensors, and lubrication oil temperature sensors, sited on a twin shaft industrial gas turbine, as shown in Figure 6. Hence, the data sets feature as high as 20 feature variables. 42098 and 49074 data samples are included respectively in each of them, one being the normal data and the other one representing the abnormal.

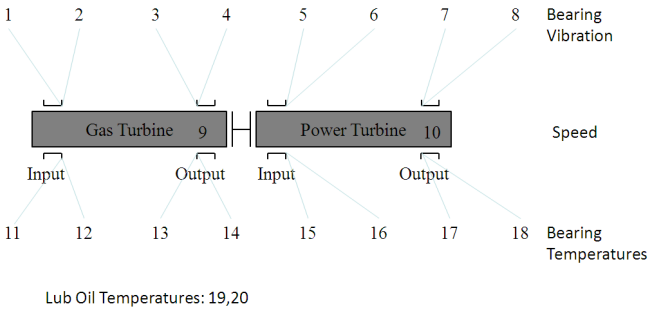


Fig. 6. The structure of 20 sensors on a twin shaft industrial gas turbine

4.2 G3Kmeans for Data Compression

In the previous studies, such as the one conducted by Zhang, *et al.* (2012), the Principal Component Analysis (PCA) Based Y-distance Indexes were utilised to distinguish the abnormal from the normal. However, it is highly restricted by the number of data points that the PCA can process. Large data sets normally require large storage space in order to form the PCA matrix. Given the fact that abundant monitoring data from different time periods is available, it is desirable to use all the information covered by a wide range of time rather than a fraction of them. Hence, G3Kmeans is applied, in this case, for the purpose of reducing such a large data set whilst still preserving 'sufficient' information contained in the original data. This has been achieved by utilising all clustered centres as the compressed data set to represent the original data set. It is envisaged that by combining several such compressed data sets registered during different periods of time, more information will be unveiled at the same time to the fault detection algorithm and more accurate and efficient fault detection could thus be achieved. Figure 7 and 8 give examples of the compressed data sets for the original two data sets.

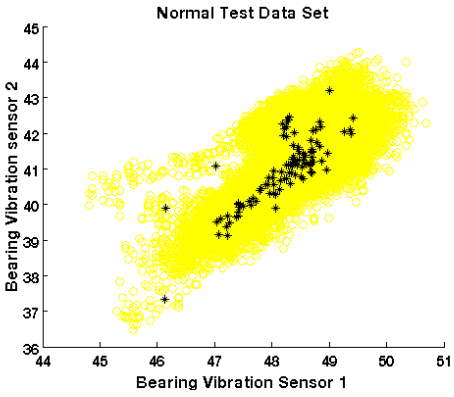


Fig. 7. Compressing 42098 normal test data into 100 data points via G3Kmeans (yellow circle: the original data set; black asterisk: the compressed data set)

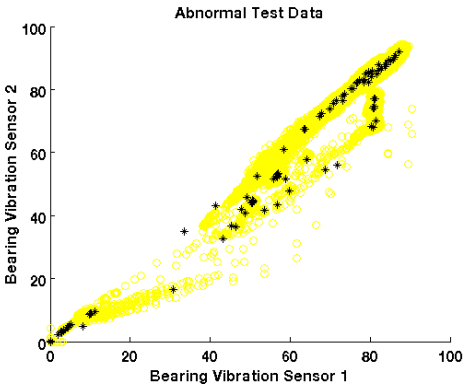


Fig. 8. Compressing 49047 abnormal test data into 100 data points via G3Kmeans (yellow circle: the original data set; black asterisk: the compressed data set)

It can be seen from Figure 7 that for the normal test data, the values of the sensors are very close to each other, which forms a narrow range regarded as the norm. Hence, the compressed data points are also close to each other, and are already good enough to capture all the information contained in the set. Figure 8 represents a different scenario where the values of the sensors are spread across wider ranges, with some outside the normal range. It is thus very crucial that a compression algorithm can capture not only the normal range but also the abnormal ranges. Visual inspection of Figure 8 confirms that all important features across different ranges have been carefully preserved by G3Kmeans. The compression ratios in both cases are 420.98:1 and 490.47:1, respectively.

5 Conclusions

In this chapter, an evolutionary based clustering algorithm, namely G3Kmeans, is introduced. The proposed algorithm is tested extensively through the artificial and real data sets. The results show that the proposed algorithm is superior to other more traditional clustering algorithms in that: 1) it is robust to different initial settings; 2) it can approach very closely to the global optimal partitions, especially for high-dimensional problems; 3) it is computationally more efficient compared to other evolutionary based clustering algorithms. Such superiority is mainly attributed to the combined local and global search operators adopted in G3Kmeans, which are specially designed for the real-valued optimisation. The encoding scheme of the proposed method also ensures a reasonable search space as opposed to GA-clustering algorithm. Due to its robustness to noisy and high dimensional scenarios, the algorithm is used for compressing large monitoring data. Results show that G3Kmeans can achieve high compression ratio whilst 'preserving' all important features contained in the original data set. As one future investigation, several compressed data sets will be joined together in order to provide more information to the fault detection algorithms at the same time in a bid to generating more accurate and more efficient fault detection.

References

1. Murthy, C.A., Chowdhury, N.: In Search of Optimal Clusters Using Genetic Algorithms. Pattern Recognition Letters 17, 825–832 (1996)
2. Bezdek, J.C., Hathaway, R.J.: Optimization of Fuzzy Clustering Criteria Using Genetic Algorithms. In: The 1st IEEE Conference on Evolutionary Computation, vol. 2(2), pp. 589–594 (1994)
3. Chen, J.: Biologically Inspired Optimisation Algorithms for Transparent Knowledge Extraction Allied to Engineering Materials Processing. PhD. Thesis, University of Sheffield (2009)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
5. Deb, K., Anand, A., Joshi, D.: A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. In: Evolutionary Computation, vol. 10 (4), pp. 371–395. MIT Press (2002)
6. Krishna, K., Narasimha Murty, M.: Genetic K-Means Algorithm. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 29(3), 433–439 (1999)
7. Hall, L.O., Özyurt, I.B., Bezdek, J.C.: Clustering with a Genetically Optimized Approach. IEEE Transactions on Evolutionary Computation 3(2), 103–112 (1999)
8. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Ann. Eugen. 7(2), 179–188 (1936)
9. Bandyopadhyay, S., Maulik, U.: An Evolutionary Technique Based on K-Means Algorithm for Optimal Clustering in $\mathbb{R}^N$. Information Sciences 146, 221–237 (2002)
10. Chiu, S.L.: Fuzzy Model Identification Based on Cluster Estimation. J. of Intelligent & Fuzzy Systems 2(3) (1994)
11. Sheng, W., Liu, X.: A Genetic K-medoids Clustering Algorithm. Journal of Heuristics 12, 447–466 (2006)
12. Zhang, Y., Bingham, C.M., Yang, Z., Gallimore, M., Ling, W.K.: Sensor Fault Detection for Industrial Gas Turbine System by Using Principal Component Analysis Based Y-distance Indexes. In: Proceeding on 8th IEEE, IET International Symposium on Communication Systems, Networks and Digital Signal Processing (2012)

Multi-label LeGo — Enhancing Multi-label Classifiers with Local Patterns

Wouter Duivesteijn[1], Eneldo Loza Mencía[2], Johannes Fürnkranz[2], and Arno Knobbe[1]

[1] LIACS, Leiden University, The Netherlands
{wouterd,knobbe}@liacs.nl
[2] Knowledge Engineering Group, TU Darmstadt, Germany
{eneldo,juffi}@ke.tu-darmstadt.de

Abstract. The straightforward approach to multi-label classification is based on decomposition, which essentially treats all labels independently and ignores interactions between labels. We propose to enhance multi-label classifiers with features constructed from local patterns representing explicitly such interdependencies. An Exceptional Model Mining instance is employed to find local patterns representing parts of the data where the conditional dependence relations between the labels are exceptional. We construct binary features from these patterns that can be interpreted as partial solutions to local complexities in the data. These features are then used as input for multi-label classifiers. We experimentally show that using such constructed features can improve the classification performance of decompositive multi-label learning techniques.

Keywords: Exceptional Model Mining, Multi-Label Classification, LeGo.

1 Introduction

Contrary to ordinary classification, in multi-label classification (MLC) one can assign more than one class label to each example [1]. For instance, when we have the earth's continents as classes, a news article about the French and American interference in Libya could be labeled with the *Africa*, *Europe*, and *North America* classes. Originally, the main motivation for the multi-label approach came from the fields of medical diagnosis and text categorization, but nowadays multi-label methods are required by applications as diverse as music categorization, semantic scene classification, and protein function classification.

Many approaches to MLC take a decompositive approach, i.e., they decompose the MLC problem into a series of ordinary classification problems. The formulation of these problems often ignores interdependencies between labels, implying that the predictive performance may improve if label dependencies are taken into account. When, for instance, one considers a dataset where each label details the presence or absence of one kind of species in a certain region, the food chains between the species cause a plethora of strong correlations between labels. But interplay between species is more subtle than just correlations between pairs

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 114–125, 2012.
© Springer-Verlag Berlin Heidelberg 2012

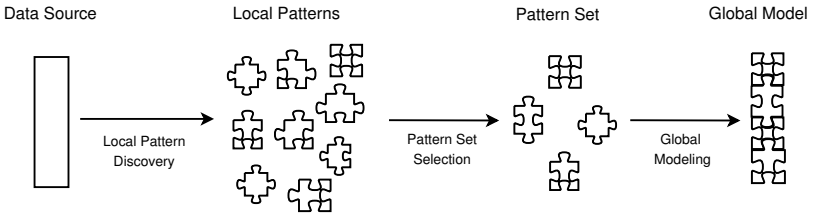


Fig. 1. The LeGo framework

of species. It has, for instance, been shown [2] that a food chain between two species (the sponge *Haliclona* and the nudibranch *Anisodoris*) may be displaced depending on whether a third species is present (the starfish *Pisaster ochraceus*), which is not directly related to the species in the food chain. Apparently, there is some conditional dependence relation between these three species. The ability to consider such interplay is an essential element of good multi-label classifiers.

In this paper we propose incorporating locally exceptional interactions between labels in MLC, as an instance of the LeGo framework [3]. In this framework, the KDD process is split up in several phases: first local models are found each representing only part of the data, then a subset of these models is selected, and finally this subset is employed in constructing a global model. The crux is that straight-forward classification methods can be used for building a global classifier, if the locally exceptional interactions between labels are represented by features constructed from patterns found in the local modeling phase.

We propose to find patterns representing these locally exceptional interactions through an instance of Exceptional Model Mining [4]; a framework that can be seen as an extension of traditional Subgroup Discovery. The instance we consider [5] models the conditional dependencies between the labels by a Bayesian network, and strives to find patterns for which the learned network has a substantially different structure than the network learned on the whole dataset. These patterns can each be represented by a binary feature of the data, and the main contribution of this paper is a demonstration that the integration of these features into the classification process improves classifier performance.

We refer the interested reader to a longer version of this paper covering additional aspects which could not be treated here due to space restrictions [6].

2 Preliminaries

In this section, we recall the cornerstones of our work: the LeGo framework for learning global models from local patterns (Section 2.1) and multi-label classification (Section 2.2). We conclude with the problem formulation (Section 2.3).

2.1 The LeGo Framework

As mentioned, the work in this paper relies heavily on the LeGo framework [3]. This framework assumes that the induction process is not executed by running a

single learning algorithm, but rather consists of a number of consecutive phases, as illustrated in Figure 1. In the first phase a local pattern discovery algorithm is employed in order to obtain a number of informative patterns, which can serve as relevant features to be used in the subsequent phases. These patterns can be considered partial solutions to local complexities in the data. In the second and third phase, the patterns are filtered to reduce redundancy, and the selected patterns are combined in a final global model, which is the outcome of the process.

The main reason to invest the additional computational cost of a LeGo approach over a single-step algorithm, is the expected increase in accuracy of the final model, caused by the higher level of exploration involved in the initial local pattern discovery phase. Typically, global modeling techniques employ some form of greedy search, and in complex tasks, subtle interactions between attributes may be overlooked as a result of this. In most pattern mining methods however, extensive consideration of combinations of attributes is quite common. When employing such exploratory algorithms as a form of preprocessing, one can think of the result (the patterns) as partial solutions to local complexities in the data. The local patterns, which can be interpreted as new virtual features, still need to be combined into a global model, but potentially hard aspects of the original representation will have been accounted for. As a result, straightforward methods such as Support Vector Machines with linear kernels can be used in the global modeling phase.

The LeGo approach has shown its value in a range of settings [3], particularly regular binary classification [7,8], but we have specific reasons for choosing this approach in the context of multi-label classification (MLC). It is often mentioned that in MLC, one needs to take into consideration potential interactions between the labels, and that simultaneous classification of the labels may benefit from knowledge about such interactions [9,10].

In a previous publication [5], we have outlined an algorithm finding local interactions amongst multiple targets (labels) by means of an Exceptional Model Mining (EMM) instance. The EMM framework [4] suggests a discovery approach involving multiple targets, using local modeling over the targets in order to find subsets of the dataset where unusual (joint) target distributions can be observed. In [5], we presented one instance of EMM that deals with discrete targets, and employs Bayesian Networks in order to find patterns corresponding to unusual dependencies between targets. This Bayesian EMM instance quenches the thirst in MLC for representations of locally unusual combinations of labels.

2.2 Multi-label Classification

Throughout this paper we assume a dataset Ω. This is a bag of N elements (*data points*) of the form $x = \{a_1, \ldots, a_k, \ell_1, \ldots, \ell_m\}$, where k and m are positive integers. We call $a_1, \ldots, a_k$ the *attributes* of x, and $\ell_1, \ldots, \ell_m \in \mathcal{L}$ the *labels* of x. Each label ℓ_i is assumed to be discrete, and the vectors of attributes are taken from an unspecified domain $\mathcal{A}$. Together we call the attributes and labels of x the *features* of x. When necessary, we distinguish the ith data point from other data points by adding a superscript i to the relevant symbols.

Attributes	Labels $\in \{0,1\}^m$
$a_1^1, \ldots, a_k^1$	$\ell_1^1, \ldots, \ell_m^1$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n$	$\ell_1^n, \ldots, \ell_m^n$

(a) input training set

Attributes	Class $\in 2^{\mathcal{L}}$
$a_1^1, \ldots, a_k^1$	$\mathcal{Y}^1$
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	$\mathcal{Y}^n$

(b) *label powerset* (LP) decomposition

Attributes	Class$_1 \in \{0,1\}$
$a_1^1, \ldots, a_k^1$	ℓ_1^1
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	ℓ_1^n

$\cdots$

Attributes	Class$_m \in \{0,1\}$
$a_1^1, \ldots, a_k^1$	ℓ_m^1
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	ℓ_m^n

(c) *binary relevance* (BR) decomposition

Fig. 2. Decomposition of multi-label training sets into binary (BR) or multiclass problems (LP). $\mathcal{Y}^i = \{y^i_1, \ldots, y^i_{|\mathcal{Y}^i|} |\ y^i_j \in \mathcal{L}\}$ denotes the assigned labels $\{\ell_j |\ \ell^i_j = 1\}$ to example x^i. In LP the (single) target value of an instance x^i is from the set $\{\mathcal{Y}^i |\ i = 1 \ldots m\} \subseteq 2^{\mathcal{L}}$ of the different label subsets seen in the training data.

The task of multi-label classification (MLC) is, given a training set $\mathcal{E} \subset \Omega$, to learn a function $f(a_1, \ldots, a_k) \to (\ell_1, \ldots, \ell_m)$ which predicts the labels for a given example. Many multi-label learning techniques reduce this problem to ordinary classification. The widely used *binary relevance* (BR) [1] approach tackles a multi-label problem by learning a separate classifier $f_i(a_1, \ldots, a_k) \to \ell_i$ for each label ℓ_i, as illustrated in Figure 2c. At query time, each binary classifier predicts whether its class is relevant for the query example or not, resulting in a set of relevant labels. Obviously, BR ignores possible interdependencies between classes since it learns the relevance of each class independently. One way of addressing this problem is by using *classifier chains* (CC) [10], which are able to model label dependencies since they stack the outputs of the models: the prediction of the model for label ℓ_i depends on the predictions for labels $\ell_1, \ldots, \ell_{i-1}$.

An alternative approach is *calibrated label ranking* (CLR) [11], where the key idea is to learn one classifier for each binary comparison of labels. CLR learns binary classifiers $f_{ij}(a_1, \ldots, a_k) \to (\ell_i \succ \ell_j)$, which predict for each label pair (ℓ_i, ℓ_j) whether ℓ_i is more likely to be relevant than ℓ_j. Thus, CLR (implicitly) takes correlations between pairs of labels into account. In addition, the decomposition into pairs of classes has the advantage of simpler sub-problems and hence commonly more accurately performing models. Finally, a simple way to take label dependencies into account is the *label powerset* (LP) approach [1], treating each combination of labels occuring in the training data as a separate label of a classification problem (Figure 2b).

We will use each of these techniques for decomposing a multi-label problem into an ordinary classification problem in the third LeGo phase (Section 4).

2.3 Problem Statement

The main question this paper addresses is whether a LeGo approach can improve multi-label classification, compared to existing methods that do not employ a preliminary local pattern mining phase. Thus, our approach encompasses:

1. find a set P of patterns representing local anomalies in conditional dependence relations between labels, using the method introduced in [5];
2. filter out a meaningful subset $S \subseteq P$;
3. use the patterns in S as constructed features to enhance multi-label classification methods.

In this paper we will use sophisticated methods in phases 1 and 3. In phase 2, we simply draw S as a random sample of P. Alternative methods were investigated in [6] but they did not provide relevant insights for our purpose. The following two sections will explore what we do in phases 1 and 3.

3 Local Pattern Discovery Phase

To find the local patterns with which we will enhance the MLC feature set, we employ an instance of Exceptional Model Mining (EMM). This instance is tailored to find subgroups in the data where the conditional dependence relations between a set of target features (our labels) is significantly different from those relations on the whole dataset. Before we recall the EMM instance in more detail, we will outline the general EMM framework.

3.1 Exceptional Model Mining

Exceptional Model Mining is a framework that can be considered an extension of the traditional Subgroup Discovery (SD) framework, a supervised learning task which strives to find *patterns* (defined on the input variables) that satisfy a number of user-defined *constraints*. A pattern is a function $p : \mathcal{A} \to \{0, 1\}$, which is said to *cover* a data point x^i if and only if $p\left(a_1^i, \ldots, a_k^i\right) = 1$. We refer to the set of data points covered by a pattern p as the *subgroup* corresponding to p. The *size* of a subgroup is the number of data points the corresponding pattern covers. The user-defined constraints typically include lower bounds on the subgroup size and on the quality of the pattern, which is usually defined on the output variables. A run of an SD algorithm results in a quality-ranked list of patterns satisfying the user-defined constraints.

In traditional SD, we have only a single target variable. The quality of a subgroup is typically gauged by weighing its target distribution deviation and its size. EMM extends SD by allowing for more complex target concepts defined on multiple target variables. It partitions the features into two sets: the attributes and the labels. On the labels a model class is defined, and an exceptionality measure φ for that model class is selected. Such a measure assigns a quality value $\varphi(p)$ to a candidate pattern p. EMM algorithms traverse a search lattice of candidate patterns, constructed on the attributes, in order to find patterns that have exceptional values of φ on the labels.

3.2 Exceptional Model Mining Meets Bayesian Networks

As discussed in Section 2, we assume a partition of the $k + m$ features in our dataset into k attributes, which can be from any domain, and m labels, which are assumed to be discrete. The EMM instance we employ [5] proposes to use Bayesian networks (BNs) over those m labels as model class. These networks are directed acyclic graphs (DAGs) that model the conditional dependence relations between their nodes. A pattern has a model that is exceptional in this setting, when the conditional dependence relations between the m labels are significantly different on the data covered by the pattern than on the whole dataset. Hence the exceptionality measure needs to measure this difference. We will employ the Weighed Entropy and Edit Distance measure (denoted φ_{weed}), as introduced in [5]. This measure indicates the extent to which the BNs differ in structure. Because of the peculiarities of BNs, we cannot simply use traditional edit distance between graphs [12] here. Instead, a variant of edit distance for BNs was introduced, that basically counts the number of violations of the famous theorem by Verma and Pearl on the conditions for equivalence of DAG models [13]:

Theorem 1 (Equivalent DAGs). *Two DAGs are equivalent if and only if they have the same skeleton and the same v-structures.*

Since these two conditions determine whether two DAGs are equivalent, it makes sense to consider the number of differences in skeletons and v-structures as a measure of how different two DAGs are. For more details on φ_{weed}, see [5].

After running the EMM algorithm, we obtain a set P of patterns each representing a local exceptionality in the conditional dependence relations between the m labels, hence completing the Local Pattern Discovery phase.

4 Global Modeling Phase

As stated in Section 2.3, we do nothing sophisticated in the Pattern Set Selection phase. Instead, we filter out a pattern subset $S \subseteq P$ by taking a random sample from P. In the current section, we use this subset S to learn a global model.

For the learning of the global multi-label classification models in the Global Modeling phase, we experiment with standard approaches including binary relevance (BR) and label powerset (LP) decompositions [1], as well as effective recent state-of-the-art learners such as calibrated label ranking (CLR) [11], and classifier chains (CC) [10]. The chosen algorithms cover a wide range of approaches and techniques used for learning multi-label problems (see Section 2.2), and are all included in Mulan, a library for multi-label classification algorithms [1].

For each classifier configuration, we learn three classifiers based on different feature sets. The first classifier uses the k features that make up the original dataset, and is denoted C_O (Figure 3a). The second classifier, denoted C_S, uses features constructed from our pattern set S. Each of these patterns maps each record in the original dataset to either zero or one. Hence they can be trivially transformed into binary features, that together make up the feature space for classifier C_S (Figure 3b). The third classifier employs both the k original and $|S|$ constructed features, in the spirit of LeGo, and is hence denoted C_L.

Attributes	Labels
$a_1^1, \ldots, a_k^1$	$\ell_1^1, \ldots, \ell_m^1$
$a_1^2, \ldots, a_k^2$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n$	$\ell_1^n, \ldots, \ell_m^n$

(a) input training set C_O

Attributes	Labels
$p_1\left(a_1^1, \ldots, a_k^1\right), \ldots, p_{\lvert S\rvert}\left(a_1^1, \ldots, a_k^1\right)$	$\ell_1^1, \ldots, \ell_m^1$
$p_1\left(a_1^2, \ldots, a_k^2\right), \ldots, p_{\lvert S\rvert}\left(a_1^2, \ldots, a_k^2\right)$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$p_1\left(a_1^n, \ldots, a_k^n\right), \ldots, p_{\lvert S\rvert}\left(a_1^n, \ldots, a_k^n\right)$	$\ell_1^n, \ldots, \ell_m^n$

(b) transformation into pattern space C_S

Attributes	Labels
$a_1^1, \ldots, a_k^1, p_1\left(a_1^1, \ldots, a_k^1\right), \ldots, p_{\lvert S\rvert}\left(a_1^1, \ldots, a_k^1\right)$	$\ell_1^1, \ldots, \ell_m^1$
$a_1^2, \ldots, a_k^2, p_1\left(a_1^2, \ldots, a_k^2\right), \ldots, p_{\lvert S\rvert}\left(a_1^2, \ldots, a_k^2\right)$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots \quad \vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n, p_1\left(a_1^n, \ldots, a_k^n\right), \ldots, p_{\lvert S\rvert}\left(a_1^n, \ldots, a_k^n\right)$	$\ell_1^n, \ldots, \ell_m^n$

(c) combined attributes in the LeGo classifier C_L

Fig. 3. A multi-label classification problem (a), its representation in pattern space (b) given the set of patterns $p_1, \ldots, p_{|S|}$, and the LeGo combination (c)

5 Experimental Setup

To experimentally validate the outlined LeGo method, we will compare the performance of the three classifiers based on different feature sets C_O, C_S, and C_L. We refer the reader to the longer version of the paper for a differentiating analysis of the results regarding the performance of the decomposition approaches, the impact on the different multi-label measures and the beneficial effect of using the binary LeGo patterns on efficiency [6].

For the experiments we selected three multi-labeled datasets from different domains. Statistics on these datasets can be found in Table 1. The column *Cardinality* displays the average number of relevant labels for a data point.

We combine the multi-label decomposition methods mentioned in Section 4 with several base learners: J48 with default settings [14], standard LibSVM [15], and LibSVM with a grid search on the parameters. In this last approach, multiple values for the SVM kernel parameters are tried, and the one with the best 3-fold cross-validation accuracy is selected for learning on the training set (as suggested by [15]). Both SVM methods are run once with the Gaussian Radial Basis Function as kernel, and once with a linear kernel using the efficient LibLinear implementation [16]. We will refer to LibSVM with the parameter grid search as MetaLibSVM, and denote the used kernel by a superscript *rbf* or *lin*.

Table 1. Datasets used in the experiments, shown with the number of examples (N), attributes (k), and labels (m), as well as the average number of labels per example

Dataset	Domain	N	k	m	Cardinality
Emotions	Music	593	72	6	1.87
Scene	Vision	2407	294	6	1.07
Yeast	Biology	2417	103	14	4.24

5.1 Experimental Procedure

All statistics on the classification processes are estimated via a 10-fold cross-validation. To enable a fair comparison of the LeGo classifier with the other classifiers, we let the entire learning process consider only the training set for each fold. This means that we have to run the Local Pattern Discovery and Pattern Subset Discovery phase separately for each fold.

For every fold on every dataset, we determine the best 10,000 patterns, measuring the exceptionality with φ_{weed} as described in Section 3.2. The search space in EMM cannot be explored exhaustively when there are numerical attributes and a nontrivial quality measure, and both are the case here. Hence we resort to a beam search strategy, configured with a beam width of $w = 10$ and a maximum search level of 2 (for more details on beam search in EMM, see [5]). We specifically select a search of modest depth, in order to prevent producing an abundance of highly similar patterns. We further bound the search by setting the minimal coverage of a pattern at 10% of the dataset.

For each dataset for each fold, we train classifiers from the three training sets C_O, C_S, and C_L for each combination of a decomposition approach and base learner. We randomly select $|S| = k$ patterns (cf. Section 4), i.e. exactly as many pattern-based features for C_S and C_L as there are original features in C_O.

5.2 Evaluation Measures

We evaluate the effectiveness of the three classifiers for each combination on the respective test sets for each fold with five measures: Micro-Averaged Precision and Recall, Subset Accuracy, Ranking Loss, and Average Precision (for details on computation cf. [11] and [1]). We find these five measures a well balanced selection from the vast set of multi-label measures, evaluating different aspects of multi-label predictions such as good ranking performance and correct bipartition.

From a confusion matrix aggregated over all labels and examples, *Precision* computes the percentage of predicted labels that are relevant, and *Recall* computes the percentage of relevant labels that are predicted. *Subset Accuracy* denotes the percentage of perfectly predicted label sets, basically forming a multi-label version of traditional accuracy. We also computed the following rank-based loss measures. *Ranking Loss* returns the number of pairs of labels which are not correctly ordered, normalized by the total number of pairs. *Average Precision* computes the precision at each relevant label in the ranking, and averages these

Table 2. Average ranks r_i of the three classifiers $C_i, i \in \{O, S, L\}$, with critical difference CD, over all test configurations, and over all test configurations barring J48

	r_O	r_S	r_L	CD
Overall	1.863	2.340	1.797	0.191
Without J48	1.971	2.296	1.733	0.214

percentages over all relevant labels. These two ranking measures are computed for each example and then averaged over all examples.

All values for all settings are averaged over the folds of the cross-validation. Thus we obtain 300 test cases (5 evaluation measures $\times$ 5 base learners $\times$ 4 decomposition approaches $\times$ 3 datasets). To draw conclusions from these raw results, we use the Friedman test with post-hoc Nemenyi test [17].

6 Experimental Results

Table 2 compares the three different representations C_O, C_S, and C_L over the grand total of 300 test cases in terms of average ranks.[1] We see that both C_O and C_L perform significantly ($\alpha = 5\%$) better than C_S, i.e. the pattern-only classifier cannot compete with the original features or the combined classifier. However, when we consider only the LibSVM$^{\text{rbf}}$ base learner, we find that the pattern-only classifier outperforms the classifier trained on original features.

The difference in performance between C_O and C_L is not significant. Although the average rank for the LeGo-based classifier is somewhat higher, we cannot claim that adding local patterns leads to a significant improvement. However, when splitting out the results for the different base learners, we notice a striking difference in average ranks between J48 and the rest. When we restrict ourselves to the results obtained with J48, we find that $r_O = 1.433$, $r_S = 2.517$, and $r_L = 2.050$, with $CD = 0.428$. Here, the classifier built from original features significantly ($\alpha = 5\%$) outperforms the LeGo classifier.

One reason for the performance gap between J48 and the SVM approach lies in the way these approaches construct their decision boundary. The SVM approaches draw one hyperplane through the attribute space, whereas J48 constructs a decision tree, which corresponds to a decision boundary consisting of axis-parallel fragments. The patterns the EMM algorithm finds in the Local Pattern Discovery phase are constructed by several conditions on single attributes. Hence the domain of each pattern has a shape similar to a J48 decision boundary, unlike a (non-degenerate) SVM decision boundary. Hence, the expected performance gain when adding such local patterns to the attribute space is much higher for the SVM approaches than for the J48 approach.

Because the J48 approach results in such deviating ranks, we investigate the relative performance of the base learners. We compare their performance on the

[1] The results were consistent over all 5 measures with respect to the used feature sets so we did not further differentiate, cf. also [6].

Table 3. Average ranks of the base learners, with critical difference CD

Approach	MetaLibSVM$^{\text{rbf}}$	MetaLibSVM$^{\text{lin}}$	LibSVM$^{\text{lin}}$	LibSVM$^{\text{rbf}}$	J48	CD
Rank	1.489	2.972	3.228	3.417	3.894	0.455

three classifiers C_O, C_S, and C_L, with decomposition methods BR, CC, CLR, and LP, on the datasets from Table 1, evaluated with the measures introduced in Section 5.1. The average ranks of the base learners over these 180 test cases can be found in Table 3; J48 performs significantly worse than all SVM methods.

We have determined that the performance difference between C_O and C_L is not significant. In order to see if we can make a weaker statement of significance between C_O and C_L, and having just established that this is the worst-performing base learner, we repeat our comparison of the classifiers C_O, C_S, and C_L on the four base learner approaches that perform best: the SVM variants. The average ranks of the three classifiers on these 240 test cases can be found in the last row of Table 2. On the SVM methods, the LeGo classifier is significantly ($\alpha = 5\%$) better than the classifier built from original features.

As stated in Section 2.2, to predict label ℓ_i the CC decomposition approach allows using the predictions made for labels $\ell_1, \ldots, \ell_{i-1}$. Hence we can view CC as a feature enriching approach, adding a feature set C. We find that adding C has an effect on performance similar to adding S, which is amplified when both are added, particularly for BR. Hence the patterns in S provide additional information on the label dependencies which is not covered by C. This aspect is also treated in more detail in the longer version of this paper [6].

7 Conclusions

We have proposed enhancing multi-label classification methods with local patterns in a LeGo setting. These patterns are found through an instance of Exceptional Model Mining, a generalization of Subgroup Discovery striving to find subsets of the data with aberrant conditional dependence relations between target features. Hence each pattern delivered represents a local anomaly in conditional dependence relations between targets. Each pattern corresponds to a binary feature which we add to the dataset, to improve classifier performance.

Experiments on three datasets show that for multi-label SVM classifiers the performance of the LeGo approach is significantly better than the traditional classification performance: investing extra time in running the EMM algorithm pays off when the resulting patterns are used as constructed features. The J48 classifier does not benefit from the local pattern addition, which can be attributed to the similarity of the local decision boundaries produced by the EMM algorithm to those produced by the decision tree learner. Hence the expected performance gain when adding local patterns is lower for J48 than for approaches that learn different types of decision boundaries, such as SVM approaches.

The Friedman-Nemenyi analysis also shows that the constructed features generally cannot *replace* the original features without significant loss in classification

performance. We find this reasonable, since these features are constructed from patterns found by a search process that is not at all concerned with the potential of the patterns for classification, but is focused on exceptionality. In fact, the pattern set may be highly redundant. Additionally it is likely that the less exceptional part of the data, which by definition is the majority of the dataset, is underrepresented by the constructed features.

To the best of our knowledge, this is a first shot at discovering multi-label patterns and testing their utility for classification in a LeGo setting. Therefore this work can be extended in various ways. It might be interesting to develop more efficient techniques without losing performance. One could also explore other quality measures, such as the plain edit distance measure from [5], or other search strategies. In particular, optimizing the beam-search in order to properly balance its levels of exploration and exploitation, could fruitfully produce a more diverse set of features [18] in the Local Pattern Discovery phase. Alternatively, pattern diversity could be addressed in the Pattern Subset Selection phase, ensuring diversity within the subset S rather than enforcing diversity over the whole pattern set P.

As future work, we would like to expand our evaluation of these methods. Recently, it has been suggested that for multi-label classification, it is better to use stratified sampling than random sampling when cross-validating [19]. Also, experimentation on more datasets seems prudent. In this paper, we have experimented on merely three datasets, selected for having a relatively low number of labels. As stated in Section 3.2, we have to fit a Bayesian network on the labels for each subgroup under consideration, which is a computationally expensive operation. The availability of more datasets with not too many labels (say, $m < 50$) would allow for more thorough empirical evaluation, especially since it would allow us to draw potentially significant conclusions from Friedman and Nemenyi tests per evaluation measure per base classifier per decomposition scheme. With three datasets this would be impossible, so we elected to aggregate all these test cases in one big test. The observed consistent results over all evaluation measures provide evidence that this aggregation is not completely wrong, but theoretically this violates the assumption of the tests that all test cases are independent. Therefore, the empirically drawn conclusions in this paper should not be taken as irrefutable proof, but more as evidence contributing to our beliefs.

Acknowledgments. This research is financially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.065.822 (Exceptional Model Mining) and by the German Science Foundation (DFG). We also gratefully acknowledge support by the Frankfurt Center for Scientific Computing.

References

1. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining Multi-label Data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer (2010)
2. Paine, R.T.: Food Web Complexity and Species Diversity. The American Naturalist 100(910), 65–75 (1966)

3. Fürnkranz, J., Knobbe, A. (eds.): Special Issue: Global Modeling Using Local Patterns. Data Mining and Knowledge Discovery Journal 20 (1) (2010)
4. Leman, D., Feelders, A., Knobbe, A.: Exceptional Model Mining. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 1–16. Springer, Heidelberg (2008)
5. Duivesteijn, W., Knobbe, A., Feelders, A., van Leeuwen, M.: Subgroup Discovery meets Bayesian networks – an Exceptional Model Mining approach. In: Proc. ICDM, pp. 158–167 (2010)
6. Duivesteijn, W., Loza Mencía, E., Fürnkranz, J., Knobbe, A.: Multi-label LeGo — Enhancing Multi-label Classifiers with Local Patterns. Technical Report, Technische Universität Darmstadt, TUD-KE-2012-02 (2012), http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2012-02.pdf
7. Knobbe, A., Valkonet, J.: Building Classifiers from Pattern Teams. In: From Local Patterns to Global Models: Proc. ECML PKDD 2009 Workshop, Slovenia (2009)
8. Sulzmann, J.-N., Fürnkranz, J.: A Comparison of Techniques for Selecting and Combining Class Association Rules. In: From Local Patterns to Global Models: Proc. ECML PKDD 2008 Workshop, Belgium (2008)
9. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
10. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier Chains for Multi-label Classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 254–269. Springer, Heidelberg (2009)
11. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel Classification via Calibrated Label Ranking. Machine Learning 73(2), 133–153 (2008)
12. Shapiro, L.G., Haralick, R.M.: A Metric for Comparing Relational Descriptions. IEEE Trans. Pattern Anal. Mach. Intell. 7, 90–94 (1985)
13. Verma, T., Pearl, J.: Equivalence and Synthesis of Causal Models. In: Proc. UAI, pp. 255–270 (1990)
14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
15. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines, Software, http://www.csie.ntu.edu.tw/~cjlin/libsvm
16. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9, 1871–1874 (2008)
17. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)
18. van Leeuwen, M., Knobbe, A.: Non-redundant Subgroup Discovery in Large and Complex Data. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 459–474. Springer, Heidelberg (2011)
19. Sechidis, K., Tsoumakas, G., Vlahavas, I.: On the Stratification of Multi-label Data. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 145–158. Springer, Heidelberg (2011)

Discriminative Dimensionality Reduction Mappings

Andrej Gisbrecht, Daniela Hofmann, and Barbara Hammer

University of Bielefeld - CITEC Centre of Excellence, Germany
bhammer@techfak.uni-bielefeld.de

Abstract. Discriminative dimensionality reduction aims at a low dimensional, usually nonlinear representation of given data such that information as specified by auxiliary discriminative labeling is presented as accurately as possible. This paper centers around two open problems connected to this question: (i) how to evaluate discriminative dimensionality reduction quantitatively? (ii) how to arrive at explicit nonlinear discriminative dimensionality reduction mappings? Based on recent work for the unsupervised case, we propose an evaluation measure and an explicit discriminative dimensionality reduction mapping using the Fisher information.

1 Introduction

The amount of electronic data available today increases rapidly, such that humans rely on automated tools which allow them to intuitively scan data volumes for valuable information. Dimensionality reducing data visualization, which displays high dimensional data in two or three dimensions, constitutes a popular tool to directly visualize data sets on the computer screen, see e.g. [2,14,19]. Dimensionality reduction is an inherently ill-posed problem, and the result of a dimensionality reduction tool largely varies depending on the chosen technology, the parameters, and partially even random aspects for non-deterministic algorithms. Often, the reliability and suitability of the obtained visualization for the task at hand is not clear at all since a dimensionality reduction tool might focus on irrelevant aspects or noise in the data. Thereby, mathematical objectives of dimensionality reduction techniques which guide the final output of the algorithms are often unintuitive and not accessible for users outside the field.

Discriminative dimensionality reduction, i.e. the integration of auxiliary information by an explicit labeling of data can help to partially overcome these problems: in discriminative dimensionality reduction, the aim is to visualize those aspects of the data which are particularly relevant for the given class information. Thus, the information which is neglected by the dimensionality reduction method is no longer arbitrary but directly linked to its relevance for the given classes. Since auxiliary labels or classes are often available and directly interpretable by humans, this offers a natural interface for applicants to shape the focus of the dimensionality reduction as required by the respective application.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 126–138, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Several linear supervised projection methods exist, such as Fisher's linear discriminant analysis (LDA), partial least squares regression (PLS), informed projections [4], or global linear transformations of the metric to include auxiliary information [8,3]. Modern techniques extend these settings to nonlinear projections of data. One way is offered by kernelization such as kernel LDA [16,1]. Alternative approaches incorporate auxiliary information to modify the cost function of dimensionality reducing data visualization. One such technique is the supervised multidimensional scaling (SMDS), which optimizes two terms simultaneously, one being classical MDS cost function and the second depending on the distances between labels [22]. Thus, being a globally linear mapping, it modifies the local structure to ensure class separation. The approaches introduced in [10,17] can both be understood as extensions of stochastic neighbor embedding (SNE), the latter minimizing the deviation of the data distributions in the original data space and projection space. Parametric embedding (PE) substitutes these distributions by conditional probabilities of classes, given a data point, this way mapping both, data points and class centers at the same time. Multiple relational embedding (MRE) incorporates several dissimilarity structures in the data space. Colored maximum variance unfolding (MVU) incorporates auxiliary information into MVU by substituting the raw data by the combination of the data and the covariance matrix induced by the given auxiliary information. Interesting locally linear, globally linear dimensionality reduction approaches based on prototype based techniques have recently been proposed in [2]. In [5], an ad hoc adaptation of the metric is used which also takes given class labeling into account. Many of these techniques are somewhat ad hoc, and they depend on critical parameters such as e.g the used kernel. Further, only few of these approaches provide an explicit dimensionality reduction mapping.

A principled way to extend dimensionality reducing data visualization to auxiliary information is offered by an adaptation of the underlying metric according to information theoretic principles. The principle of learning metrics has first been introduced in [11,18]. Here, the standard Riemannian metric is substituted by a form which measures the information of the data for the given classification task. The Fisher information matrix induces the local structure of this metric and it can be expanded globally in terms of path integrals. This metric is integrated into the self-organizing map (SOM), multidimensional scaling (MDS), and a recent information theoretic model for data visualization [11,18,21]. A drawback of the proposed method is its high computational complexity and its restriction to the given data only.

In this contribution, we propose to learn an explicit mapping of the data space to the low-dimensional projection space which takes the given class information into account in terms of the Fisher information. Thereby, we rely on recent extensions of the powerful unsupervised visualization technique t-distributed SNE (t-SNE) to kernel mappings [19,7,6]. Since the resulting technique, Fisher kernel t-SNE, can be trained on a subset of all data only, it scales well to large or streaming data sets by means of an out-of-sample extension using the explicit embedding function.

The principled approach to focus on the Fisher information opens another perspective to design and evaluate discriminative dimensionality reduction. So far, it has not yet been clear how to quantitatively evaluate discriminative dimensionality reduction. In the literature, evaluation often relies on human visual inspection only as reported e.g. in [20]. The virtually single quantitative measure used so far in applications relies on an evaluation of the classification error of the data in low dimensions, as measured e.g. by a simple k-nearest neighbor technique in the projection space [21]. Albeit this can give some hint in how far priorly known semantically meaningful clusters are preserved while projecting, this technique cannot evaluate in how far relevant topological structures of the original data are preserved. Indeed, the evaluation technique would rank those methods as best which achieve the highest classification accuracy e.g. by simply mapping the data to simple points corresponding to their labels. This way, relevant structure such as multi-modality or overlap of classes would be disrupted, such that the usefulness of the classification error as evaluation measure for discriminative dimensionality reduction is questionable. Here, we argue that recent quantitative evaluation measures for unsupervised dimensionality reduction as introduced in [13] can be transferred to discriminative dimensionality reduction by means of the Fisher information, displaying intuitive results in typical cases.

Now, we first shortly recall the Fisher information and a way of its estimation. Then we address quantitative evaluation measures for dimensionality reduction and show in illustrative cases that this evaluation measure in combination with the Fisher information provides intuitive results for the evaluation of discriminative dimensionality reduction. Afterwards, we provide an extension of the popular dimensionality reduction method t-SNE to give an explicit discriminative dimensionality reduction mapping of the data: Fisher kernel t-SNE. We demonstrate the approach and its generalization ability on benchmark examples.

2 The Fisher Metric

The Fisher information is a way of measuring the information of an observable variable for an unknown parameter. In our case, observables are given by the data in high dimensional space, the additional parameter is given by the auxiliary class labeling. The explicit class labeling should be used to specify which information of the data is relevant and, consequently, should be displayed in dimensionality reduction techniques. Interestingly, the Fisher information can serve as a universal bound for any unbiased estimator of the parameter, as specified by the Cramér-Rao bound. The Fisher distance is a Riemannian metric based on the local Fisher information in which the labeling information is used to emphasize the directions of the manifold according to the class distribution. It has been proposed e.g. in [11,18] as a technique to enhance visualization techniques such as the SOM or MDS to learning metrics induced by auxiliary information.

We assume data $\mathbf{x}$ are sampled leading to points $\mathbf{x}_i$. These are equipped with auxiliary class information c_i which are instances of a finite number of possible classes c. The data manifold is equipped with a Riemannian metric which takes

this class labeling into account. For this purpose, the tangent space at position $\mathbf{x}_i$ is equipped with the quadratic form

$$d_1(\mathbf{x}_i, \mathbf{x}_i + d\mathbf{x}) = (d\mathbf{x})^T \mathbf{J}(\mathbf{x}_i)(d\mathbf{x}).$$

Here $\mathbf{J}(\mathbf{x})$ denotes the Fisher information matrix

$$\mathbf{J}(\mathbf{x}) = E_{p(c|\mathbf{x})} \left\{ \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right)^T \right\}.$$

to quantify the information contained in the data which is relevant for the given labeling. As usual, a Riemannian metric on the data manifold is induced by minimum path integrals using these quadratic forms locally.

For practical applications, the Fisher information has to be estimated based on given data. In this work we estimate conditional probabilities $p(c|\mathbf{x})$ from the data $(\mathbf{x}_i, c_i)$ using the Parzen nonparametric estimator

$$\hat{p}(c|\mathbf{x}) = \frac{\sum_i \delta_{c=c_i} \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2/2\sigma^2)}{\sum_j \exp(-\|\mathbf{x} - \mathbf{x}_j\|^2/2\sigma^2)}.$$

The Fisher information matrix becomes

$$\mathbf{J}(\mathbf{x}) = \frac{1}{\sigma^4} E_{\hat{p}(c|\mathbf{x})} \left\{ \mathbf{b}(\mathbf{x}, c) \mathbf{b}(\mathbf{x}, c)^T \right\}$$

where

$$\mathbf{b}(\mathbf{x}, c) = E_{\xi(i|\mathbf{x},c)}\{\mathbf{x}_i\} - E_{\xi(i|\mathbf{x})}\{\mathbf{x}_i\}$$

$$\xi(i|\mathbf{x}, c) = \frac{\delta_{c,c_i} \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2/2\sigma^2)}{\sum_j \delta_{c,c_j} \exp(-\|\mathbf{x} - \mathbf{x}_j\|^2/2\sigma^2)}$$

$$\xi(i|\mathbf{x}) = \frac{\exp(-\|\mathbf{x} - \mathbf{x}_i\|^2/2\sigma^2)}{\sum_j \exp(-\|\mathbf{x} - \mathbf{x}_j\|^2/2\sigma^2)}.$$

Here, the operator E denotes the empirical expectation, i.e. weighted sums with weights depicted in the subscripts. If large data sets or out-of-sample extensions are dealt with, we will use a subset of the data only for the estimation of the Fisher matrix, i.e. the sums run over a fixed subset of labeled data $(\mathbf{x}_i, c_i)$.

In addition, path integrals have to be approximated in practical computations. Different ways to approximate these values have already been discussed in [18]. Since local information is most relevant in our applications, we can rely on the approximation of the Fisher distance by T-point distances as proposed in [18]. That means, we sample T equidistant points on the line from $\mathbf{x}_i$ to $\mathbf{x}_j$ and we approximate the Riemannian distance on the manifold by the term

$$d_T(\mathbf{x}_i, \mathbf{x}_j) = \sum_{t=1}^{T} d_1 \left(\mathbf{x}_i + \frac{t-1}{T}(\mathbf{x}_j - \mathbf{x}_i), \mathbf{x}_i + \frac{t}{T}(\mathbf{x}_j - \mathbf{x}_i) \right)$$

where $d_1(\mathbf{x}_i, \mathbf{x}_j)$ is the standard distance as evaluated in the tangent space of $\mathbf{x}_i$. Locally, this approximation gives good results such that a faithful dimensionality reduction of data can be based thereon.

This Fisher distance or its approximation, respectively, specifies the topology of the data space measured according to its relevance for the specified class labeling. It can now be integrated into dimensionality reduction techniques to extend the latter to discriminative variants, as proposed e.g. in [18] for SOM and MDS.

3 Evaluation of Discriminative Dimensionality Reduction

One crucial aspect of machine learning tools is how to evaluate the obtained results. Dimensionality reduction and data visualization being inherently ill-posed, evaluation measures eventually depend on the context at hand. Nevertheless, quantitative evaluation methods are necessary to automatically compare and evaluate dimensionality reducing data visualization on a large scale. Interestingly, as reported in [20], a high percentage of publications on data visualization evaluates results in terms of visual impression only – about 40% out of 69 papers referenced in [20] did not use any quantitative evaluation criterion. In the last years, a few formal mathematical evaluation measures of dimensionality reduction have been proposed in the literature. In this contribution, we will use the co-ranking framework proposed in [13,15] as a highly flexible and generic tool to evaluate the preservation of pairwise relationships when projecting data to low dimensions, which is accepted as a standard evaluation measure for unsupervised dimensionality reduction.

Assume points $\mathbf{x}_i$ are mapped to projections $\mathbf{y}_i$ using some dimensionality reduction technique. The co-ranking framework essentially evaluates, in how far neighborhoods in the original space and the projection space correspond to each other. Let δ_{ij} be the Euclidean distance of $\mathbf{x}_i$ and $\mathbf{y}_i$ and d_{ij} be the Euclidean distance of $\mathbf{y}_i$ and $\mathbf{y}_j$. The rank of $\mathbf{x}_j$ with respect to $\mathbf{x}_i$ is given by $\rho_{ij} = \{k | \delta_{ik} < \delta_{ij}$ or $(\delta_{ik} = \delta_{ij}$ and $k < j)\}|$. Analogously, the rank of r_{ij} for the projections can be defined based on d_{ij}. The co-ranking matrix Q [13] is defined by $Q_{kl} = |\{(i,j) \mid \rho_{ij} = k \text{ and } r_{ij} = l\}|$. Errors of a dimensionality reduction correspond to rank errors, i.e. off-diagonal entries in this matrix. Usually, the focus of a dimensionality reduction is on the preservation of local relationships. Therefore, small rank errors corresponding to neighborhood preservation for small ranks are considered. In [13], an intuitive sum has been proposed, the *Quality*

$$Q_{\mathrm{NX}}(K) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{l=1}^{K} Q_{kl}.$$

where N denotes the number of points. This summarizes all 'benevolent' points which change their rank only within a fixed neighborhood K. To display the quality, usually the quality curve is plotted for a range $K > 1$. A good visualization results corresponds to a curve with high values $Q_{\mathrm{NX}}(K)$ for small neighborhood

sizes K, i.e. a locally faithful preservation of ranks while projecting the data. Interestingly, this framework can be linked to an information theoretic point of view as specified in [21] and it subsumes several previous evaluation criteria [13].

Note that an evaluation of the coranking matrix is quite costly for large data sets. Therefore, a sampling strategy has recently been proposed and tested in [7]: The quality curve is repeatedly evaluated for a random subsample only, and the results are averaged. The resulting curve (if rescaled to the original range of $1 \leq K \leq N$) can be considered as good approximation of the original one.

In discriminative dimensionality reduction, data are accompanied by an explicit class labeling. An evaluation of a dimensionality reduction technique by means of the quality $Q_{\mathrm{NX}}(K)$ does not take these labels into account, i.e. loss of information in terms of rank errors is considered as error regardless of whether this information has an influence on the label or not. Typically, supervised dimensionality reduction techniques are currently either evaluated only by visual inspection, or they are evaluated by referring to the classification error of the data in the projection space. In this way, it can be tested in how far the projections are conform with the labeling. Typically, the error of simple classifiers such as k-nearest neighbor classifiers in the low dimensional projection space is measured for this purpose, see e.g. [21]. Naturally, this evaluation is meaningful only if all aspects relevant for visualization are contained in the class labels itself, and dimensionality reduction is basically turned into a supervised classification problem: Indeed, a visualization is considered as optimum if data are mapped directly to their labels in this context, neglecting all other topological information such as e.g. overlapping classes or multimodal classes.

Therefore, we propose to integrate the Fisher metric into this framework. Obviously, it is directly possible to compute Q_{NX} based on the distance d_T in the original space instead of the standard Euclidean distance. This way, the quality measures in how far rankings as induced by the information which is relevant for the given labeling are preserved when projecting the data to low dimensionality. This evaluation measure, which we refer to as $Q_{\mathrm{Fisher-NX}}$, thus combines both aspects: is the original topology of the data preserved to a sufficient extent, and, if not, is the information which is marked as relevant by the label information preserved. In particular, information such as overlap of given classes and multimodality is preserved by this evaluation measure.

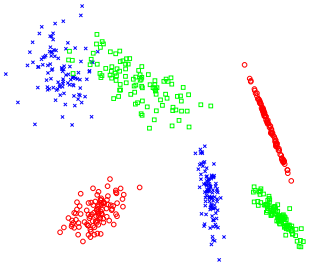
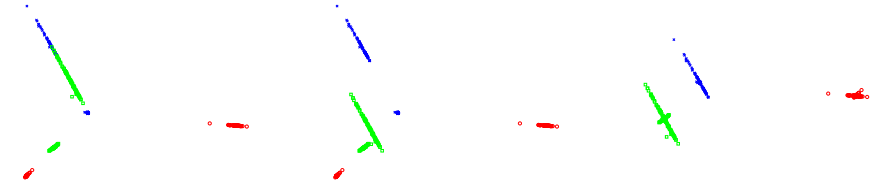
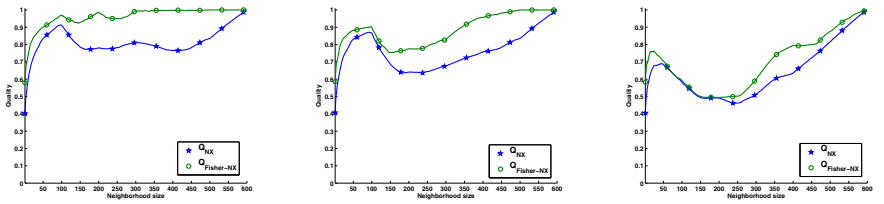


Fig. 1. Toy data used as a demonstration of the behavior of the quality evaluation based on the Fisher distance



(a) MDS on Fisher distance (b) resolve cluster overlap (c) single modes



(d) evaluation of MDS (e) evaluation overlap (f) evaluation single modes

Fig. 2. 'Projection' of data using the Fisher metric and its evaluation

We demonstrate this claim in an intuitive example, see Fig. 1. Two-dimensional data contained in three classes with two modes each are considered. We evaluate three mappings using Q_{NX} and $Q_{\mathrm{Fisher-NX}}$. These are mapped to two dimensions using classical multidimensional scaling where the distance of the original data is given by the Fisher metric, see 2a. In consequence, a distortion of the classes takes place such that multimodality, cluster separation, and cluster overlap are preserved, but cluster widths are shrinked locally in the direction which is not relevant for the class label. Based on this MDS projection, we replace some of the modes by hand such that the overlap of two clusters is resolved (see 2b), or data of every class are mapped to one mode only (see 2c. We evaluate all settings using Q_{NX} and $Q_{\mathrm{Fisher-NX}}$. Note that a value close to 1 for a given neighborhood K corresponds to the fact that the corresponding measure regards neighborhoods of size K as perfectly preserved by the projection method. Unlike Q_{NX}, the measure $Q_{\mathrm{Fisher-NX}}$ accepts distortions of the data which do not affect the labeling information (see 2d). Still, it takes care that topological distortions of the projection are counted as error if they do affect the relationship of data and labels, such as a change of the number of modes of a class (see 2f) or a display of originally overlapping classes as non overlapping (see 2e). Both changes lead to a decrease of the measure $Q_{\mathrm{Fisher-NX}}$ as well as the original Q_{NX}. Note that, for both settings, the classification accuracy of the projection would be increased, i.e. an evaluation of the discriminative projection by means of the classification error only is not capable of detecting these topological distortions. Hence $Q_{\mathrm{Fisher-NX}}$ seems a good way to quantitatively evaluate discriminative dimensionality reduction results.

4 Discriminative Kernel t-SNE

The t-distributed stochastic neighbor embedding (t-SNE) has been proposed in [19] as a highly flexible dimensionality reduction technique. It preserves probabilities as induced by pairwise distances in the data and projection space. Data points $\mathbf{x}_i$ in the original space induce pairwise probabilities $p_{ij} = (p_{(i|j)} + p_{(j|i)})/(2N)$ where N is the number of data points and

$$p_{j|i} := \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}.$$

The bandwidth parameter is set locally such that the effective number of neighbors corresponds to a reasonable fraction of the data set. Projections $\mathbf{y}_i$ induce pairwise probabilities

$$q_{ij} := \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}.$$

t-SNE aims at finding projections $\mathbf{y}_i$ such that the difference of these probabilities as measured by the Kullback-Leibler divergence is minimized, using a gradient technique [19]. Obviously, this technique does not provide an explicit mapping $\mathbf{x} \to \mathbf{y} = \mathbf{y}(\mathbf{x})$. Hence, when dealing with novel data points, a new optimization problem has to be solved. Another problem of t-SNE is given by its computational costs which are quadratic. Therefore, the technique can hardly be applied to large data sets.

Therefore, it has been proposed to extend t-SNE to an explicit kernel mapping, kernel-t-SNE, in [7]. This mapping is characterized by the function:

$$\mathbf{x} \mapsto \mathbf{y}(\mathbf{x}) = \sum_j \alpha_j \cdot \frac{k(\mathbf{x}, \mathbf{x}_j)}{\sum_l k(\mathbf{x}, \mathbf{x}_l)}$$

with parameters $\alpha_j \in \mathbb{R}^d$. The points $\mathbf{x}_j$ are taken from a fixed sample of data points used to train the mapping. k is an appropriate kernel function such as the Gaussian kernel, which we will use in the following. The parameters α_j of the mapping have to be determined based on a given training sample $\mathbf{x}_i$ of points. In [7], different techniques have been tested, a very simple one usually leading to competitive results, which we will use in the following: First, the sample points $\mathbf{x}_i$ are mapped to projections $\mathbf{y}_i$ by means of standard t-SNE. Then, mapping parameters α_j are determined to minimize the sum squared error of these projections $\mathbf{y}_i$ and the function values $\mathbf{y}(\mathbf{x}_i)$. An explicit solution of the matrix $\mathbf{A}$ of parameters α_j can be obtained as

$$\mathbf{A} = \mathbf{K}^{-1}\mathbf{Y},$$

where $\mathbf{K}$ is the normalized Gram matrix with entries $k(\mathbf{x}_i, \mathbf{x}_j)/\sum_j k(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{Y}$ denotes the matrix of projections $\mathbf{y}_i$, and $\mathbf{K}^{-1}$ refers to the pseudo-inverse.

It has been demonstrated in [7,6] that this technique offers a possibility which extends the excellent behavior of t-SNE from a small training set to the full data space, thereby usually displaying very good generalization ability. A problem occurs, however, if regularities of the data such as e.g. clusters are not yet clearly pronounced if the data are represented by a small subset only. In these cases, t-SNE does not show the characteristics of the data if trained for a small subset only. Here auxiliary information by explicit class labeling can help to provide this missing information without the necessity to use a larger training set for the dimensionality reduction mapping kernel-t-SNE.

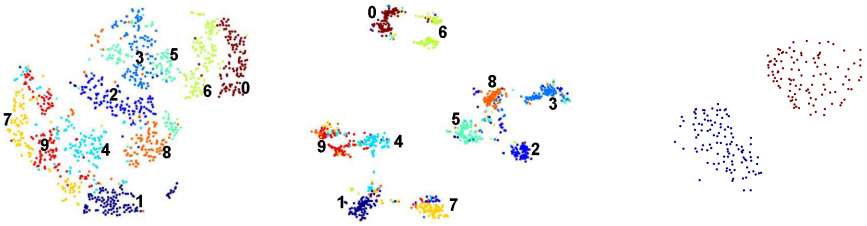
Note that it is possible to extend kernel-t-SNE mapping to a discriminative dimensionality reduction mapping by referring to the Fisher distance, provided the kernel $k(\mathbf{x}, \mathbf{x}_j)$ is of the form $k(\mathbf{x}, \mathbf{x}_j) = f(d(\mathbf{x}, \mathbf{x}_j))$ with Euclidean distance d such as the Gaussian kernel. Then, we can substitute d by the Fisher distance or its approximation d_T, respectively, using an estimation of the Fisher information $\mathbf{J}(\mathbf{x})$ based on a finite training sample $\mathbf{x}_i$. This directly leads to a parameterized dimensionality reduction mapping where the given class labels are taken into account. Note that we only need class labels for the given training set since the Fisher information $\mathbf{J}(\mathbf{x})$ can be evaluated based on the distances of $\mathbf{x}$ to the training set only. As before, we use an initial training set to train the parameters α_j of discriminative kernel-t-SNE, whereby the training set is obtained mapping the training points $\mathbf{x}_i$ to low dimensional data points based on their Fisher information.

Thus, to visualize N data points, we first sample m points, where typically $m \ll N$, and compute the Fisher distances between them. These distances are then used to obtain an explicit mapping via kernel-t-SNE, which in turn allows to map the remaining points. In doing so, we additionally need to compute the Fisher distances between the m training points and the $N - m$ new points. The described procedure leads to a complexity of $O(m^3 + mN)$, which is an improvement in comparison to the complexity $O(N^2)$ of t-SNE, as well as a reduction in needed computations of the expensive Fisher distances, again $O(N^2)$.

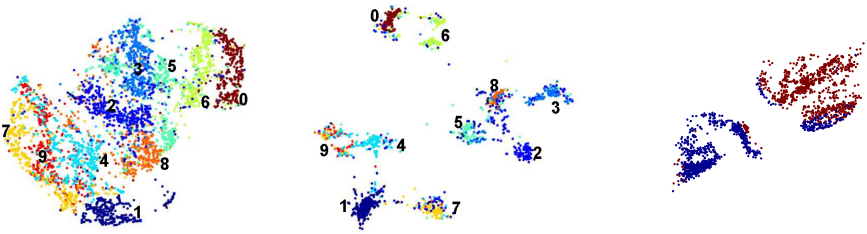
5 Experiments

For the experiments we use two data sets, both representing ten handwritten digits. The USPS data set, which consists of 11.000 points with 256 dimensions [9] and the MNIST data set, which consists of 60,000 points with 768 dimensions [12]. Each data set is projected to 30 dimensions using PCA. For training and the representation of the kernel mapping we take 10% and 2% of the data for USPS and MNIST respectively. The remaining data is treated as the test set. For the estimation of the Fisher information 1% of the data are used for both data sets. Quality evaluation is done by repeated sampling over 100 points as detailed above. We compare kernel t-SNE, discriminative kernel t-SNE and supervised MDS [22]. Due to the limitation of the available code [1] to only two classes, we restrict the setting to digits 5 and 8 for SMDS.

[1] `cran.r-project.org`, package **superMDS**

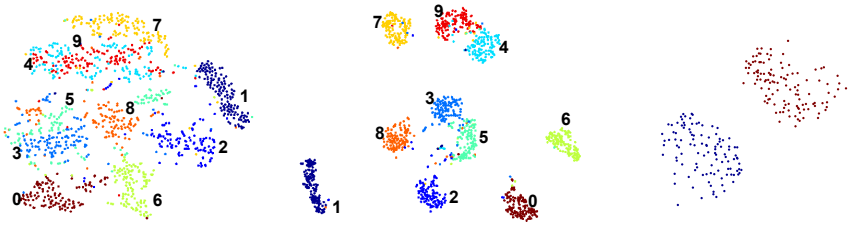


(a) t-SNE for the training set (b) discriminative t-SNE for the training set (c) supervised MDS for the training set

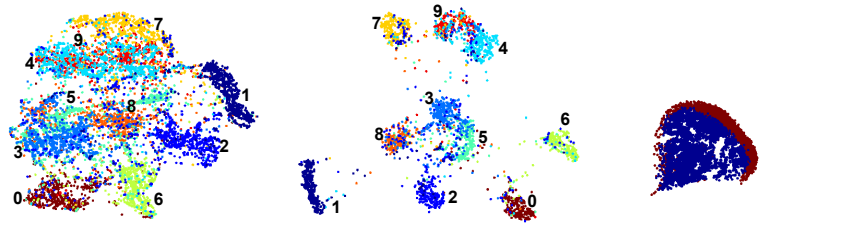


(d) kernel t-SNE mapping for the test set (e) discriminative kernel t-SNE mapping for test set (f) supervised MDS for the test set

Fig. 3. Comparison of the visualization of USPS for kernel t-SNE, discriminative kernel t-SNE and supervised MDS on the test and training sets

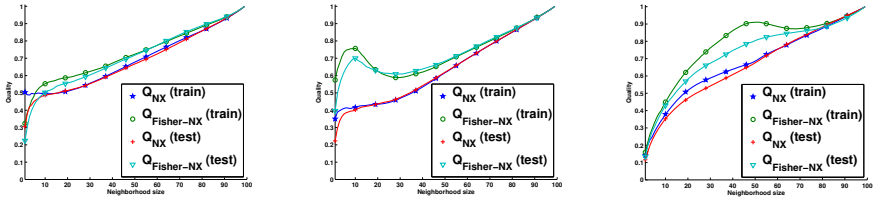


(a) t-SNE for the training set (b) discriminative t-SNE for the training set (c) supervised MDS for the training set



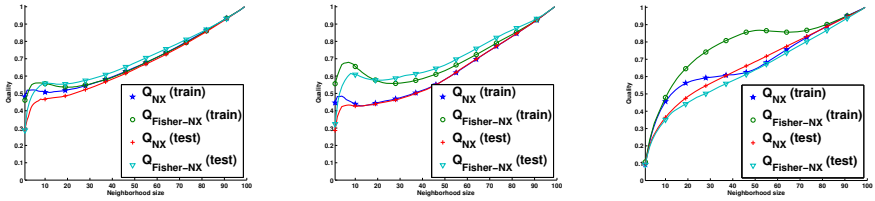
(d) kernel t-SNE mapping for the test set (e) discriminative kernel t-SNE mapping for test set (f) supervised MDS for the test set

Fig. 4. Comparison of the visualization of MNIST for kernel t-SNE, discriminative kernel t-SNE and supervised MDS on the test and training sets



(a) Quality of kernel t-SNE (b) Quality of discriminative kernel t-SNE (c) Quality of SMDS

Fig. 5. Quality evaluation of the USPS map using the (Fisher-)quality measure



(a) Quality of kernel t-SNE (b) Quality of discriminative kernel t-SNE (c) Quality of SMDS

Fig. 6. Quality evaluation of the MNIST map using the (Fisher-)quality measure

The visualizations of USPS and MNIST are shown in Fig. 3 and Fig. 4 respectively, with the corresponding qualities depicted in Fig. 5 and Fig. 6.

Unlike SMDS, both, kernel t-SNE and discriminative kernel t-SNE display a good generalization towards novel data, as can be observed when inspecting the visualizations or quality measures of the training and test sets. This is probably because SMDS is lacking of an explicit mapping function and needs to optimize a cost function for the new data points. Furthermore, while the cluster structure is not yet apparent for kernel t-SNE due to the small fraction of data used for training, it is clearly visible for discriminative kernel t-SNE due to the additional explicit label information. This difference becomes clearly apparent when inspecting the Fisher quality. Similarly, SMDS is able to utilize the auxiliary information to produce an improved mapping, although on a different scale. Considering the Fisher quality, discriminative kernel t-SNE preserves the local structure, whereas SMDS preserves the global structure. This is because t-SNE takes into account the local neighborhoods, but ignores the global relations. On contrary, SMDS is a combination of a linear mapping, which retains the global structure, and a mapping, which separates the classes, thus tearing apart the local neighborhoods.

6 Discussion

We have addressed the topic of discriminative data visualization as a technique to intuitively solve ambiguities of the ill-posed problem of dimensionality reduction by auxiliary information. Based on the Fisher information, we have proposed a formal evaluation measure to evaluate discriminative dimensionality reduction and we have demonstrated that this measure offers an evaluation which focuses on topographic characteristics of the data as concern the labeling rather than a simple classification error only. In addition, we have proposed a framework of how popular t-SNE can be extended to a discriminative dimensionality reduction mapping which provides good results when trained on a small subset of the given data only. The obtained results also display the perspective to speed up visualization techniques based on explicit mapping functions. Methods such as t-SNE possess squared complexity such that they are not suitable for the display of large data sets. Thus, an explicit mapping offers the possibility to train the mapping on a small subsample only, providing extensions to the full data set by means of the mapping function. However, depending on the situation it can happen that a small subset of the data does not contain enough information to infer a good t-SNE mapping e.g. because cluster structures are not yet clearly visible in this subset only. We have demonstrated the potential of auxiliary information in form of class labels to enhance the generalization ability in such settings.

Funding. This work has been supported by the German Research Foundation (DFG) under grant number HA2719/7-1 and by the Cluster of Excellence 277 Cognitive Interaction Technology funded in the framework of the German Excellence Initiative.

References

1. Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural Computation 12, 2385–2404 (2000)
2. Bunte, K., Biehl, M., Hammer, B.: A general framework for dimensionality reducing data visualization mapping. Neural Computation 24(3), 771–804 (2012)
3. Bunte, K., Schneider, P., Hammer, B., Schleif, F.-M., Villmann, T., Biehl, M.: Limited rank matrix learning, discriminative dimension reduction and visualization. Neural Networks 26, 159–173 (2012)
4. Cohn, D.: Informed projections. In: Becker, S., Thrun, S., Obermayer, K. (eds.) NIPS, pp. 849–856. MIT Press (2003)
5. Geng, X., Zhan, D.-C., Zhou, Z.-H.: Supervised nonlinear dimensionality reduction for visualization and classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B 35(6), 1098–1107 (2005)
6. Gisbrecht, A., Lueks, W., Mokbel, B., Hammer, B.: Out-of-sample kernel extensions for nonparametric dimensionality reduction. In: ESANN 2012, pp. 531–536 (2012)
7. Gisbrecht, A., Mokbel, B., Hammer, B.: Linear basis-function t-sne for fast nonlinear dimensionality reduction. In: IJCNN (2013)

8. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems 17, pp. 513–520. MIT Press (2004)
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York (2001)
10. Iwata, T., Saito, K., Ueda, N., Stromsten, S., Griffiths, T.L., Tenenbaum, J.B.: Parametric embedding for class visualization. Neural Computation 19(9), 2536–2556 (2007)
11. Kaski, S., Sinkkonen, J., Peltonen, J.: Bankruptcy analysis with self-organizing maps in learning metrics. IEEE Transactions on Neural Networks 12, 936–947 (2001)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324 (1998)
13. Lee, J., Verleysen, M.: Quality assessment of dimensionality reduction: Rank-based criteria. Neurocomputing 72(7-9), 1431–1443 (2009)
14. Lee, J.A., Verleysen, M.: Nonlinear dimensionality redcution. Springer (2007)
15. Lee, J.A., Verleysen, M.: Scale-independent quality criteria for dimensionality reduction. Pattern Recognition Letters 31, 2248–2257 (2010)
16. Ma, B., Qu, H., Wong, H.: Kernel clustering-based discriminant analysis. Pattern Recognition 40(1), 324–327 (2007)
17. Memisevic, R., Hinton, G.: Multiple relational embedding. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 17, pp. 913–920. MIT Press, Cambridge (2005)
18. Peltonen, J., Klami, A., Kaski, S.: Improved learning of riemannian metrics for exploratory analysis. Neural Networks 17, 1087–1100 (2004)
19. van der Maaten, L.J.P., Hinton, G.E.: Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research 9, 2579–2605 (2008)
20. Venna, J.: Dimensionality reduction for Visual Exploration of Similarity Structures. PhD thesis, Helsinki University of Technology, Espoo, Finland (2007)
21. Venna, J., Peltonen, J., Nybo, K., Aidos, H., Kaski, S.: Information retrieval perspective to nonlinear dimensionality reduction for data visualization. Journal of Machine Learning Research 11, 451–490 (2010)
22. Witten, D.M., Tibshirani, R.: Supervised multidimensional scaling for visualization, classification, and bipartite ranking. Computational Statistics and Data Analysis 55, 789–801 (2011)

Finding Interesting Contexts for Explaining Deviations in Bus Trip Duration Using Distribution Rules

Alípio M. Jorge[1], João Mendes-Moreira[2], Jorge Freire de Sousa[3], Carlos Soares[4], and Paulo J. Azevedo[5]

[1] LIAAD-INESC TEC DCC-FCUP, Universidade do Porto
[2] LIAAD-INESC TEC, DEI-FEUP, Universidade do Porto
[3] UGEI-INESC TEC, DEGI-FEUP, Universidade do Porto
[4] INESC TEC, FEP, Universidade do Porto
[5] Haslab-INESC TEC, Universidade do Minho

Abstract. In this paper we study the deviation of bus trip duration and its causes. Deviations are obtained by comparing scheduled times against actual trip duration and are either delays or early arrivals. We use distribution rules, a kind of association rules that may have continuous distributions on the consequent. Distribution rules allow the systematic identification of particular conditions, which we call contexts, under which the distribution of trip time deviations differs significantly from the overall deviation distribution. After identifying specific causes of delay the bus company operational managers can make adjustments to the timetables increasing punctuality without disrupting the service.

Keywords: Bus trip duration deviations, distribution rules.

1 Introduction

In the last two/three decades, passenger transport companies have made important investments in information systems, such as Automatic Vehicle Location (AVL), automatic passenger counting, automated ticketing and payment, multimodal traveler information systems, operational planning and control software and data warehouse technology, among others. As a consequence of this effort in Advanced Public Transportation Systems, passenger transport companies have been able to collect massive amounts of data. However, as in other areas of activity, the data collected are not being used as much as they could be in supporting public transport companies to accomplish their mission, despite the potential of both data and existing knowledge.

The planning of public transport companies is a complex task. It has as major goal: the achievement of the adequate offer of trips using the resources at a minimal cost. The two main resources are drivers and buses. The uncertainty of trip duration [14] must be taken into account in the definition of schedules, in order to obtain an adequate offer of trips at minimal costs. The problem is that

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 139–149, 2012.
© Springer-Verlag Berlin Heidelberg 2012

it is not known how much uncertainty should be assumed because it is difficult to evaluate the trade-off between the operational costs (due to the amount of resources used) and client satisfaction. Additionally, the metrics used to measure client satisfaction vary according to the type of routes. For example, in highly frequent urban routes with a five minutes headway (where headway is the time gap between two consecutive vehicles) client satisfaction assessment is based on the stability of the headway. On low frequent suburban routes (e.g. 60 minutes headway) a metric based on the deviation from departure time is preferred.

In this paper we present a study on how to take advantage of stored AVL data in order to promote adjustments to existing timetables. For that we use the data mining technique of distribution rules [9]. We intend to detect systematic deviations between the actual and the scheduled trip duration and identify the causes of such deviations. Such tool can be integrated in a decision support tool for timetable adjustments [13].

We start by describing distribution rules, the data mining technique that we use to study trip duration deviation. We then present the datasets used and their preparation. Next, we provide details and results of the data mining step. Results are discussed both from a data mining and an operational point of view.

2 Distribution Rules

Distribution rules (DR) [9] are constructs similar to association rules (AR), but having a distribution of an attribute of interest A on the consequent. Whereas the antecedent of a DR is similar to the one of an AR, its consequent is a distribution of A under the conditions stated in the antecedent. In the DR setting, all the antecedents Ant which correspond to an interesting distribution $D_{A|Ant}$ (read as "the distribution of A given Ant") are found. In this case, interesting means that the distribution of A under Ant is significantly different from the distribution of A without any constraints (*a priori*).

Definition 1. *A distribution rule (DR) is a rule of the form* $Ant \rightarrow A = D_{A|Ant}$*, where Ant is a set of items as in a classical association rule, A is a property of interest (the target attribute), and* $D_{A|Ant}$ *is an empirical distribution of A for the cases where Ant is observed. This attribute A can be numerical or categorical.* $D_{A|Ant}$ *is a set of pairs* $A_j/freq(A_j)$ *where* A_j *is one particular value of A occurring in the sample and* $freq(A_j)$ *is the frequency of* A_j *for the cases where Ant is observed.*

In Figure 1 we can see one distribution rule derived from "Auto MPG", a data set with descriptions of cars where the property of interest (P.O.I.) is their fuel consumption in miles per galon (MPG) [6]. The antecedent is shown above the chart. The darker curve shows the density of $MPG|Ant$, the grey curve shows the density of MPG overall. We can also see some measures characterizing the rule: KS-interest, a measure of interest of the rule given by $1 - p_{KS}$, where p_{KS} is the *p-value* of the Kolmogorov Smirnov test; the support (Sup) of the

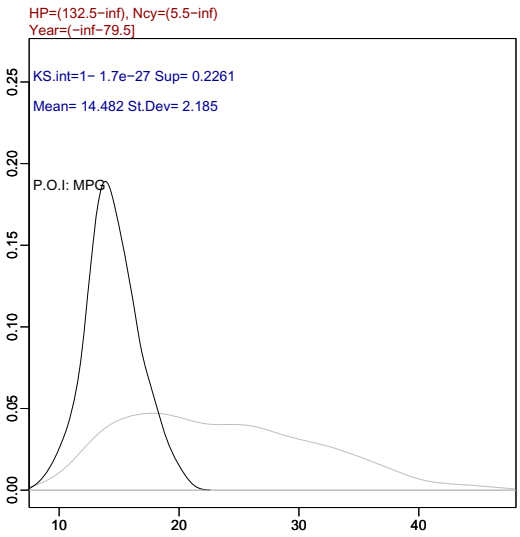


Fig. 1. Distribution rule for the "Auto MPG" data set

antecedent; the mean of $MPG|Ant$ and its standard deviation. The represented densities are estimated using kernel density estimation [8].

Given a dataset, the task of distribution rule discovery consists in finding all the DR $Ant \rightarrow A = D_{A|Ant}$, where Ant has a support above a determined minimum σ_{min} and $D_{A|Ant}$ is statistically significantly different (w.r.t. a pre-defined level of significance) from the default distribution $D_{A|\emptyset}$. The default distribution is the one obtained with all the values of A for the whole dataset or a distribution obtained from a holdout data set. To compare the distributions we use *Kolmogorov-Smirnov* (KS) [4], a statistical goodness of fit test. The value of the KS statistic is calculated by maximizing $|F_s(x) - F(x)|$, where $F(x)$ is the empirical cumulative distribution function for the whole domain of A and $F_s(x)$ is the cumulative distribution function for the cases covered by Ant.

3 Trip Time Deviation

The ultimate aim of this work is to improve the quality of urban bus service. One important aspect is adjusting schedules to operational conditions and vice versa. For that, we need to know what are the factors, or combination of factors, that are associated with deviations in trip duration. That can be done using descriptive data mining techniques [7]. The extracted patterns can then be used to help operational managers taking measures. In this paper we use the case of urban line 205 of STCP, the main Oporto bus operator.

Our property of interest is "trip duration deviation", named *Deviation* in this paper. This is defined as the difference, in seconds, between the time the bus actually takes from departure point to destination and the published scheduled

duration. Its a priori distribution corresponds to the whole population. Our operational problem of finding relevant factors of deviation will be translated, as a data mining problem, into discovering contexts that are associated with statistically significant changes in the distribution of the variable *Deviation*. Below, we formally define the notion of context.

Definition 2. *A context is a combination of conditions that can be observed at a given moment and influences operation. In logical terms, and in this work, a context is a conjunction of logical literals* $Cond_1 \wedge \ldots \wedge Cond_n$.

The descriptive data mining technique of distribution rule discovery presented in section 2 is able to find relevant contexts, given one property of interest. This discovery problem is also related to the problem of subgroup discovery [10][11]. What we do is: given a dataset with the description of trips (operational conditions and deviation), we obtain all interesting distribution rules. Each rule covers a subgroup of trips and associates one particular context with one particular distribution of the variable Deviation which is sufficiently different from its a priori distribution. To discover the rules we use the program Caren [1].

4 Data Preparation

From the data collected by the company, we analyse two different periods. The first period spans from January to September 2007 (dataset1 with 14169 records) and the second from November 2007 to March 2008 (dataset2 with 9203 records). The attributes of the datasets are described in Table 1. Each dataset line describes one bus trip from departure to destination. Along with static descriptors we have the actual time taken in that journey. From the published schedule we obtain the scheduled duration. The difference between actual duration and scheduled duration gives us the deviation.

We start by analyzing trip duration. The two boxplots in Figure 2 show that we have a very high number of outliers corresponding to extreme durations, mostly for the first period. However, our experience indicates that most of these outliers are caused by operational errors. In particular, bus drivers must press a button at the end of each trip so that arrival time is recorded. However, this operation may fail for different reasons. As a consequence, the recorded arrival time will be the next arrival time (or the one after that) which multiplies trip duration. Simple observation suggests a cut above 6000, for dataset1, and a cut above 5000 for dataset2. With this data cleaning operation we reduce the possibility of artificial deviations.

The attribute StartTime, originally in seconds elapsed on that day, has been discretized to 24 values ($\{0, 1, \ldots, 23\}$, where the value of k represents the interval between hour k and $k + 1$. This hourly discretization aims to capture the effects of different times of the day on trip time deviation.

Table 1. Attributes of the datasets, including derived attributes

Attribute	Description
Date	Date of trip
DepartureS	Departure time (in seconds)
Departure	Departure time discretized in 24 values (by hour)
Model	Vehicle model
Driver	Driver number. Zero means driver shift
DayOfYear	1st January is 1, 1st Feb is 32, etc.
WeekDay	From Monday to Sunday
DayType	Normal days, holidays, etc.
Duration	Duration of trip in seconds
SchDeparture	Scheduled time of departure
SchArrival	Scheduled time of arrival
SchDuration	Scheduled trip duration (seconds)
Deviation	Duration-SchDuration (seconds)

5 Discovering Interesting Contexts

We now look at the variable *Deviation* and use distribution rules to see how its distribution varies with the context. Our aim is to find combinations of conditions (contexts) that are associated with tendencies for positive and negative deviations, respectively delays and early arrivals.

From dataset1, using a level of significance of 0.05 on the Kolmogorov-Smirnov test and a minimum support of 2%, we obtained 30 rules. To avoid the generation of redundant rules we have also used a statistical significance filter. A rule $Ant_l \rightarrow D_l$ is added to the set of discovered rules only if there is no immediately simpler rule $Ant_s \rightarrow D_s$, such that Ant_s has exactly one item less than Ant_l and the distributions D_l and D_s are not significantly different. This latter test is performed using the same level of significance used for rule generation. We will examine some of the rules in more detail. We will use dataset2 to check the findings in dataset1. From dataset2, under the same conditions, we have obtained 25 rules.

5.1 Delays

In Fig. 3 we can see four of the top rules wrt positive deviation from dataset1. Positive deviation is associated with situations of frequent delays.

An emergent context is departure time between 2 and 3 P.M (Departure=14). Other contexts involving departure time around and immediately after lunch time also appear. This is an interesting information which implies that perhaps more care should be taken on that period, either by changing behavior (in order to observe defined schedules) or by adjusting schedules to operational conditions. The second rule shown indicates that a top cause of delay is the combination of driver's shifting (Driver=0 indicates change of driver) with the use of a particular

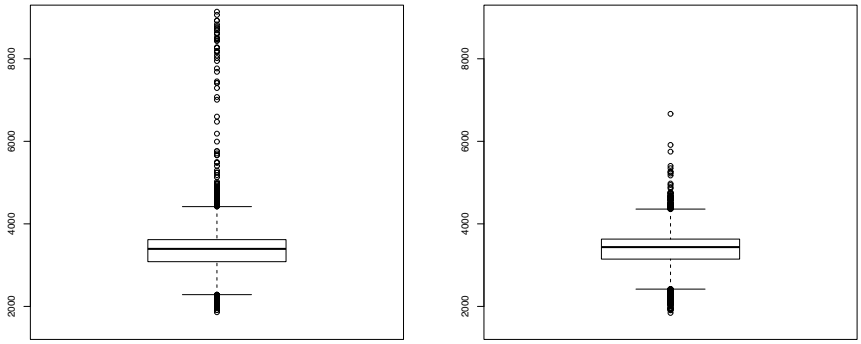


Fig. 2. Boxplots for trip duration in dataset1 and dataset2, respectively

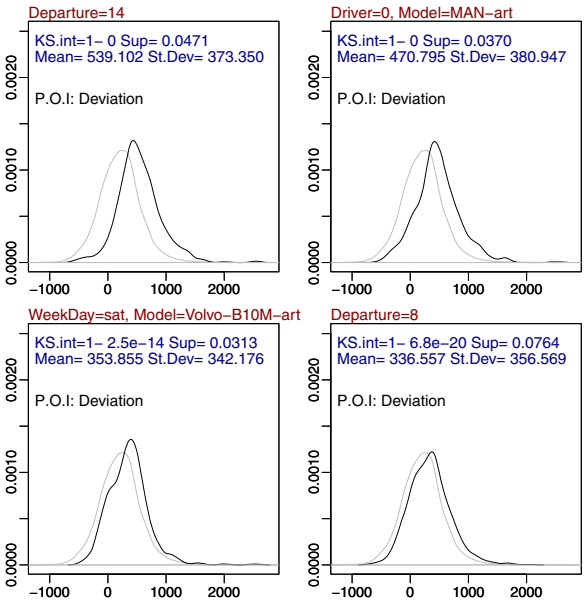


Fig. 3. Four top distribution rules for dataset1. The lighter line represents the a priori distribution of *Deviation*. The darker line represents the distribution in the context described by the condition or conditions above the box.

model of articulated vehicles. This draws the attention for the possibility of inefficiency in the process of transferring one vehicle from one driver to another. On the other hand, articulated vehicles may have difficulties in complying to the schedule. This may be caused by difficult manouvres in face of abusive street parking or narrow streets. The third context shown is Saturdays with another model of articulated vehicles. The fact that Saturdays are associated with delays

must also be studied by operational management. We can see that in these three contexts the distribution of trip duration deviation is clearly shifted to the right, with means over 300 seconds (5 minutes). Another interesting context for positive deviation is related to departure time between 8 and 9 A.M. These top rules cover from 3% to 7.6% of the trips.

The rules obtained from dataset2 confirm the importance of departure time. Early afternoon hours have a tendency for positive deviations. These are top rules here too. Driver change is also a cause for delay. The association between articulated vehicles and positive deviations is not observed. However, there is an association between the non-articulated model "MAN-3s" and negative deviations. A simple explanation for not having the model "MAN-art" as an interesting context here is that in this case this model represents more than 90% of the vehicles, whereas in dataset1 it is about 56%. Thus, the context "MAN-art" does not have a significantly different distribution of the whole set of trips.

5.2 Early Arrivals

From the same set of rules (dataset1) we obtain contexts that are related to negative deviations, in particular deviation distributions that are to the left of the a priori distribution (Fig. 4). Negative deviation is associated with early arrivals. In this case we observe that buses leaving after 7 P.M and before 8 tend to arrive earlier than in general. Moreover, there is a large majority of cases when these buses arrive before scheduled time. This is clear by observing the

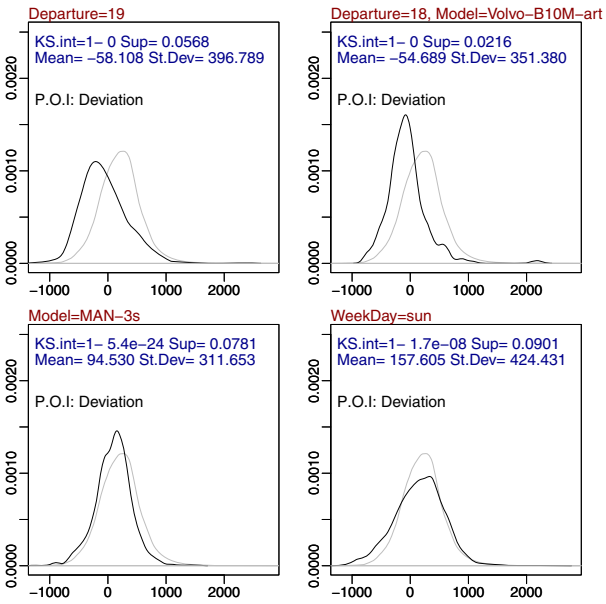


Fig. 4. Four top distribution rules with negative deviation for dataset1

resulting distribution for that context. Buses leaving beween 5 and 7 P.M. also have this tendency, but not as much. The use of vehicle model "Man-3s" is also associated with decreased delays. It is interesting to observe this association with this particular model, which is not articulated.

Another interesting context is related to departure time between 10 and 11 P.M. In this case there is a very good behavior with a high concentration around scheduled times (not shown in the Figure), i.e., practically no deviation is observed. This concentration effect is not surprising for this time of the day. The same behavior would also be expected for Sundays and holidays. Instead, however, Sundays appear as a day of many early arrivals, which can be very inconvenient for passengers. Similarly to Sundays, holidays have a large tendency for early arrivals. In fact, we can observe that early arrivals are also a tendency for departure between 8 and 9 P.M. but not for departures after 9 P.M.

With the rules obtained from dataset2 we confirm that early arrivals are associated with the early evening period (starting at 5 P.M. but mostly between 7 and 9 P.M.). As we have said above we can associate a non-articulated model with a lower tendency to delay. Regarding days of week and type of day, we also have Sunday as an interesting context.

6 Discussion

The use of distribution rules shows some advantages with respect to simple regression. With DRs we are able to find interesting contexts, or subgroups of trips, and look at the distribution of values instead of only having one or two parameters (typically mean and standard deviation). The discovery strategy based on the Kolmogorov-Smirnov test does actually look for interesting distributions. These may be shifted to the right, indicating a delay tendency, shifted to the left, indicating an early arrival tendency. By visually observing the discovered interesting distributions, we find particular situations of interest, as it was the case of Sundays and departures after 10 P.M. Moreover, we can distinguish between these two contexts which would, at a first sight, be similar. This exploratory visual inspection of relevant contexts provides the operational manager with hints for service improvement. The information thus obtained can be used to ameliorate the schedules. Next, we discuss some situations based on the results presented in Fig. 3 and 4.

In low frequency routes the planning should encourage timely departure times. For this type of routes, slack times (the time gap between the end of one trip and the beginning of the next for the same vehicle) should be used in order to accommodate trip time variation. The amount of variance assumed by the schedule should be a given percentage of the sample trip time distribution given by DR. It defines the trade-off between operational costs and client satisfaction. This choice is done by the planner. We use the right bottom plot of Fig. 4 to exemplify. The headway of this route on Sundays goes from 20 to 30 minutes (low frequency). We observe that a meaningful percentage of the trips have deviations lower than 0. This means that there is a high probability that the

bus will pass, at least in the last stops of the route, before the scheduled passing time. Considering the low frequency of this route on Sundays, this may imply that some clients lose the bus causing dissatisfaction. It would be advisable to reduce the scheduled trip time in this case or to communicate with drivers.

Situations like the one presented in the right bottom plot of Fig. 3 should be analyzed according to the slack time used. That is, if the slack time does not cover the majority of cases where the trip exceeded the scheduled trip duration, the slack time should be increased. On the other hand, if the planned slack time covers all the cases, it could be reduced in order to avoid waste of resources. Alternatively, bus driver scheduling could be optimized by including shifts at the end of trips that are expected to have higher slacks, in order to maximize driving time. The amount of delays covered by the slack time is, once more, something that must be defined by the planner.

There are other situations that should also be taken into account, such as the identification of problems with a particular vehicle model that should be considered during vehicle rostering; or measures to reduce delays due to the occurrence of driver's shifting during the trip.

6.1 Using DR to Determine Slack Time

The obtained distribution rules can be used to adjust slack times (ST) in order to optimize resources. Given a context Con we find the rule $Ant \rightarrow D$ whose conditions best apply to it. Then, we can determine the minimal duration t_{min} of ST that fails to cover at most a given percentage p_{uncov} of the trips. In this case, covering means that the slack time is sufficient to avoid delay. The value of t_{min} is the solution of the equation below, where fd_{Ctxt} is the density function given by the DR which applies to the context $Ctxt$.

$$t \text{ such as } p_{uncov} = \int_{-\infty}^{t} fd_{Ctxt}dx \tag{1}$$

If the value of t_{min} is negative then there is space to reduce trip time for the particular context considered. The rule that applies to a context is the one with the largest antecedent made true by the context. Ties are resolved by preferring rules with higher support.

6.2 Related Work

This paper discusses a tool that can be used to: (1) detect deviations between actual and scheduled trip duration and its causes; (2) support the definition of scheduled trip durations, slack times or frequencies.

As far as we know, the only existing study that addresses the first objective [5] uses classification based on association [12]. The authors discretize deviation into four classes. All discovered association rules have discretized deviation as consequent. This approach does not give useful information in order to address the second objective, as DR do (as described in Sect. 5).

Other works address only the second objective [3,15]. Using an economic perspective, [3] defines as objective function, the cost expressed in terms of scheduled trip durations, lateness and earliness unit costs. Using this approach it is possible to define the optimal scheduled trip durations and slack times (time between trips) for given ratios between the unit cost of scheduled trip durations and both the lateness and earliness unit costs. Another contribution of this work is the inclusion in the model of the effect of relaxation when the slack time is larger. I.e., it is known that when the schedule is tight, the actual trip duration is shorter than when it is large. Carey calls it the behavioral response. What Carey shows is that the timetable definition should be neither too tight, to avoid delays in departures, nor too large, to avoid behavioral inefficiency.

Under certain conditions, slack times can be optimized [15]. Using this approach, the shorter the slack time is, the shorter the scheduled headway is. The function to optimize defines the passengers' expected waiting time in terms of the scheduled headway and the variance of the delay. By using the function defined in [2] for the passengers' arrival at the bus stops, it is possible to adapt the solution to problems with large headways. These are analytic simplified general models. We can use distribution rules to model and estimate waiting time based on collected data. Moreover, this estimation can be done with respect to well defined emerging contexts instead of all the trips.

Our approach differs from these two [3,15] by leaving to the planner the decision on how to deal with the trade-off between operational costs and clients' satisfaction. Moreover, it is not made any assumption about the distribution of the data. The only assumption is about the sample. It is assumed that the sample is representative of the population.

7 Conclusion and Future Work

In this paper we have exploited distribution rule discovery to study deviations in bus trip duration. We have used real data collected from an urban bus company. Distribution rules allowed us to discover operational contexts that are, with high significance, statistically associated with relevant deviations in trip duration. Discovered rules can also be used to support the adjustment of timetables and schedules (e.g. redefining slack times).

Our next steps are to integrate these findings with bus operation in an ongoing collaboration with the bus company. Some of the discovered relevant contexts are already known by operational managers. Some others may yield suggestions that are not practical, due to the complexity of the processes. Some may even go against legally established principles. However, this work has shown that schedules must be brought closer to operational conditions. Moreover we can identify paths for addressing the problem of trip duration deviation. In particular, we will incorporate this tool in a decision support system for timetable adjustments [13]. The application of this approach to the tens of lines of the bus company can have a great impact in the reduction of inefficiency and the increase of client satisfaction.

Acknowledgements. This work is part-funded by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness), by the Portuguese Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701.

References

1. Azevedo, P.J.: CAREN - class project association rule engine (2008), `http://www.di.uminho.pt/~pja/class/caren.html`
2. Bowman, L.A., Turnquist, M.A.: Service frequency, schedule reliability and passenger wait times at transit stops. Transportation Research Part A 15, 465–471 (1981)
3. Carey, M.: Optimizing scheduled times, allowing for behavioural response. Transportation Research Part B 32(5), 329–342 (1998)
4. Conover, W.J.: Practical Nonparametric Statistics, 3rd edn. John Wiley & Sons, New York (1999)
5. Duarte, E., Mendes-Moreira, J., Belo, O.: Exploração de técnicas de classificação associativa no planeamento de horários de transportes públicos (exploitation of associative classification techniques in the planning the schedules of public transports). In: 9 Conferência da Associação Portuguesa de Sistemas de Informação, Viseu - Portugal (2009)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
7. Hand, D., Manila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
8. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (August 2001)
9. Jorge, A.M., Azevedo, P.J., Pereira, F.: Distribution Rules with Numeric Attributes of Interest. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 247–258. Springer, Heidelberg (2006)
10. Kavšek, B., Lavrač, N., Jovanoski, V.: APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery. In: Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 230–241. Springer, Heidelberg (2003)
11. Klősgen, W.: Explora: A multipattern and multistrategy discovery assistant. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (1996)
12. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD 1998: Proceedings of the fourth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 80–86. ACM Press, New York (1998)
13. Mendes-Moreira, J., Duarte, E., Belo, O.: A decision support system for timetable adjustments. In: 13th EURO Working Group on Transportation Meeting (EWGT 2009), Padua - Italy (2009)
14. Mendes-Moreira, J., Jorge, A., de Sousa, J.F., Soares, C.: Comparing state-of-the-art regression methods for long term travel time prediction. Intelligent Data Analysis 16(3), 427–449 (2012)
15. Zhao, J., Dessouky, M., Bukkapatnam, S.: Optimal slack time for schedule-based transit operations. Transportation Science 40(4), 529–539 (2006)

Curve Fitting for Short Time Series Data from High Throughput Experiments with Correction for Biological Variation

Frank Klawonn[1,2], Nada Abidi[2], Evelin Berger[2], and Lothar Jänsch[2]

[1] Department of Computer Science
Ostfalia University of Applied Sciences
Salzdahlumer Str. 46/48, D-38302 Wolfenbuettel, Germany
[2] Cellular Proteomics
Helmholtz Centre for Infection Research
Inhoffenstr. 7, D-38124 Braunschweig, Germany

Abstract. Modern high-throughput technologies like microarray, mass spectrometry or next generation sequencing enable biologists to measure cell products like metabolites, peptides, proteins or mRNA. With the advance of the technologies there are more and more experiments that do not only compare the cell products under two or more specific conditions, but also track them over time. These experiments usually yield short time series for a large number of cell products, but with only a few replicates. The noise in the measurements, but also the often strong biological variation of the replicates makes a coherent analysis of such data difficult. In this paper, we focus on methods to correct measurement errors or deviations caused by biological variation in terms of a time shift, different reaction speed and different reaction intensity for replicates. We propose a regression model that can estimate corresponding parameters that can be used to correct the data and to obtain better results in the further analysis.

1 Introduction

Modern biology is interested in better understanding mechanisms within cells. For this purpose, products of cells like metabolites, peptides, proteins or mRNA are measured and compared under different conditions, for instance healthy cells vs. infected cells. In recent years, there is also a stronger focus on time series data to analyse the behaviour of the cell products over time. Since the experiments yielding the data are usually quite expensive and time consuming, often measurements from only a few time points are available. Such experiments usually yield regulation or expression values – the abundance or absence of a cell product compared to the initial measurement or a control experiment – for a large number of cell products, but with only a few replicates.

There are various specific challenges in the analysis of expression time series data. A categorisation of the challenges and an overview on various approaches in the specific context of gene expression data can be found in [1]. In this paper, we focus on what is called the *data analysis level* of analysing gene expression time series data.

One of the main problems in the analysis of such data for deriving models of the behaviour of the cell products is the high variance of the data due to noisy measurements and differences in the metabolism of the individual biological organisms.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 150–160, 2012.
© Springer-Verlag Berlin Heidelberg 2012

In order to detect cell products with common behaviour over time, simple models need to be fitted to the data taking into account that apart from the noisy measurements there might also be a phase shift as well as variations of the speed and the reaction intensity of the biological process. This leads to a non-linear fitting problem for which we propose a solution based on methods from linear regression and alternating optimisation [2].

The problem of time shift and stretch has also been addressed in [3] and applied to identify differentially expressed genes in time series expression data [4]. However, our approach differs significantly from the ideas presented in [3,4] where the focus is put on the alignment of pairs of time series. We assume that the alignment of time series must be consistent for the whole replicate.

Furthermore, we do not restrict our approach to gene expression data, but explicitly use it for proteomics data. In contrast to gene expression data that are usually based on microarrays, proteomics relies on mass spectrometry which tends to produce a much higher number of missing values compared to microarrays.

We first give a formal definition of the problem we consider in terms of a regression problem in Section 2 and describe our algorithm to estimate the parameters of the regression problem in Section 3. Section 4 discusses practical aspects including the requirements of our approach and how to apply it correctly. An evaluation of our method based on artificial and real data is provided in Section 5 where we also provide a visualisation of the results, an essential part of the analysis of such data [5].

2 Problem Formalisation

Table 1 shows the principle structure of the data, we have to deal with. We consider N cell products (genes, proteins, peptides or metabolites) that are measured at T different time points $s_1, \ldots, s_T$. The intervals between time points can vary. We also have R replicates. In the ideal case, the replicates are (almost) identical, i.e. we would expect

$$x_{n,t}^{(1)} \approx \ldots \approx x_{n,t}^{(R)}$$

for all $n \in \{1, \ldots, N\}$ and all $t \in \{s_1, \ldots, s_T\}$. The table might also contain missing values.

Table 1. General structure of the data

Time point	s_1			...	s_T		
Cell product	Replicate 1	...	Replicate R	...	Replicate 1	...	Replicate R
i_1	$x_{1,1}^{(1)}$	...	$x_{1,1}^{(R)}$	...	$x_{1,T}^{(1)}$	...	$x_{1,T}^{(R)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_N	$x_{N,1}^{(1)}$	...	$x_{N,1}^{(R)}$	...	$x_{N,T}^{(1)}$	...	$x_{N,T}^{(R)}$

There are two influences that need to be taken into account.

- Noise that comes from the measurement procedure and
- general variations of the biological organism that is investigated.

The noise coming from the measurement procedure and devices is not expected to have any specific bias or systematic structure. Part of the variations in the replicates might be caused by small errors in the experimental setting or differences in the metabolism of the organism under investigation. A (small) shift can occur when the start of the measurement of the organism differs slightly. For instance, although a systematic virus infection in a mouse experiment will try to ensure that all mice get the same dose of the virus and that the infection is effective. However, since this cannot be guaranteed completely, the actual reaction to the infection – for instance due to the effective amount of virus intake – might start earlier or later for some individuals. Apart from this time shift, the speed of the metabolism of organisms can lead to different speeds of the process that is measured. Finally, the strength of the reaction of the organims might vary.

In order to take these aspects into account, we consider the following model. We assume that the behaviour of the measured values of a single cell product n can be described by some parametric function $f_n(t; \mathbf{c}_n)$. $\mathbf{c}_n$ denotes a parameter vector that determines the function. For reasons of simplicity, we usually choose a low degree polynomial for $f_n(t; \mathbf{c}_n)$. Of course, other functions like regression splines or other more complex parametric functions as in [6] could also be used. This would, however, make sense only, when there are sufficiently many time points.

We assume that for each replicate there might be a time shift, a different speed for the process and a different strength of the reaction, i.e. different heights of the amplitudes. Therefore, we would expect $x_{n,t}^{(r)} \approx h_r \cdot f_n(a_r t + b_r; \mathbf{c}_n)$ where

- b_r compensates the time shift,
- a_r is for the correction of the different speeds of the metabolisms and
- h_r accounts for the variation in strength of the reaction.

In order to determine the parameters a_r, b_r, h_r and the parameter vectors $\mathbf{c}_n$, we minimise the following error function.

$$E = \sum_{r=1}^{R} \sum_{n=1}^{N} \sum_{t=1}^{T} \delta \left(h_r \cdot f_n(a_r s_t + b_r; \mathbf{c}_n) - x_{n,t}^{(r)} \right) \tag{1}$$

where δ is a suitable error measure. The classical least squares approach would use $\delta(e) = e^2$. But robust regression (see for instance [7,8]) that can better cope with outliers can also be used as an alternative to the least squares approach. In addition, the weights calculated within robust regression could be used to detect outliers in terms of a single measurement or a complete cell product.

3 Parameter Estimation

The nonlinear minimisation problem posed by the error function (1) is solved in the following way.

1. We assume replicate $r = 1$ to be the "standard" and do not adjust the parameters a_1, b_1, h_1 and set them to $a_1 = 1$, $b_1 = 0$ and $h_1 = 1$.
2. The parameter sets $\mathbf{c}_n$ and (a_r, b_r, h_r) are adapted alternatingly.
3. We first initialise the parameters (a_r, b_r, h_r) $(r = 2, \ldots, R)$ by $a_r = 1$, $b_r = 0$ and $h_r = 1$. This means that the initialisation starts with no time shift for all replicates ($b_r = 0$), identical speed ($a_r = 1$) and identical strength of the reaction ($h_r = 1$) for each replicate.
4. The parameter vectors $\mathbf{c}_n$ can be updated independently. If the functions f_n are polynomials of degree K

$$f_n(t) \;=\; \sum_{k=0}^{K} c_n^{(k)} t^k,$$

the regression problem for f_n $(n = 1, \ldots, N)$ is

$$E_n \;=\; \sum_{r=1}^{R} \sum_{t=1}^{T} \delta \left(\sum_{k=0}^{K} c_n^{(k)} \cdot \left[h_r \cdot (a_r s_t + b_r)^k \right] - x_{n,t}^{(r)} \right). \tag{2}$$

Since the values a_r, b_r and h_r are assumed to be fixed in this step and the values s_t and $x_{n,t}^{(r)}$ are given anyway, the determination of the coefficients $c_n^{(k)}$ is a standard linear regression problem that can be solved in the usual way, depending on the choice of δ, i.e. by the standard least squares approach for $\delta(e) = e^2$ or by the corresponding algorithm for robust regression for other choices of δ.
5. The parameters a_r, b_r and h_r can be updated independently for each replicate r $(r = 2, \ldots, R)$ when the parameter vectors $\mathbf{c}_n$ are assumed to be fixed. However, here a strategy for nonlinear optimisation must be applied, but each time only for a problem with three variables.

Steps 4. and 5. are repeated alternatingly until convergence is reached, i.e. until the change of the parameters drops below a pre-specified threshold $\varepsilon > 0$ or until a maximum number of iteration steps has been carried out.

In order to avoid overfitting, we only assume K to be the maximum degree of the poloynomial for representimg the behaviour of a cell product. The regression problem in step 4 is also carried out with polynomials of degree lower than K. For cell product, the degree of the polynomial is chosen based on the Akaike information criterion [9].

The above described algorithm has been implemented within the open source statistics software R [10], using standard least squares regression as well as robust regression based on Huber's ρ and Tukey's biweight. The nonlinear optimisation problem for the adaptation step of the parameters a_r, b_r and h_r exploits the R function `optim`. There is definitely potential to find an optimisation strategy that is better tailored to this specific problem.

4 Practical Aspects

An important restriction for our method is that we must assume that the replicates are really replicates over the whole time scale. In some experiments, it is necessary to destroy

or kill the organism to take the measurements at each time point. In such experiments, one would start initially with $R*T$ replicates in remove R replicates at each time point. This, however, means that the replicates measured at time point s_t are not at all related to the replicates measured at time point s_{t+1}. In such a setting, it does not make sense to adjust the shift and the time and intensity scale parameters a_r, b_r and h_r. Therefore, our approach is only applicable when the same replicates are measured over the whole set of time points.

For the estimation of parameters a_r, b_r and h_r, one could in principle use all all measured cell products. However, very often only a small fraction of these cell products is involved in the biological process of interest. Most of the other cell products might do "nothing" or something completely unrelated to the specific conditions or process of interest. For these cell products, it cannot be expected that their behaviour over time can be aligned over the replicates.

Therefore, not all measured cell should be used to estimate the parameters a_r, b_r and h_r, but only those where an observable effect is expected, i.e. those ones that are known to be involved in the process that is stimulated in the experiment. Otherwise, cell products with random fluctuations that are not involved in the process of interest could cause difficulties for the estimation. Note that it is not necessary to use all cell products that are involved in the process of interest for the estimation of the parameters a_r, b_r and h_r. It is sufficient to use some. Then, in a second step, the parameter vectors $\mathbf{c}_n$ of the functions $f_n(t; \mathbf{c}_n)$ for the cell products that have not been considered so far can be estimated assuming the shift and scale parameters a_r, b_r and h_r to be fixed.

One should be aware of the problem of overfitting. When polynomials of degree K are used to represent the time series, then the number of parameters to be adjusted is

$$K \cdot N + 3 \cdot R.$$

The number of values in the table on which the estimation is based is

$$N \cdot T \cdot R$$

minus the number of missing values.

In order to judge whether the estimation of the parameters a_r, b_r and h_r is meaningful at all, we recommend to carry out simulations with random data sets. The measured values in the Data Table 1 are replaced by independent random values, for instance from a standard normal distribution, so that there is no connection between the replicates at all. One can compare the fitting error, when only the coefficients $\mathbf{c}_n$ are used and when in addition the parameters a_r, b_r and h_r are adjusted. The fitting error will always be smaller in the latter case. The comparison of the reduction of the fitting error for such complete random data set gives an indication of an improvement achieved by the additional parameters a_r, b_r and h_r. When the improvement for the real data set is in the same range, then the fitting of these parameters is not meaningful for the data set. This strategy to estimate the improvement achievable by the parameters a_r, b_r and h_r is similar to the idea of using the corrected R^2 measure for regression problems that also takes into account how much the R^2 measure would reduce for a random data set without any linear dependency.

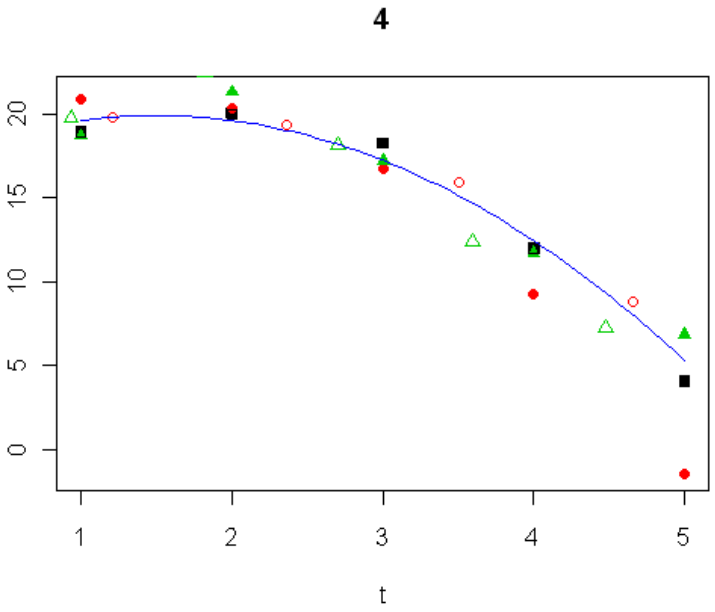


Fig. 1. An example for a result with an artificial data set

5 Examples

In order to visualise the results of our fitting algorithm, we use the following graphical representation. For each item one could plot r separate curves, one for each replicate that incorporates the factor h_r which accounts for the strength of the reaction of the item and also a modification of the time axis based on the estimated parameters a and b. Instead, we plot only one curve for each item and transform the data points correspondingly. The corrected time value for the time point s_t of replicate r is $a_r \cdot s_t + b_r$. The corrected regulation factor of item n in replicate r at time point s_t is $x_{n,t}^{(r)}/h_r$ instead of $x_{n,t}^{(r)}$.

Figure 1 shows the results for one "cell product" of a data set that was generated in the following way. For each of the N (here $N = 12$) cell products, we generate a polynomial of degree K (here $K = 2$) with random coefficients from a normal distribution. We also generate random values for the parameters a_r, b_r and h_r. We have chosen $R = 3$ here. These parameters were chosen to be in accordance with the data from the real experiment that we also consider in this paper, except for the number of time points. The parameters a_r, b_r and h_r are not chosen completely randomly, since we would not expect arbitrary shifts in time and arbitrary scaling of the speed and the intensity of the reaction. Therefore, we generate the values by

$$\begin{aligned} a_r &= 1 + \varepsilon_r^{(a)}, \\ b_r &= \varepsilon_r^{(b)}, \\ h_r &= 1 + \varepsilon_r^{(h)} \end{aligned}$$

where the values $\varepsilon_r^{(a)}, \varepsilon_r^{(b)}$ and $\varepsilon_r^{(h)}$ are random values from a normal distribution with mean zero and a small standard deviation. Then the entry in the data table at time point s_t for cell product n of replicate r is then chosen as

$$x_{n,t}^{(r)} \;=\; h_r \cdot \sum_{k=0}^{K} c_n^{(k)} (a_r s_t + b_r)^k + \varepsilon_{n,t,r} \tag{3}$$

where $\varepsilon_{n,t,r}$ is random (noise) value from a normal distribution with mean zero.

The values for the three replicates in Figure 1 are displayed by squares, triangles and circles. The filled symbols depict the original entries in the data table according to Equation (3). The nonfilled symbols are the corrected values taking the estimated parameters a_r, b_r and h_r into account as described in the beginning of this section. Note that the filled and nonfilled squares are identical, since they correspond to replicate 1 which is taken to be the "standard" and the parameters are fixed to $a_1 = 1$, $b_1 = 1$ and $h_1 = 1$.

One can see in Figure 1 that after the correction based on the estimated parameters a_r, b_r and h_r, the estimated curve fits the data very well. It should be noted that this is just one example out of 12 "cell products". For the other "cell products" the results are very similar. Also the comparison of the randomly generated polynomial coefficients and randomly generated parameters a_r, b_r and h_r with their estimates shows that the original values can be recovered quite well.

We have also repeatedly tested our algorithm on an artificial data set where we have

- generated random coefficients for the polynomials that were used in all repetitions,
- chosen the values for the parameters a_r, b_r, h_r for the three replicates as indicated in the second column of Table 2 and
- added random noise.

We have repeated this experiment 100 time where the coefficients of the polynomials and the parameters a_r, b_r, h_r have not been changed. Only the noise values were generated again in each of the 100 trials. We have carried out this experiment for almost

Table 2. Median absolute error of the parameters a_r, b_r *and* h_r

Noise	$a,\ b$ *and* h	*Median error*
$\sigma = 0.0001$	$a = (1.00, 0.85, 1.17)$	$Median(\max(\frac{\hat{a}}{a}, \frac{a}{\hat{a}})) = 1.021601$
	$b = (0.00, -0.18, 0.18)$	$Median(\lvert\hat{b} - b\rvert) = 0.09684161$
	$h = (1.00, 0.78, 1.28)$	$Median(\max(\frac{\hat{h}}{h}, \frac{h}{\hat{h}})) = 1.014896$
$\sigma = 0.6$	$a = (1.00, 0.85, 1.17)$	$Median(\max(\frac{\hat{a}}{a}, \frac{a}{\hat{a}})) = 1.052601$
	$b = (0.00, -0.18, 0.18)$	$Median(\lvert\hat{b} - b\rvert) = 0.1335851$
	$h = (1.00, 0.78, 1.28)$	$Median(\max(\frac{\hat{h}}{h}, \frac{h}{\hat{h}})) = 1.120215$

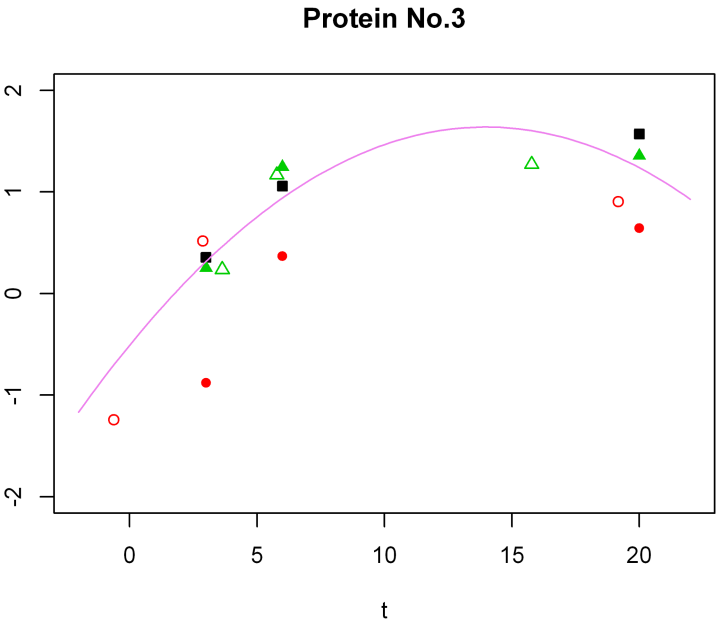


Fig. 2. An example for a good fit for the real data set

no noise ($\sigma = 0.0001$) and more significant noise ($\sigma = 0.6$). Table 2 shows the median absolute error for these experiments. Note that for the parameters a_r and h_r we measure the median absolute error in terms of the quotients $\max(\frac{\hat{a}}{a}, \frac{a}{\hat{a}})$ and $\max(\frac{\hat{h}}{h}, \frac{h}{\hat{h}})$, repsectively, since these coefficients are multiplicative factors and not additice terms like the coefficients b_r.

We now consider a real proteomics data set. For a time-resolved analysis of growth receptor signalling, human prostate cancer cells were stimulated for 3, 6, and 20 minutes and compared to a non-stimulated sample. Proteins were isolated, digested and phosphopeptides were enriched as described in [11]. Peptides of the four samples were quantitatively labeled with iTRAQ according to the manufacturer's protocol (Applied Biosystems), purified for MS (mass spectrometry) and separated via the nano-LC-MS/MS technology. Subsequently, the generated peptide spectra were queried against the UniProtKB/Swiss-Prot database using Mascot Daemon. More than 2000 proteins were measured in this experiment.

As mentioned earlier already, most of these proteins will not be directly involved in the growth receptor signalling process and it would therefore not make sense to estimate the parameters a_r, b_r and h_r based on all measured proteins. Instead, we have based the estimation of these parameters on 12 proteins which are known to be involved in the process with high probability. Figure 2 shows a typical example for the fitting result for these 12 proteins. After the correction based on the parameters a_r, b_r and h_r, the data (nonfilled symbols and the filled squares) fit quite well to the curve.

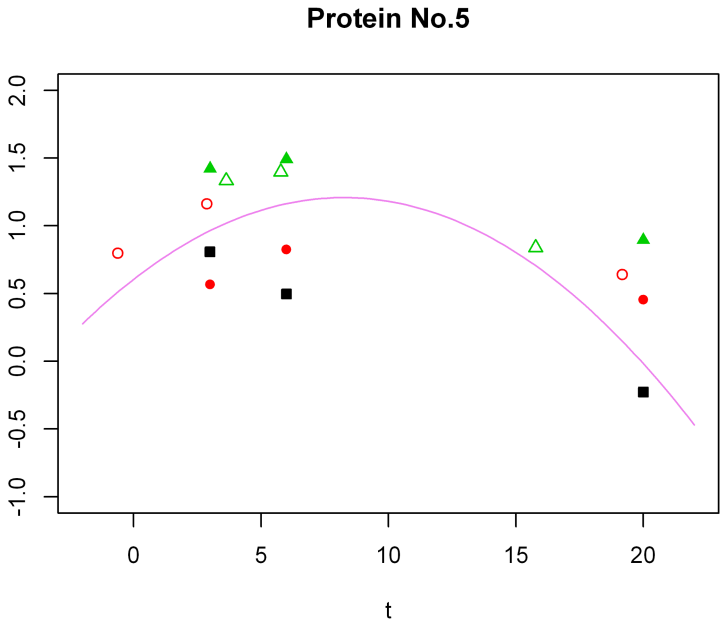


Fig. 3. An example that shows a deviating behaviour of the replicate represented by the squares

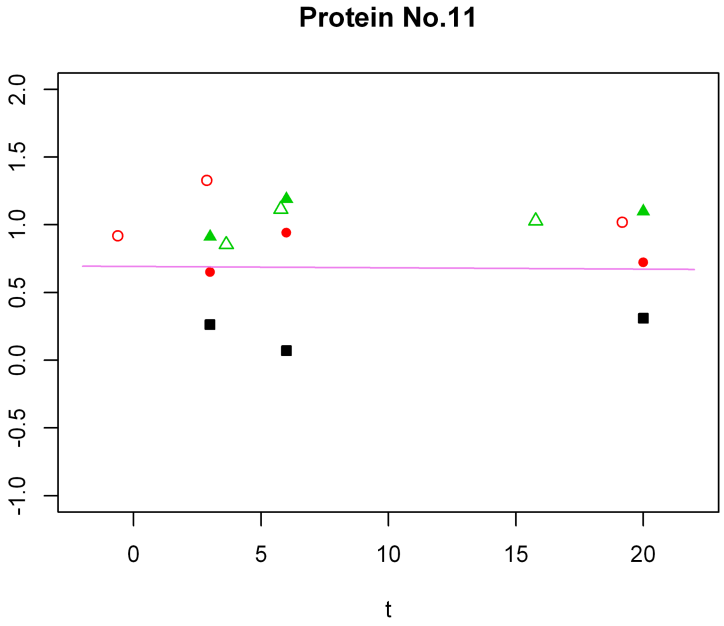


Fig. 4. Based on the Akaike information criterion a regression line instead of a polynomial of degree two is chosen

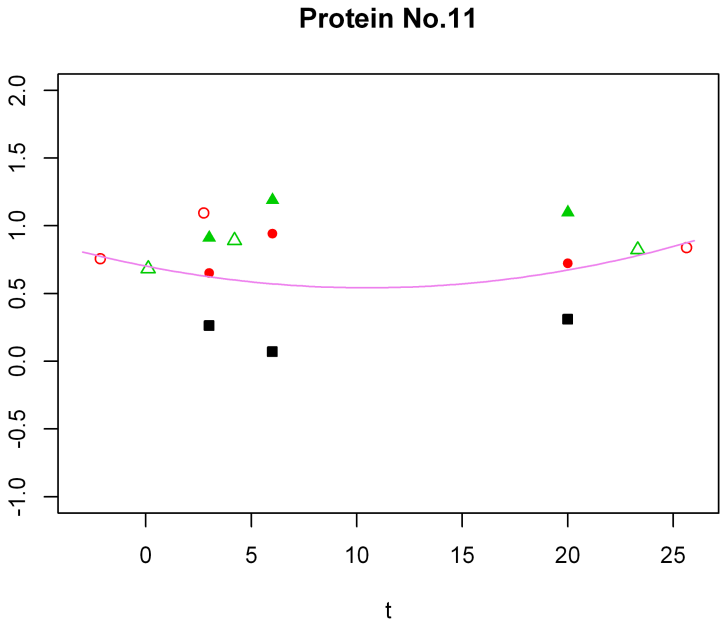


Fig. 5. Without the Akaike information criterion a polynomial of degree two is fitted to this protein showing inconsistent behaviour

But could this just be an effect of overfitting? In order to check this, we have generated 100 data sets which were filled with random numbers and have applied our method with and without estimating the parameters a_r, b_r and h_r. In average, the additional estimation of the parameters yielded a reduction of the mean squared error by 10% for the random data set. For our real data set, the reduction of the mean squared error was almost 60%.

Figure 3 shows an example with a good fit for two replicates, but one deviating replicate. The two replicates represented by the triangles and circles show at least the same tendency in regulation over time. However, replicate 1 (the squares) shows a completely different behaviour.

There is also an example where a regression line was preferred to the polynomal of degree two according to the Akaike information criterion. Figure 4 shows this example with quite incoherent behaviour of the three replicates.

Without the application of the Akaike information criterion one would always obtain a polynomial of degree two for all proteins. Figure 5 shows the result for the same protein as in Figure 4 when the Akaike information criterion is not applied. Note that without applying the Akaike information criterion the resulting value for the parameters a_r, b_r, h_r will also differ, since different fits of the course will also influence the choice of these parameters in the alternating optimisation procedure.

It is out of the scope of this paper to discuss the biological role of the 12 proteins and to discuss why the protein in Figure 4 deviates from what we expect. The parameters a_r, b_r and h_r derived from the 12 proteins where also used for the estimation of the curves for the other more than 2000 proteins. Those proteins that show a good fitting for the estimated curves are candidates that might also be involved in the growth receptor signalling process of prostate cancer. They can be filtered automatically, by choosing only those plots where the mean squared error for the fitted curve is small enough.

6 Future Work

We have presented a method to correct time shift, time scale and different intensities of reactions for short time series data from high-throughput experiments as they are common in genomics, proteomics and metabolomics. The work is still in an initial phase and needs more thorough evaluation with artificial data sets and real world data sets. More robust results can be obtained with robust regression and outliers – like the one in Figure 4 – could be detected automatically based on the weights that robust regression provides. It could also be useful to limit the range of the parameters a_r, b_r, h_r in order to avoid unrealistic changes like an extremely large time shift. This would lead to a constraint optimisation problem.

The computed curves can be used for cluster analysis to find groups of cell products with similar profiles in their behaviour over time.

References

1. Bar-Joseph, Z.: Analyzing time series gene expression data. Bioinformatics 20, 2493–2503 (2004)
2. Bezdek, J.C., Hathaway, R.J.: Convergence of alternating optimization. Neural, Parallel Sci. Comput. 11(4), 351–368 (2003)
3. Bar-Joseph, Z., Gerber, G., Jaakkola, T., Gifford, D., Simon, I.: Comparing the continuous representation of time series expression profiles to identify differentially expressed genes. Proc. Natl. Acad. Sci. USA 100, 10146–10151 (2003)
4. Bar-Joseph, Z., Gerber, G., Jaakkola, T., Gifford, D., Simon, I.: Continuous representations of time series gene expression data. J. Computational Biology 3, 341–356 (2003)
5. Ernst, J., Bar-Joseph, Z.: STEM: a tool for the analysis of short time series gene expression data. BMC Bioinformatics 7 (2006), doi:10.1186/1471–2105–7–191
6. Ndukum, J., Fonseca, L., Santos, H., Voit, E., Datta, S.: Statistical inference methods for sparse biological time series data. BMC Systems Biology 5(1), 57 (2011)
7. Hoaglin, D., Mosteller, F., Tukey, J.: Understanding Robust and Exploratory Data Analysis. Wiley, New York (2000)
8. Huber, P.: Robust Statistics. Wiley, New York (2004)
9. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19, 716–723 (1974)
10. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009)
11. Villén, J., Gygi, S.: The SCX/IMAC enrichment approach for global phosphorylation analysis by mass spectrometry. Nature Protocols 3, 1630–1638 (2008)

Formalizing Complex Prior Information to Quantify Subjective Interestingness of Frequent Pattern Sets

Kleanthis-Nikolaos Kontonasios and Tijl DeBie

University of Bristol, Intelligent Systems Laboratory,
Merchant Venturers Building,
Bristol, BS8 1UB, UK

Abstract. In this paper, we are concerned with the problem of modelling prior information of a data miner about the data, with the purpose of quantifying subjective interestingness of patterns. Recent results have achieved this for the specific case of prior expectations on the row and column marginals, based on the Maximum Entropy principle [2,9]. In the current paper, we extend these ideas to make them applicable to more general prior information, such as knowledge of frequencies of itemsets, a cluster structure in the data, or the presence of dense areas in the database. As in [2,9], we show how information theory can be used to quantify subjective interestingness against this model, in particular the subjective interestingness of tile patterns [3]. Our method presents an efficient, flexible, and rigorous alternative to the randomization approach presented in [5]. We demonstrate our method by searching for interesting patterns in real-life data with respect to various realistic types of prior information.

1 Introduction

Defining subjectively interesting patterns has quickly become an important subfield of Frequent Itemset Mining (FIM). In this context the interestingness of discovered patterns is associated not only with objective criteria, such as the support, but also with the user's prior knowledge about the data.

In particular the notion of 'surprisingness' [11] of an itemset has received a lot of attention. According to this definition, an itemset is deemed interesting if it significantly contrasts with the user's prior knowledge or expectations about the dataset. However, formalizing users' expectations and incorporating them into the data mining procedure gave rise to new challenges.

To meet these challenges two major approaches have been developed so far. The first was proposed in [4] and relies on creating randomized versions of the database using operators (*swaps*) which preserve the required dataset characteristics. The interestingness of an itemset can then be quantified by its *p-value* using empirical hypothesis testing. The method was initially designed for preserving the sums along rows and columns (*Row-Column marginals*) of the

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 161–171, 2012.
© Springer-Verlag Berlin Heidelberg 2012

database and it was later extended to more complicated prior knowledge, such as row clusters and itemset frequencies [5].

The second approach [2] is based on creating explicit probabilistic models (MaxEnt models) for the database, which encode the user's prior knowledge. In particular the Maximum Entropy principle [6] is employed for encoding the user's expectations about the dataset and the *InformationRatio* of a tile, an information-theoretic measure, was defined to quantify interestingness. Row and column marginals were used again as prior knowledge at the initial stage.

Despite the fact that randomization techniques broke new ground in quantifying interestingness they suffer from some drawbacks. First of all the MCMC processes which they employ are computationally intensive. In addition there is no theoretical guarantee that the resulting database sample is indeed a sample from the uniform distribution of databases under the specified prior knowledge. Most importantly, the scope of the method is limited to empirical hypothesis testing. The use of p-values may not be appropriate to some settings. Furthermore, the resolution of an *empirical* p-value is limited to one divided by the number of samples taken, such that patterns with small p-values (the ones that are interesting) cannot be ranked reliably.

In contrast, MaxEnt models subject to row and column sum constraints can be fitted using a very efficient polynomial-time algorithm. In addition these models are amenable to other uses besides empirical hypothesis testing. Furthermore, they can also be used for mining interesting *pattern sets* by employing set covering algorithms. Lastly, the preservation of the prior knowledge as *expected* values on certain statistics of the data, instead of the exact preservation, may often be more suitable, due to random fluctuations and errors in the dataset.

The main drawback of the MaxEnt approach for assessing the interestingness of itemsets is the limited number of types of prior knowledge it has been developed for. Because of this, the approach cannot yet be used for dealing with complex prior knowledge, or for iterative data mining schemes such as proposed in [5]. In this paper we address this problem by examining how different kinds of prior knowledge can be imposed on a MaxEnt model. In particular, we show that the MaxEnt modeling approach is able to deal with all types of prior knowledge that were considered earlier in [5] using swap randomization approaches.

In Section 2, we formalize the different types of prior knowledge considered in this paper, and we define some specific problem settings of potential practical interest by considering combinations of these. Section 3 is devoted to the presentation of the solutions of these problems. In Section 4, we recapitulate the details of the *InformationRatio*, a subjective interestingness measure for tiles [3] based on the MaxEnt model that was introduced in [2,9]. Finally, in Sec. 5 we report the empirical results of our methodology applied to two real-life databases.

2 Prior Knowledge in Itemset Mining

In this section, we will discuss various types of prior knowledge that may be of interest in practical (and possibly iterative) data mining applications. First we

consider specific types, and subsequently we define three problem settings based on certain combinations of these types that may be of particular interest.

2.1 Notation

Let $D \in \{0,1\}^{m \times n}$ denote a $m \times n$ binary database. The columns of the database are considered items and the rows transactions containing sets of items. The set of items/columns is denoted as I and the set of transactions/rows as T. Individual items will be denoted by i, and transactions by t. An itemset $\boldsymbol{i}$ is a subset of the set of columns/items, i.e. $\boldsymbol{i} \subseteq I$. A transaction set will be denoted as $\boldsymbol{t} \subseteq T$. A transaction $t \in T$ *supports* an itemset $\boldsymbol{i} \iff D(t,i) = 1, \quad \forall i \in \boldsymbol{i}$. The number of transactions supporting an itemset is called the *support* of an itemset and it is denoted $sup_{\boldsymbol{i}_k}$ for the itemset $\boldsymbol{i}_k$.

2.2 Prior Knowledge

The types of prior knowledge considered in this paper are the following.

Type 1: Marginals. The marginal of row t, denoted d_t^r, is the sum of all values in row t in database D. Column marginals are defined similarly (denoted by d_i^c for column i).

Row and column marginals have been used extensively in the past as prior knowledge [4,5,2,9,12] due to their intuitive meaning in a large variety of applications and datasets.

Type 2: Tile Sums. A tile τ is a pair of an itemset $\boldsymbol{i}$ and a set of transactions $\boldsymbol{t}$. A *tile sum* d_k^τ of tile k is the sum of the $|\boldsymbol{t}||\boldsymbol{i}|$ values of database elements indexed by the tile components.

Knowledge of a *tile sum* is a very general type of prior knowledge. It is easy to see that row/column marginals can be expressed as tile sums (each row and column corresponds to a tile). In addition, a set of 'tile sum' constraints can encompass the *RowCluster* prior knowledge examined in [5].

Type 3: Itemset Frequencies. The frequency of an itemset is the number of the transactions that *support* the itemset.

Tiles and itemset frequencies as prior knowledge are similar to each other. Their difference is that in the case of tiles a specific submatrix in the database model is targeted by the modelling procedure and the frequency of the corresponding itemsets is modelled implicitly. Quite usefully, tile sums can be used conveniently for modelling both exact and noisy itemset frequencies. On the other hand modelling itemset frequencies captures a global view of the dataset, as transactions are not specified. However this comes with an additional computational cost, as will be shown later. Both tile and itemset constraints can be well suited for an iterative data mining scheme, depending on the kind of patterns being mined (tiles or itemsets).

2.3 Problem Statements

It is our objective to create a probabilistic database model that encodes the user's prior knowledge about the data, by means of constraints the model must satisfy. In Section 3, we will examine how this can be done efficiently in the case where the prior knowledge can be expressed as constraints on the database statistics considered in Section 2.2. In particular, we will consider the following three problem settings:

We wish to highlight the following three modelling problems, combining different kinds of prior knowledge:

Problem 1: Marginals (Type 1). *Given a binary database D, construct a probabilistic model P for D that preserves in expectation the row and column marginals.*

Problem 2: Tile sums and Marginals (Types 1+2). *Given a binary database D and a set of tiles, construct a probabilistic model P for D that preserves in expectation the row and column marginals and tile sums.*

Problem 3: Itemset frequencies and Column marginals (Types 1+3). *Given a binary database D and a set of itemsets with their frequencies, construct a probabilistic model P for D that preserves in expectation the column marginals and itemset frequencies.*

3 Modelling Prior Knowledge Using the Maximum Entropy Principle

We have now formalized some simple but powerful types of prior knowledge, and discussed three problem settings of practical interest. Space restrictions do not permit us to provide full detail about how these problems are solved. However, in this section, we will give indications as to how this is done, and show the mathematical shape of the resulting distribution for each of the Problem types discussed. Detailed solutions for *Problems 2,3* and proofs of Theorems 2 and 3 are given in the accompanying technical report [10] of this paper.

3.1 Maximum Entropy Modelling

Each of the problems defined in Sec. 2.3 searches for a probability distribution that satisfies a number of constraints. These constraints are defined in terms of database properties that need to be preserved in expectation. Let us denote these properties as functions f_s defined over the space of possible values for the database. Then, these constraints can be written in terms of the expected value of these properties, where the expectation is taken with respect to P:

$$\sum_D P(D) f_s(D) = d_s \quad \forall s. \tag{1}$$

Here the sum is over all possible values of the database. To define a valid probability distribution, additionally P needs to be constrained to be positive and normalized (summing to one).

In general, this set of constraints uniquely determines P only for the most trivial cases. Consequently, an additional inductive bias is needed for determining the whole set of model parameters. In [2] the case was made to use the distribution of maximum entropy as the least biased distribution among those satisfying the constraints [6]. Finding this *MaxEnt* distribution consists of solving the optimization problem:

$$\max_P - \sum_D P(D) \log(P(D)), \tag{2}$$

subject to the constraints specified in Eq.(1), complemented with the normalization and positivity constraints.

In the following subsection this general formulation is specified for each of the three problem settings from Section 2.2 by defining the constraint functions f_s for each of them and the resulting distributions are presented.

3.2 Problem 1: Modelling Marginals

For the *Marginals* problem the constraints f_s can be expressed as:

$$f_{s_t}(D) = \sum_i D_{t,i} = d_t^r \ \forall t \in T, \qquad f_{s_{m+i}}(D) = \sum_t D_{t,i} = d_i^c \ \forall i \in I. \tag{3}$$

Theorem 1. *The Maximum Entropy distribution for Problem 1 is given by [2]:*

$$P(D) = \prod_{t,i} \frac{\exp\{D_{t,i}(-\lambda_t^r - \lambda_i^c)\}}{1 + \exp\{-\lambda_t^r - \lambda_i^c\}},$$

where λ_t^r and λ_i^c are the Lagrange Multipliers for row t and column i respectively.

The resulting distribution factorizes in a product of $m \times n$ independent Bernoulli distributions. Each of them defines the distribution for a particular entry in the binary database. It is noted that the success probability of the *(t,i)* entry depends only on the Lagrange Multipliers of the row t and column i.

3.3 Problem 2: Modelling Tile Sums and Marginals

To solve Problem 2 the formulation of Problem 1 is used again for modelling the marginals and only one additional set of constraints for the tiles τ_k is required. The additional set of constraints for modelling tiles is:

$$f_{s_{m+n+k}}(D) = \sum_{t,i \in \tau_k} D_{t,i} = d_k^\tau \ \ \forall k.$$

Theorem 2. *The Maximum Entropy distribution for Problem 2 is given by:*

$$P(D) = \prod_{t,i} \frac{\exp\{D_{t,i}(-\lambda_t^r - \lambda_i^c - \sum_k \lambda_k I_k(D_{t,i}))\}}{1 + \exp\{-\lambda_t^r - \lambda_i^c - \sum_k \lambda_k I_k(D_{t,i})\}},$$

where λ_t^r, λ_i^c and λ_k^τ are the Lagrange Multipliers for row t, column i and tile τ_k respectively, and $I_k(D_{t,i})$ an indicator function with $I_k(D_{t,i}) = 1$ if $(t,i) \in \tau_k$

This solution presents some similarities with the one obtained for the *Marginals* problem. Again the resulting distribution is a product of independent Bernoulli trials, one for each entry in the database. In this case however, the success probability depends not only on the corresponding row and column multiplier but also on the Lagrange Multipliers for the tiles that the entry participates in.

3.4 Problem 3: Modelling Itemset Frequencies and Column Marginals

For the column marginal information in Problem 3, the constraints in Eq.(3) are reused. Additionally, a set of constraints for the itemset frequencies is used:

$$f_{s_{m+k}}(D) = \sum_t I_{i_k}(t) = sup_{i_k} \quad \forall \boldsymbol{i}_k \subset I,$$

where $I_{i_k}(t)$ is an indicator function with $I_{\boldsymbol{i}}(t) = 1$ if transaction t supports itemset $\boldsymbol{i}$.

Since we do not impose a constraint on row marginals in this Problem, the resulting P is a product distribution of identical distributions for each single transaction. Thus, it suffices if we compute a model for a transaction, rather than for the entire database as we did for the other Problems.

In general the MaxEnt optimization problem produces probability distributions that belong to the exponential family. In particular, for all the problems examined so far the resulting distribution is a product of Bernoulli distributions.

In addition, for all choices of the functions f_s considered in this paper, the distribution factorizes as a product of factors, which are called *potential functions* in the Graphical Models(GM) or Markov Networks (MN) literature [8]. Thus, we can rely on existing techniques developed in this research community for inferring the parameters of the distributions.

The difference between Problems 1 and 2 with Problem 3 is that for the former cases the distribution factorizes as a product of functions over a single variable only. The corresponding MN representation is a graph with no edges between the variables/nodes and employing sophisticated MN techniques for parameter fitting is not necessary.

In contrast, the dependencies between items imposed by the constraints on the itemset frequencies lead to a factorization of P over *sets of variables*, namely those that are jointly in such an itemset. The MN representation will contain a clique for each of the itemsets, and each itemset gives rise to a potential function in the factorized probability distribution corresponding to this MN.

Theorem 3. *The Maximum Entropy distribution for Problem 3 is given by:*

$$P(D) = \prod_w \frac{1}{Z_w(\lambda^c, \lambda^{\boldsymbol{i}})} \cdot \exp\{-\sum_{k \in w} \lambda_k^{\boldsymbol{i}_k} \sum_t I_{\boldsymbol{i}_k}(t) - \sum_{i \in w} \lambda_i^c \sum_t D_{t,i}\},$$

where Z_w is the partition function for every connected component in the tree.

The shape of the solution presented above reveals the dependencies between items that belong to the same itemset constraint. Due to these dependencies, computing the gradient here is less easy than in Problems 1 and 2. In each iteration of the gradient descent method, the computation of the partition function is required, for which MN techniques are necessary. In this paper, we opted to use the *Junction Tree* algorithm for this purpose [8], a well-known technique for performing probabilistic inference. While this method is only exact for cliques that form a junction tree, it is sufficiently general for the experiments we carried out. For cliques that do not form a clique tree, approximate techniques are available [8]. More details can be found in the accompanying report [10].

4 Finding the Most Interesting Set of Tiles

So far we have described the computation of MaxEnt models that encode the user's prior knowledge about the database. These models can be exploited in a lot of different ways. In this section we explain how they can be used to compute the most subjectively interesting set of patterns. Like in [3,9], we focus on quantifying interestingness of *sets* of tiles instead of individual tiles in isolation, in order to limit the redundancy between the results. In particular, we recapitulate here the main parts of the method proposed in [2,9] to assess the interestingness of tile patterns given a background model and we discuss the adaptations needed to deal with the more complex MaxEnt models in this paper. To this end, for each tile its *InformationContent*, *DescriptionLength* and finally *InformationRatio* are defined.

4.1 The Information Ratio of a Tile

Since a probabilistic model for the database is at hand, a natural choice for defining interestingness of tiles is using concepts from Information theory. In particular the definition of *SelfInformation* of a random variable given in [1] is extended in order to encompass tiles. The *InformationContent* of a tile is defined as $IC(\tau) = -\log(P(\tau))$. For Problems 1 and 2, the *InformationContent* can be calculated efficiently as the sum of $|I_\tau||T_\tau|$ log probabilities of independent Bernoulli variables, where I_τ and T_τ are the items and transactions in tile τ. For Problem 3, the (log-) probability can be computed efficiently by performing inference for the variables I_τ using the Junction Tree algorithm.

In [2,9], it was pointed out that not only the information content is relevant in defining interestingness, but also the description complexity of the tile itself. Indeed, the only way to convey the information to the data miner captured by

the tile would be to communicate the tile, which comes at a cost itself. This raises however the question of how the description complexity of a tile can be measured. Here we use the approach from [9], i.e.:

$$DL(\tau) = -\sum_{t \in T_\tau} \log(p_t) - \sum_{t \notin T_\tau} \log(1 - p_t) - \sum_{i \in I_\tau} \log(p_i) - \sum_{i \notin I_\tau} \log(1 - p_i),$$

where $p_i = \frac{column\ marginal}{number\ of\ rows}$ and $i \in I$ (correspondingly for $p_t, t \in T$ for columns).

The ratio between the *InformationContent* and the *DescriptionLength* is therefore a representation of the compression ratio of information embedded in the tile: it is the ratio of the amount of information the data miner gets about the database by seeing the tile, and the number of bits needed to transmit this information. This ratio is the interestingness measure proposed in this paper, named *InformationRatio*.

4.2 Calculating the Most Interesting Set of Tiles

The *InformationRatio* measure allows the ranking of individual tiles. Finding the most interesting set of tiles (tiling), however, can not be trivially calculated by this ranking as the itemsets at the top of the list are usually highly redundant. A natural option is employing a set covering algorithm. Indeed, the problem of finding the tiling with the largest overall *InformationContent* subject to an upper bound on the overall *DescriptionLength* can be reduced to a *Budgeted Maximum Set Coverage Problem*. This problem is well known and a greedy approach is sufficient for solving it with a good approximation [7,2,9].

5 Experiments

Datasets. We tested our approach on two real-world datasets, the *KDDcoauthors* and *KDDabstracts* datasets.

To create the *KDDcoauthors* dataset, entries regarding the authors of all the main conference and workshop papers and posters from both the research and industrial track of KDD conference were collected. A transactional database involving 1669 transactions over 3058 items was formed by considering each paper a transaction involving different authors. A common phenomenon in academic publications is that students publish often with their supervisors. These relations are well-known and can be retrieved from various sources. In this paper, we used the student-supervisor data kindly provided to us by Chi Wang [13]. For someone aware of this fact, the student-supervisor collaborations are less interesting in comparison to other collaborations e.g. cross-departmental or between the academia and the industry. Following this rationale, the prior knowledge for the experiments on *Tile sums-Marginals* and *Itemset frequencies-Column marginals* modelling are formed by student-supervisor pairs.

The *KDDabstracts* dataset [2,9] was created by collecting abstracts from papers published in KDD. Each abstract forms a transaction in the binary database

and each stemmed word an item. The formed database consists of 843 transactions over 6159 words. It is reasonable to believe that the associations between words making up a keyword are known to a domain expert, such that he is not interested in patterns that can be explained by these associations. Consequently a 'prior knowledge' dataset consisting of 228 keywords is formed.

Experimental Settings. For each dataset a table with the top-5 most interesting results is presented. Each table contains subtables with results obtained using the different kinds of prior knowledge as in *Problems 1-3* from Section 2.3 in order of increasing information when possible. The aim of this treatment is to give an example how the results evolve under different kinds of prior knowledge rather than presenting per se the most interesting set of tiles for the particular datasets, as this depends on the actual prior knowledge of the user.

Results. Regarding the *KDDcoauthors* dataset, the first list in Table 1(a) presents most of the drawbacks in using frequency as an interestingness measure. First of all, the itemsets are very small in cardinality. Furthermore, a large amount of redundancy is observed, with variations of the itemset {*Yiming Ma,Wynne Hsu,Bing Liu*} appearing often. Furthermore, individual authors with a large number of publications in KDD dominate the resulting list.

The second list presents the most interesting itemsets using the *InformationRatio* measure and *Marginals* modelling. The absence of the disadvantages of the previous list is evident. The itemsets are larger, due to *InformationRatio* preferring approximately square tiles. In addition, the redundancy is removed. Modelling marginals results in excluding itemsets of small cardinality that contain individual authors with a large number of publications in KDD. The list, however, contains student - supervisor pairs in eight itemsets, with many of them containing only such relations. Modelling tile sums and itemset frequencies is aiming to refine such results.

Indeed, the itemsets deemed interesting using additional tile sum constraints, while sharing all the advantages of their counterparts in the second list, contain no student-supervisor pairs recorded in our database. The fourth list (*Problem 3*) contains three such relations, but they are only part of larger itemsets. In particular one of these itemsets, {*Eddie C. Shek, Richard R. Muntz, Edmond Mesrobian*}, form a chain of supervisor-student relations.

Table 1(b) presents the corresponding results for the *KDDabstracts* dataset. Similar conclusions as above can be drawn. The first list is extremely redundant with common knowledge set of words exclusively. The redundancy is removed in the second list and the sets of words are interesting since most of them describe subareas or tasks from the KDD community. The keywords in prior knowledge are filtered with the next experiments. Interestingly, it is observed that the itemset {*unlabelled, labelled, supervised, learn*} which contains the itemset {*supervised, learn*} is removed and another variant enters the top-5 list ({*unlabelled, labelled, data*}). Itemsets from the second list not in the prior knowledge database are in general retained.

Table 1. Top-5 most interesting itemsets for the *KDDcoauthors* dataset (a) and *KDDabstracts* dataset (b) using Frequency (upper table), Information Ratio with constraints on Marginals (second table), Information Ratio with constraints on Marginals and Tile sums (third table) and Information Ratio with constraints on Column marginals and Itemset frequencies (lower table). The second column contains the corresponding support, the third the conditional Information Ratio [2,9] and the last column indicates whether the itemset or its subitemsets are contained in the Prior knowledge database

(a) *KDDcoauthors dataset*

Itemsets	*Support*	*IR*	*Prior Knowl.*
Frequency			
W. Hsu, B. Liu	9	-	No
Y. Ma, B. Liu	8	-	No
M. Ester, H.P. Kriegel	7	-	No
C.C. Aggarwal, P.S. Yu	7	-	No
H. Tong, C. Faloutsos	7	-	No
Inf. Ratio - Constraints on Marginals			
R. Bhaumik, C. Williams, R. D. Burke, B. Mobasher	3	1.2482	Yes
Y. Ma, W. Hsu, B. Liu	6	1.2414	No
J. Wu, J. Chen, H. Xiong	4	1.2116	No
W. Perrizo, W. Jockheck, A. Perera, D. Ren, W. Wu, Yi Zhang	2	1.1744	Yes
K. Zhang, B.A. Shaphiro, J. Tsong-Li Wang, D. Shasha	3	1.1353	Yes
Inf.Ratio - Constraints on Marginals and Tile sums			
Y. Ma, W. Hsu, B. Liu	6	1.2223	No
S. Tao, N. Anerousis, X. Yan	2	0.9804	No
M.L. Antonie, A. Coman	3	0.9530	No
A. Gershman, G. Wei, T. Gardinier	2	0.9273	No
N. Ahmed Syed, K. Kay Sung, H. Liu	2	0.9037	No
Inf.Ratio - Constraints on Column marginals and Itemset frequencies			
W. Jockheck, A. Perera, D. Ren, W. Wu	2	2.0327	No
K. Zhang, B.A. Shaphiro, J. Tsong-Li Wang	3	2.0279	No
C. Williams, R. Bhaumik, R.D. Burke	3	2.0120	No
E.C. Shek, R.R. Muntz, E. Mesrobian	2	1.8981	Yes
D. Kumar, M. Potts, R.F. Helm	2	1.6687	No

(b) *KDDabstracts dataset*

Itemsets	*Support*	*IR*	*Prior Knowl.*
Frequency			
data, paper	389	-	No
data, algorithm	334	-	No
data, mine	312	-	No
data, propos	308	-	No
data, result	305	-	No
Inf.Ratio - Constraints on Marginals			
svm, machin, vector, support	25	0.8801	Yes
art, state	39	0.7954	No
unlabelled, labelled, supervised, learn	10	0.7261	Yes
gene, express	25	0.7243	Yes
rule, associ, mine	36	0.7187	Yes
Inf.Ratio - Constraints on Marginals and Tile sums			
art, state	39	0.7918	No
row, column, algorithm	12	0.6876	No
unlabelled, labelled, data	14	0.6808	No
answer, question	18	0.6634	No
recal, precis	14	0.6496	No
Inf.Ratio - Constraints on Column marginals and Itemset frequencies			
labelled, unlabelled, semi, supervised	8	2.0996	No
svm, vector, train, support, machin	12	2.0708	Yes
outlier, detect, high, show, base, propos	9	2.0597	No
privaci, individu, mine, result	8	1.9762	No

6 Conclusions

In this paper, we have shown how the MaxEnt modelling strategy [2,9] can be extended for use with flexible types of prior information. This makes the MaxEnt modeling approach a complete alternative to the swap randomization approaches [5]. In contrast with swap randomizations, the MaxEnt approaches presented here do not suffer from convergence issues. Indeed, with the possible exception of itemset constraints, when they do not form a junction tree, fitting each of the MaxEnt models discussed can be done in polynomial time. Furthermore, their analytical form makes them easy to use in the definition of analytically computable measures of interestingness, as we demonstrated for tiles.

References

1. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience (2005)
2. DeBie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. In: Data Mining and Knowledge Discovery (2010)
3. Geerts, F., Goethals, B., Mielikainen, T.: Tiling databases. In: Discovery Science (2004)
4. Gionis, A., Mannila, H., Mielikainen, T., Tsaparas, P.: Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(3) (2007)
5. Hanhijarvi, S., Ojala, M., Vuokko, N., Puolamaki, K., Tatti, N., Mannila, H.: Tell me something I don't know: Randomization strategies for iterative data mining. In: Proc. of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2009 (2009)
6. Jaynes, E.T.: On the rationale of maximum-entropy methods. Proceedings of the IEEE 70 (1982)
7. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Information Processing Letters 70 (1999)
8. Koller, D., Friedman, N.: Probabilistic Graphical Models, Principles and Techniques. MIT Press (2009)
9. Kontonasios, K.-N., DeBie, T.: An information-theoretic approach to finding informative noisy tiles in binary databases. In: SDM, pp. 153–164. SIAM (2010)
10. Kontonasios, K.-N., DeBie, T.: Formalizing complex prior information to quantify subjective interestingness of frequent pattern sets (supplementary document). Technical report, University of Bristol (2011), `https://patterns.enm.bris.ac.uk/projects/Mining`, subjectively interesting patterns using prior knowledge
11. Silberschatz, A., Tuzhilin, A.: What makes patterns interesting in knowledge discovery systems. IEEE Trans. on Knowl. and Data Eng. 8(6), 970–974 (1996)
12. Tatti, N., Mampaey, M.: Using background knowledge to rank itemsets. Data Min. Knowl. Discov. 21, 293–309 (2010)
13. Wang, C., Han, J., Jia, Y., Tang, J., Zhang, D., Yu, Y.: Mining advisor-advisee relationships from research publication networks. In: KDD 2010 (2010)

MCut: A Thresholding Strategy for Multi-label Classification

Christine Largeron, Christophe Moulin, and Mathias Géry

Université de Lyon, F-42023, Saint-Étienne, France CNRS UMR 5516, Laboratoire Hubert Curien Université de Saint-Étienne Jean Monnet, F-42023, France

Abstract. The multi-label classification is a frequent task in machine learning notably in text categorization. When binary classifiers are not suited, an alternative consists in using a multiclass classifier that provides for each document a score per category and then in applying a thresholding strategy in order to select the set of categories which must be assigned to the document. The common thresholding strategies, such as RCut, PCut and SCut methods, need a training step to determine the value of the threshold. To overcome this limit, we propose a new strategy, called MCut which automatically estimates a value for the threshold. This method does not have to be trained and does not need any parametrization. Experiments performed on two textual corpora, XML Mining 2009 and RCV1 collections, show that the MCut strategy results are on par with the state of the art but MCut is easy to implement and parameter free.

1 Introduction

Recently, multi-label classification has attracted much attention from the researchers in pattern recognition as well as in data mining and more generally in machine learning, caused by an increasing number of applications like for example text categorization which is considered in this article. Given a set of pre-defined categories (or classes) and a training set of pre-classified documents, the task consists in learning the categories' descriptions in order to be able to classify a new document into the categories [18,23]. In the context of multi-label classification, each document is associated with one or more labels whereas each document is associated with exactly one label in the single-label classification. Moreover, when the classifier has three or more categories to choose from, it is a multiclass classification whereas it is a binary classification when the number of categories is limited to two, usually a category and its complement. With the imbalance between the labels, their overlapping, in the case of multi-label classification, pose new challenges and give opportunities to design new algorithms more suited for document mining. Among the methods developed to solve this problem, Tsoumakas distinguished two main approaches [20].

In the first one, the multi-label classification problem is transformed into one or more single-label classification problems. The most common transformation consists in learning one binary classifier per label [9]. However, this approach does

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 172–183, 2012.
© Springer-Verlag Berlin Heidelberg 2012

not consider the correlations between the different labels of each instance. In fact, as noted by Sebastiani, it requires that categories are stochastically independent of each other but this hypothesis is not necessarily verified, in particular when the categories are organized in hierarchy, as for instance for web pages hosted by Internet portals [18].

The second approach includes usual methods, called algorithm adaptation methods [20], which have been extended in order to handle multi-label data. We can cite, for instance, the C4.5 algorithm adapted by Clare and King [1], the Adaboost.MH and Adaboost.MR developed by Schapire *et al.* [17] and its extension adaboost.MH [3], the algorithm, based on the SVMs principle, proposed by Elisseeff [7] or the MMP [4] and the BP-MLL [25] algorithms derived from neural networks. Similarly, several improvements have been designed for the kNN classifier, as for example, those developed by [13] or the ML-kNN adaptation of the kNN lazy learning algorithm [24]. However, in this case, the solution brought to the multi-label problem is dependant of the classification algorithm.

In the general case, the result obtained for a new document with a multiclass classifier is not a set of categories but a ranked list of the categories, eventually associated with a score corresponding to the estimated probability of the category and the user must decide which categories must be retained. Unfortunately, in many real applications, the number of categories for a new document is unknown in such a way that the choice of the number of categories is not obvious. In this case, the selection of the final categories assigned to the document requires a post-processing. Some approaches, as the Metalabeler proposed by Tang *et al.* [19], learn the number of labels to associate with documents, while others, which we will consider in the following, consist in determining a threshold such that only categories with higher scores are selected.

This threshold can be defined using two main approaches. In the first one, called *probability thresholding* [18], the threshold is inferred, by a theoretical result, from probabilities computed by the classifier. However, this approach can be only applied when on the one hand the probabilities of class membership are estimated by the classifier and on the other hand, the effectiveness of the classification is evaluated by decision measures. The second approach, called *experimental thresholding* [18], consists in testing different values as thresholds on a validation set and choosing the value which maximizes the effectiveness. Different algorithms have been reported in the literature. The most common are the *RCut* method [15,22] also called *t-per-doc* thresholding [10], the *PCut* or proportional thresholding used in [10,11,15,23,22] and finally, the *SCut* or *CSV thresholding* [2,15,22]. In these three methods as well as in their variants like *RTCut* or *SCutFBR* [22], the threshold must be fixed by a user or estimated from a training set. To overcome this drawback, we propose in this article a method, called *Maximum Cut (MCut)* which automatically determines, without a learning step, the set of categories to which a new document belongs to, from the categories' scores of this document. We also present a comparative study of these different methods on several corpora. These experiments have been done on usual benchmarks composed of short documents such as the well known Reuters

Collection [12] but also on a large collection of XML documents extracted from the Wikipedia encyclopedia: the INEX XML Mining collection [5].

In the aim of introducing our notations, a brief presentation of the vector space model (VSM - [16]), used to represent the documents, is given in section 2. The usual thresholding strategies *RCut, SCUt, PCut* as well as the Maximum Cut strategy are respectively described in detail in sections 3 and 4. The following sections 5 and 6 present the comparative experiments and the results obtained on different datasets.

2 Document Model for Categorization

The vector space model, introduced by Salton et al. [16], has been widely used to represent textual documents as vectors which contain the terms weights. Given a collection D of documents, an index T, where $|T|$ denotes the cardinality of T, gives the list of terms (or features) encountered in the documents of D. In order to calculate weights, the TF.IDF formula can be used [16]. TF (Term Frequency) corresponds to the relative frequency of a term t_j in a document d_i and IDF (Inverse Document Frequency) measures the discriminatory power of the term t_j.

3 Thresholding Strategies

In the context of multi-label classification, one score $\phi(\overrightarrow{d_i}, c_k)$ is produced by the multiclass classifier for each document d_i and each category c_k where $c_k \in C$ with $C = \{c_1, \ldots, c_r\}$ denotes the set of categories. Given these scores, the problem consists in determining the set of categories $\hat{L}(d_i) \subseteq C$ which must be attributed to the document d_i.

To solve this problem, three main algorithms have been proposed: *RCut*, *PCut* and *SCut*. They are based on the same principle: given a previous chosen threshold, they compare the scores returned by the classifier at this threshold in order to select the categories with scores higher than the threshold. The value of the threshold can be either specified by the user or it can be learned from a training set. These common strategies are described below.

3.1 *RCut* or *t-per-document Thresholding*

The *RCut* strategy, also called *t-per-document* thresholding, classifies each document in exactly t categories [10,15,22]. Indeed, given a document d_i, the scores $(\phi(\overrightarrow{d_i}, c_k), k = 1, \ldots, r)$ are ranked and the t top ranked categories are assigned to d_i.

Note that if t equals 1, *RCut* is equivalent to the $\arg\max_k(\phi(\overrightarrow{d_i}, c_k), k = 1, \ldots, r)$ which predicts the most probable category. Moreover, as *RCut* is document pivoted, it is well-suited for online applications where the decisions for a document are independent from those for the others.

In view to distinguish categories with the same ranking, [22] introduced $RTCut$, an extension of $RCut$ which combines the categories ranks and the scores $\{\phi(\overrightarrow{d_i}, c_k), k = 1, \ldots, r\}$ but like $RCut$, this improvement requires the choice of a threshold.

However, $RCut$, as well as $RTCut$, is global since it affects the same number of categories to each document. From this point of view, it can suffer from inconsistency in so far as a document d_i can be classified in the category c_k and $d_{i'}$ not assigned to c_k when $\phi(\overrightarrow{d_i}, c_k) < \phi(\overrightarrow{d_{i'}}, c_k)$.

3.2 *PCut* or *Proportional Thresholding*

If $RCut$ is document-pivoted, $PCut$ is category-pivoted. Indeed, $PCut$, also called *proportional thresholding*, assigns to the category c_k a number of documents n_k in function of the percentage of documents belonging to this category in the training set [15,22]. More precisely, given a category c_k, the scores $(\phi(\overrightarrow{d_i}, c_k), i = 1, \ldots, |D|)$ are ranked and the n_k top ranked documents are assigned to the category with $n_k = P(c_k) \times x \times r$ where $P(c_k)$ is the prior probability for a document to belong to c_k, r, is the number of categories, and x is a parameter which must be estimated using a training set.

Contrary to $RCut$, $PCut$ is not suitable for online applications because the decision for a new instance requires the scores for all the others. Moreover, it is clear that its performance depends on the stability of the categories distribution in the training set and in the rest of the corpus.

3.3 *SCut* or *CSV Thresholding*

At the difference of $PCut$ and $RCut$ which are global strategies, $SCut$, also called *CSV thresholding*, is local in the sense that a threshold is fixed per category [2,15,22]. For each category, different values are tested and the value which maximizes the effectiveness of the classifier on a training set is chosen. But, due to the risk of overfitting, there is no guarantee that the optimum value determined on the training set will be the best one in the whole collection. Moreover, as noted by Yang [22], many false alarms result from a too low threshold while many misses for the category result from a too high threshold, notably when there are few instances associated with the label. The extension $SCutFBR$ was introduced to limit the affect of the threshold on the effectiveness measure for rare categories [22].

However, $SCutFBR$, like the previous strategies, requires a value for the parameter, which can be either specified by the user or learned using a training set, eventually with a risk of overfitting. To avoid such selection, we propose in this article the Maximum Cut Strategy, noted $MCut$.

4 The Maximum Cut Thresholding ($MCut$)

The Maximum Cut thresholding $MCut$ determines automatically a threshold for each document in order to select a subset of categories from the scores

$(\phi(\overrightarrow{d_i}, c_k), k = 1, \ldots, r)$ for the document d_i. The underlying idea consists in assigning to d_i the subset of categories corresponding to scores really higher than the others. This leads to identify the highest difference between successive scores and to select as threshold the middle of the interval defined by these last scores.

This method is based on the following principle. Given a document d_i, the scores $(\phi(\overrightarrow{d_i}, c_k), k = 1, \ldots, r)$ are ranked in decreasing order. The sorted list obtained is denoted $S = (s(l), l = 1, \ldots, r)$ where $s(l) = \phi(\overrightarrow{d_i}, c_k)$ if $\phi(\overrightarrow{d_i}, c_k)$ is the l^{th} highest value in S. Let t be the highest difference between successive scores (i.e. the maximum gap for S) defined by:

$$t = \arg\max\{(s(l) - s(l+1)), l = 1, \ldots, r-1\} \tag{1}$$

The value *mcut* retained as threshold is then defined by:

$$mcut = \frac{s(t) + s(t+1)}{2} \tag{2}$$

The categories assigned to d_i are those corresponding to a score $\phi(\overrightarrow{d_i}, c_k)$ higher than *mcut*:

$$\hat{L}(d_i) = \{c_k \in C | \phi(\overrightarrow{d_i}, c_k) > mcut\} \tag{3}$$

The $MCut$ algorithm is presented in Algorithm 1 while Figure 1 gives a graphical illustration of this method. A curve of the scores in their decreasing order (*i.e.* $s(l), l = 1, \ldots, r$ in function of l) is drawn for two documents: d_1 (on the left side) and d_2 (on the right one). With $MCut$, the document d_1 is assigned to one category (on the left) while the document d_2 is assigned to three categories (on the right).

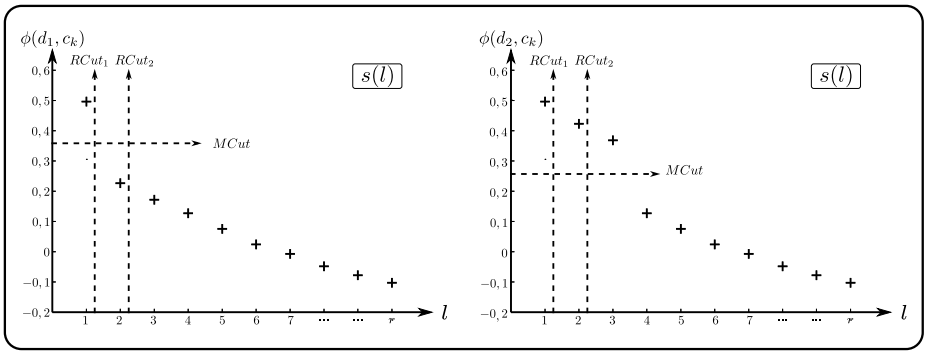


Fig. 1. Illustration comparing RCut and MCut thresholding strategies

$MCut$ presents the advantage to be local: the number of categories assigned to a document is not the same for all the documents. Moreover, $MCut$ requires neither a training set in order to select a threshold nor to know the scores

computed by the classifier over the whole collection. So it is suited for online application. It does not suppose that the distribution of the categories stays stable. Finally, it does not depend on a specific algorithm but its performance depends on the discriminative power of the classification algorithm. $MCut$ is

Data: $Tab[k] = \phi(\overrightarrow{d_i}, c_k) : k = 1 \dots r$
Result: $\hat{L}(d_i)$
$\hat{L}(d_i) = \emptyset$;
$S = sort(Tab)$;
$Sdiff[ind] = (S[ind] - S[ind+1]) : ind = 1 \dots r-1$;
$t = index(max(Sdiff))$;
$mcut = \frac{S[t]+S[t+1]}{2}$;
for $k \leftarrow 1$ **to** r **do**
  if $Tab[k] > mcut$ **then**
    $\hat{L}(d_i) = \hat{L}(d_i) \cup c_k$;
  end
end
return $\hat{L}(d_i)$

Algorithm 1. $MCut$ algorithm

compared with the $RCut$ strategy in Figure 1. For $RCut_1$ (resp. $RCut_2$), the t parameter is set up to 1 (resp. 2). The same set of categories is affected to d_1 by $RCut_1$ and $MCut$, while $RCut_2$ assigns one category more (see Figure 1 : left side). In the second case, the set of categories is different (see Figure 1 : right side). d_2 belongs to three categories with $MCut$ strategy while it is still associated to one category (resp. two categories) with the $RCut_1$ (resp. $RCut_2$) strategy.

5 Experiments

In order to compare the proposed $MCut$ thresholding strategy with the usual thresholding strategies $RCut$ and $PCut$, we performed several experiments on two text categorization collections. As indicated by Yang [22], we will not considered $SCut$ strategy which does not provide good results. After presenting the collections, we will define criteria used for the evaluation and the experimental protocol.

5.1 Collections Description

Two collections are used to evaluate the thresholding strategies. These collections are split into a training set and a testing set that are described in Table 1.

The first collection is the XML Mining collection 2009[1] [6]. It is composed of 54 632 XML documents extracted from the Wikipedia XML Corpus. This

[1] `http://www.inex.otago.ac.nz/tracks/wiki-mine/wiki-mine.asp`

Table 1. Training and testing sets description

	XML Mining		RCV1 (topics)		RCV1 (regions)	
	train	test	train	test	train	test
Number of documents	10 978	43 654	23 149	781 265	23 149	781 265
Number of categories	39	39	101	103	228	296
Number of documents with only one label	9053	36 042	734	23 871	18 638	616 762
Average of number of categories per document	1.46	1.45	3.18	3.24	1.28	1.32
Variance of number of categories per document	3.15	2.96	1.84	1.98	0.53	0.65

subset of Wikipedia represents 39 overlapping categories, each corresponding to one subject or topic (for example: American civil war, Baseball, Politics, World War I). Each document belongs to at least one category. The training set is composed of about 20% of the collection which corresponds to 10 978 documents. On the training set, the mean of the number of categories per document is 1.46 and 82% of the documents belongs to only one category. Documents are rather long with 2103.47 words in average per document. In Table 1, we can see that the variance of the number of categories per document is important. Several Wikipedia documents can be considered as link pages and thus, they belong to a great number of categories.

The second collection is the well known Reuters Corpus Volume I (RCV1) collection[2] [12]. RCV1 collection is an archive of 800 000 manually categorized newswire stories made available by Reuters, Ltd. Documents are shorter than in the XML Mining collection with 75.73 words per document on average. Different subsets are defined in this collection (topics, regions and industries). The first one is the subset of topics where documents are assigned to 103 topic codes. On average, about 3 topics are associated with each document. The second subset is the subset of regions where documents are associated with 296 codes of region with about one region per document on average. The last one is the industries subset which categorizes documents into 350 codes of industry. Contrary to the two first subsets (topics and regions), in the subset of industries, some documents are not categorized and for this reason, experiments are performed using only the subsets of topics and regions. As shown in Table 1, contrary to the XML Mining collection, in the RCV1 collection, some categories are not represented in the training set. Moreover categories are organized in hierarchy.

5.2 Evaluation Criteria

In order to evaluate multi-label classification results, we used three common criteria : *Exact*, *Micro* and *Macro*. *Exact* is the exact match ratio which correspond to the proportion of correct predictions in the documents to classify.

[2] `http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm`

A document is well classified if and only if all associated labels are correctly predicted. This *Exact* criteria is very restrictive and is not necessarily the most suited to measure the performance of the classification as it does not take into account the partial matches. Thus, we also used the *Micro* and *Macro* criteria which are the micro and macro average of the F1-measure. These two criteria are, for example, used to evaluate performance in the XML Mining competition [6]. The F1-measure is a well-known criteria which corresponds to the harmonic mean of precision and recall.

5.3 Experimental Protocol

The documents belonging to the XML Mining collection are provided while only the TF.IDF description vectors are given for RCV1 collection. So, we will describe how we process the XML Mining collection before explaining how we perform the classification.

The first step of the definition of the TF.IDF description vectors of the XML Mining collection consists in a pre-processing of the documents. Indeed the original number of terms that appear in the documents is very important with 1 136 737 terms. In order to reduce the size of the index, we applied the Porter Algorithm [14]. We also remove a large number of irrelevant terms that could degrade the categorization, e.g.: the numbers (2311, -1224, 0d254c, etc.), terms that appear less than three times, or terms that appear in almost all the documents of the training set corpus. After their deletion, the index size is reduced to 295 721 terms on all the documents. All the documents are represented with a TF.IDF description vector representation presented in section 2.

The categorization is performed using a Support Vector Machines (SVM) classifier. SVM was introduced by Vapnik for solving pattern recognition problems using Structural Risk Minimization principal [21]. In our experiments, the multiclass SVM algorithm available in the Liblinear library [8] has been used (L2 loss support vector machine primal with default parameters).

In order to compare the different approaches, we first perform the classical binary categorization (denoted binary in the following sections) that consists in the training of a classifier for each category. As previously pointed out, this approach supposes that categories are independent of each other. Then, in considering all categories simultaneously, we apply a multiclass SVM providing a score for each document-category pair (d_i, c_k) and we compare the sets of categories $\hat{L}(d_i)$ attributed to the document d_i respectively by the $MCut$, $RCut$ and $PCut$ thresholding strategies. For the $RCut$ strategy, we perform 4 runs with values of t varying from 1 to 4. All obtained results are detailed in section 6.

6 Results

6.1 XML Mining Collection

Table 2 shows all the obtained results on the XML Mining collection. As we can see on this table, the binary run obtains the poorest results if we compare with

Table 2. Results for the XML Mining and RCV1 collection

	XML Mining			RCV1 (topics)			RCV1 (regions)		
	Exact	Micro	Macro	Exact	Micro	Macro	Exact	Micro	Macro
binary	0.330	0.402	0.324	**0.509**	**0.799**	**0.494**	0.746	**0.835**	0.407
$MCut$	0.524	**0.600**	**0.560**	0.443	0.627	0.306	**0.765**	0.670	0.289
$RCut_1$	**0.594**	0.597	0.536	0.030	0.444	0.179	0.756	0.823	0.433
$RCut_2$	0.037	0.535	0.515	0.193	0.671	0.378	0.101	0.683	**0.490**
$RCut_3$	0.006	0.460	0.454	0.325	0.750	0.482	0.017	0.554	0.464
$RCut_4$	0.002	0.401	0.398	0.041	0.710	0.491	0.003	0.461	0.423
$PCut$	0.438	0.544	0.513	0.378	0.655	0.355	0.722	0.774	0.363

those obtained with the $MCut$, $RCut_1$ and $PCut$ thresholding strategies. The $RCut_1$ strategy assigns only one category to documents and obtains the best result in considering the exact match ratio criteria. As we have seen on Table 1, 82% of documents have only one label. This is the reason why this thresholding strategy works well on this collection. The $MCut$ strategy slightly decreases the exact match ratio from 0.594 to 0.524 but also slightly improves the micro (respectively the macro) average F1-measure from 0.597 to 0.600 (respectively from 0.536 to 0.560). The $PCut$ strategy obtains quite good results while the $RCut_2$, $RCut_3$ and $RCut_4$ thresholding obtain poor results because the average of categories per document is 1.46 (Table 1).

6.2 RCV1 Collection (Topics)

Obtained results on the RCV1 collection for the topic codes, are summarized in Table 2. For the thresholding strategies, depending on the considered evaluation criteria, the $MCut$ and the $RCut$ strategies are better than the $PCut$ strategy. Indeed, the best results for the exact match ratio are obtained with the $MCut$ strategy, while the $RCut_3$ (resp. $RCut_4$) obtains the best results for the micro (resp. macro) average F1-measure. As shown by Table 1, the variance of the number of categories per document is low. This relative stability of the number of categories assigned to documents is consequently in favour of the $RCut$ thresholding strategy.

6.3 RCV1 Collection (Regions)

The results for the RCV1 collection with the subset of region codes are presented in Table 2. The binary run, as for the topics subset, obtains very good results but, with the exact match ratio criteria, the $MCut$ strategy has better results while the $RCut_2$ gets the best for the macro average F1-measure. The $RCut_1$ thresholding strategy is globally the best strategy according to the different criteria. As for the RCV1 topic codes, the $RCut$ strategy is favoured by the low variance of the number of categories per document as presented on Table 1.

6.4 Discussion

When categories are independent, learning binary classifiers for each category is a good strategy. However this hypothesis of independency is very restrictive and can not be considered for all collections. When it is not verified, thresholding strategies are good alternatives. The *RCut* thresholding strategy works well when the number of categories that must be assigned to each document is roughly the same over all the collection. In that case, the parameter that corresponds to the number of categories that must be assigned to document is easy to calculate. For instance, in our experiments $RCut_1$ gives good results on the XML Mining and on the RCV1 regions collections in which the percentage of single-label documents is important.

Concerning the *PCut* strategy, this solution is not suited for online applications as it is a category-pivoted strategy. Moreover, it can only provide good results when the categories' proportion stay the same in the training set and in the testing set.

Finally, the *MCut* thresholding strategy seems to be a interesting alternative to these existing algorithms that can be used in all cases. This is a well suited strategy when the number of categories is very different between documents. In the other cases, it obtains comparable results for the micro and macro average F1-measures with the other thresholding strategies and it often provides the best result in considering the exact match ratio. It offers also the advantage to be easy to apply.

7 Conclusion

The multi-label classification is a challenging problem. In this article, we focused our attention on thresholding strategies that aim to select a number of categories for each document. Comparatively to common thresholding strategies which need to select a value for the parameter, the *MCut* strategy, proposed in this article, is completely automatic. It does not have to be trained and it does not need any parametrization. Moreover, *MCut* strategy does not require assigning a same number of categories to each document as the *RCut* strategy and it does not suppose that the distribution of categories is known and stays stable like in the *PCut* strategy. In experiments done on several datasets, it provides good results in comparison with *RCut* and *PCut* even when the characteristics of the dataset were in favour of these last methods. For all these reasons, among the methods, simple to implement, the *MCut* strategy offers an interesting solution that can be used with large corpora that are currently available.

References

1. Clare, A., King, R.D.: Knowledge Discovery in Multi-label Phenotype Data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001)

2. Cohen, W.W., Singer, Y.: Context-sensitive learning methods for text categorization. In: Proceedings of the 19th ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval (SIGIR 1996), pp. 307–315 (1996)
3. De Comité, F., Gilleron, R., Tommasi, M.: Learning multi-label alternating decision trees from texts and data. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 251–274. Springer, Heidelberg (2003)
4. Crammer, K., Singer, Y., Jaz, K., Hofmann, T., Poggio, T., Shawe-taylor, J.: A family of additive online algorithms for category ranking. Journal of Machine Learning Research (JMLR) 3, 1025–1058 (2003)
5. Denoyer, L., Gallinari, P.: The wikipedia xml corpus. Special Interest Group on Information Retrieval Forum (SIGIR 2006) 40(1), 64–69 (2006)
6. Denoyer, L., Gallinari, P.: Report on the xml mining classification track at inex 2009. In: INitiative for the Evaluation of XML Retrieval 2009 Workshop Preproceedings (INEX 2009), pp. 339–343 (2009)
7. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 681–687 (2001)
8. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research (JMLR) 9, 1871–1874 (2008)
9. Har-Peled, S., Roth, D., Zimak, D.: Constraint Classification: A New Approach to Multiclass Classification. In: Cesa-Bianchi, N., Numao, M., Reischuk, R. (eds.) ALT 2002. LNCS (LNAI), vol. 2533, pp. 365–379. Springer, Heidelberg (2002)
10. Lewis, D.D.: An evaluation of phrasal and clustered representations on a text categorization task. In: Proceedings of the 15th ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval (SIGIR 1992), pp. 37–50 (1992)
11. Lewis, D.D., Ringuette, M.: A comparison of two learning algorithms for text categorization. In: Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR 1994), pp. 81–93 (1994)
12. Lewis, D.D., Yang, Y., Rose, T.G., Dietterich, G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research (JMLR) 5, 361–397 (2004)
13. Luo, X., Zincir-Heywood, A.N.: Evaluation of Two Systems on Multi-class Multi-label Document Classification. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 161–169. Springer, Heidelberg (2005)
14. Porter, M.: An algorithm for suffix stripping. Program 3, 130–137 (1980)
15. Montejo-Ráez, A., Ureña-López, L.A.: Selection Strategies for Multi-label Text Categorization. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 585–592. Springer, Heidelberg (2006)
16. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
17. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2-3), 135–168 (2000)
18. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
19. Tang, L., Rajan, S., Narayanan, V.K.: Large scale multi-label classification via metalabeler. In: Proceedings of the 18th International Conference on World Wide Web (WWW 2009), pp. 211–220 (2009)

20. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining (IJDWM 2007) 3(3), 1–13 (2007)
21. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1995)
22. Yang, Y.: A study of thresholding strategies for text categorization. In: Proceedings of the 24th ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval (SIGIR 2001), pp. 137–145 (2001)
23. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd ACM Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval (SIGIR 1999), pp. 42–49 (1999)
24. Zhang, M.-L., Zhou, Z.-H.: A k-nearest neighbor based algorithm for multi-label classification. In: Proceedings of the 1st IEEE International Conference on Granular Computing (GrC 2005), pp. 718–721 (2005)
25. Zhang, M.-L., Zhou, Z.-H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering (TKDE 2006) 18, 1338–1351 (2006)

Improving Tag Recommendation Using Few Associations

Matthijs van Leeuwen[1,2] and Diyah Puspitaningrum[1]

[1] Dept. of Information & Computing Sciences, Universiteit Utrecht, The Netherlands
[2] Dept. of Computer Science, KU Leuven, Belgium
matthijs.vanleeuwen@cs.kuleuven.be, diyah@cs.uu.nl

Abstract. Collaborative tagging services allow users to freely assign tags to resources. As the large majority of users enters only very few tags, good tag recommendation can vastly improve the usability of tags for techniques such as searching, indexing, and clustering. Previous research has shown that accurate recommendation can be achieved by using conditional probabilities computed from tag associations. The main problem, however, is that enormous amounts of associations are needed for optimal recommendation.

We argue and demonstrate that pattern selection techniques can improve tag recommendation by giving a very favourable balance between accuracy and computational demand. That is, few associations are chosen to act as information source for recommendation, providing high-quality recommendation and good scalability at the same time.

We provide a proof-of-concept using an off-the-shelf pattern selection method based on the Minimum Description Length principle. Experiments on data from Delicious, LastFM and YouTube show that our proposed methodology works well: applying pattern selection gives a very favourable trade-off between runtime and recommendation quality.

1 Introduction

Online collaborative tagging platforms allow users to store, share and discover resources to which tags can be freely assigned. Well-known examples include Delicious (bookmarks), Flickr (photos), LastFM (music), and YouTube (videos). Tags can be chosen completely freely, allowing both generic and very specific tags to be assigned. As a result, tags enable very powerful tools for e.g. clustering [2,8] and indexing [9] of resources.

Tagging systems also have their downsides though. Apart from linguistic problems such as ambiguity and the use of different languages, the most apparent problem is that the large majority of users assign only very few tags to resources. Sigurbjörnsson and Van Zwol [14] did an analysis of tag usage on photo sharing platform Flickr, and showed that 64% of all tagged photos are annotated with 3 or less tags. Obviously, this severely limits the usability of tags for the large majority of resources.

As a solution to this problem, methods for a wide range of tag recommendation tasks have been proposed. In this paper we consider the most generic

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 184–194, 2012.
© Springer-Verlag Berlin Heidelberg 2012

tag recommendation task possible. That is, we only assume a binary relation between resources and tags. Because it is so generic, it can be applied in many instances, e.g. on platforms where heterogeneous resource types co-exist, making it impossible to use resource-specific methods. Also, many collaborative tagging platforms maintain only a single set of tags per resource, irrespective of which user assigned which tag.

Sigurbjörnsson and Van Zwol [14] were the first to propose a recommendation method for this setting, based on pairwise tag co-occurrences in the tagsets previously assigned to resources. Using these existing tag assignments, referred to as collective knowledge, candidate tags are generated and subsequently ranked by (aggregated) conditional probabilities. LATRE, introduced by Menezes et al. [11], builds upon this by using larger sets of co-occurring tags, in the form of association rules, instead of only pairwise co-occurrences. To avoid mining and caching enormous amounts of associations, LATRE mines the needed rules for each query individually. The empirical evaluation showed that it is indeed beneficial to use tag associations consisting of more than just two tags.

The big problem, however, is that although 'on-demand' mining circumvents the need to cache millions of associations, the *online* mining of association rules is computationally very demanding. As such, LATRE provides better recommendations at the cost of scalability.

1.1 Aims and Contributions

Although recommendation using pairwise associations [14] is both accurate and very fast, exploiting associations with more than two tags can improve accuracy even further [11]. Unfortunately, this comes at the cost of memory space and computational complexity. In this paper, we demonstrate that the balance between accuracy and computational complexity can be improved by using a carefully selected *small set* of associations. To be more precise, we will show how pattern selection can positively contribute to association-based recommendation.

In Section 2, we will first provide the notation that we will use throughout the paper, and formally state the recommendation problem that we consider. After that, Section 3 will explain association-based recommendation in more detail. First, the above mentioned method using pairwise associations will be detailed. Second, both our own generalisation to larger associations and LATRE will be discussed. Third, we will introduce FastAR, for Fast Association-based Recommendation, which needs only few associations to achieve high accuracies, making it much faster than its competitors. For this, FastAR uses associations selected by the Krimp algorithm [13], a pattern selection method based on the Minimum Description Length principle. Given a database and a large set of patterns, it returns a set of patterns that together compresses the database well. These so-called *code tables* have been shown to provide very accurate descriptions of the data, which can be used for e.g. tag grouping [8].

After all methods have been introduced, they will be empirically compared in Section 4. Finally, we round up with related work and conclusions in Sections 5 and 6.

2 Tag Recommendation

We consider binary relations between a set of resources $\mathcal{S}$ and a set of tags $\mathcal{T}$, i.e. $\mathcal{S} \times \mathcal{T}$. That is, each tag can be assigned to any number of resources, and each resource can have any number of tags assigned. In the following, each resource is represented solely by its set of associated tags, simply dubbed a *transaction*.

Let database $\mathcal{D}$ be a bag of transactions over $\mathcal{T}$. A transaction $t \in \mathcal{D}$ is a subset $t \subseteq \mathcal{T}$, and $|\mathcal{D}|$ denotes the number of transactions in $\mathcal{D}$. A tagset X is a set of tags, i.e. $X \subseteq \mathcal{T}$, X occurs in a transaction t iff $X \subseteq t$, and the length of X is the number of tags it contains, denoted by $|X|$. Maximum length parameter *maxlen* restricts the number of tags a tagset X may contain, i.e. $|X| \leq maxlen$. The support of a tagset X in database $\mathcal{D}$, denoted by $sup_{\mathcal{D}}(X)$, is the number of transactions in $\mathcal{D}$ in which X occurs. That is, $sup_{\mathcal{D}}(X) = |\{t \in \mathcal{D} \mid X \subseteq t\}|$. A tagset X is *frequent* if its support exceeds a given minimum support threshold *minsup*, i.e. $sup_{\mathcal{D}}(X) \geq minsup$. Due to the A Priori property, all frequent tagsets can be mined efficiently [3].

2.1 The Problem

Informally, we consider the following tag recommendation task. Assume given a database $\mathcal{D}$ consisting of tagsets that have previously been assigned to resources. When a user assigns a new tagset I to a resource, the recommendation algorithm is invoked. Using source database $\mathcal{D}$ and query I as input, it recommends a tagset $R(\mathcal{D}, I) \subseteq (\mathcal{T} \setminus I)$ to the user. Unlike the use of the word 'set' may suggest, order is important; $R(\mathcal{D}, I)$ has to be ranked according to relevance to the user.

In practice, query I consists of very few tags in the large majority of cases, since most users do not manually specify many tags. Therefore, high-quality tag recommendation is of uttermost importance for resources annotated with 3 or fewer tags. All the more so, since these input sizes potentially benefit most from recommendation when more than 3 tags are given, recommendation is probably less necessary.

Apart from the quality of the recommendations, performance from a computational point of view is also important. Tag recommendation is often implemented in online web services and thus needs to be fast. Although we do not formalise this in the problem statement, we will evaluate this in the Experiment section. The tag recommendation problem can be stated as follows.

Problem 1 (Tag Recommendation). Given a source database $\mathcal{D}$ over tag vocabulary $\mathcal{T}$, and an input tagset I, recommend a set of tags $R(\mathcal{D}, I) \subseteq (\mathcal{T} \setminus I)$ ranked by relevance.

3 Association-Based Tag Recommendation

3.1 Pairwise Conditional Probabilities

Sigurbjörnsson and Van Zwol [14] were the first to propose a method for tag recommendation that uses only the tagsets previously assigned to resources,

which they referred to as collective knowledge. The method is based on pairwise tag co-occurrences in this collective knowledge. In short, given a query I, all tags co-occurring with an $i \in I$ are ranked based on their conditional probabilities.

In more detail, it works as follows. Given an input tagset I and source database $\mathcal{D}$, a candidate list C_i of the top-m most frequently co-occurring tags (with i) is constructed for each $i \in I$ (where m is a parameter). For each candidate tag $c \in C_i$, its empirical conditional probability is then given by

$$P(c \mid i) = \frac{sup_{\mathcal{D}}(\{c, i\})}{sup_{\mathcal{D}}(\{i\})}. \tag{1}$$

For any singleton input tagset $I = \{i\}$, the candidate tags in C_i are simply ranked according to $P(c \mid i)$ and then returned as recommendation. For any longer I, however, the rankings obtained for each individual input tag will have to be aggregated. The authors proposed and tested two different strategies for this, i.e. *voting* and *summing*, but we will focus on the latter as this was shown to perform better. First, a set C containing all candidate tags is constructed: $C = \bigcup_{i \in I} C_i$. Next, individual conditional probabilities are simply summed to obtain a score $s(c)$ for each candidate tag $c \in C$. This is given by

$$s(c) = \sum_{i \in I} P(c \mid i). \tag{2}$$

Finally, all candidate tags are ranked according to score function s, providing the final recommendation. In the original paper [14], the concept of *promotion* was introduced to weigh certain tags, which slightly improved recommendation quality. However, this requires parameter-tuning and to avoid unfair comparisons, we do not consider any tag weighing schemes in this paper.

The principle of using conditional probabilities is very strong, and maybe it is for that reason that very few improvements on this technique have been proposed since (considering only methods that assume exactly the same task). We will compare to this baseline method in Section 4. For this purpose, we will dub it PairAR (or PAR for short), for Pairwise Association-based Recommendation.

3.2 Many Associations

An obvious generalisation of PAR is to use not only *pairwise* associations between tags, but associations of *any length*. This may lead to better recommendation whenever the input tagset contains more than one tag, because we can compute more accurate empirical conditional probabilities. Suppose for example that we have an input tagset $I = \{x, y\}$ and a candidate tag z. With PAR we would compute $s(c) = P(z \mid x) + P(z \mid y)$ and use this for ranking, but with longer associations we could also compute $P(z \mid xy)$ and exploit this information.

The most naïve approach to do this would be to compute and use *all* associations of length $|I| + 1$, which would be sufficient to compute all needed conditional probabilities. Unfortunately, were we to cache all associations, this would come down to mining all frequent tagsets up to maximum length $|I| + 1$,

with a minimum support threshold of 1. In practice, this is infeasible due to the so-called pattern explosion, and larger *minsup*s are also problematic for realistically sized datasets. Even if it is at all possible to mine the desired associations, using all of them for recommendation is likely to be slow.

Nevertheless, we adopt this approach to see how it performs in practice and dub it NaiveAR (or NAR for short). First, it mines all frequent tagsets $\mathcal{F}$, with $maxlen = |I| + 1$ (hence a different set of associations for each input length is required) and a specified *minsup*. Additionally, the top-m co-occurrences per tag are computed as is done by the PairAR baseline. This to ensure that there are always candidate tags, even for rare input tags.

As candidates, all tags that co-occur with any of the input tags in any $F \in \mathcal{F}$ are considered, augmented with the top-m tags for each individual input tag. Conditional probabilities and scores $s(c)$, as defined in Equation 2, are computed as before, except that $\mathcal{F}$ is consulted for any conditional probabilities that cannot be obtained from the top-m's. There is a good reason for using the summing strategy of $s(c)$ despite having access to the 'exact' conditional probability $P(z \mid xy)$. That is, this exact probability is not always available due to the *minsup* parameter, and for those cases the summing strategy has the same effect as with PAR. Preliminary experiments showed this to be a good choice.

As discussed, LATRE, introduced by Menezes et al. [11], is a very similar method that also uses longer associations. To overcome the pattern explosion, LATRE uses on-demand mining of association rules for each individual query. Three parameters can be used to reduce the number of rules: 1) a maximum number of tags per rule, 2) minimum support, and 3) minimum confidence. Note that the main difference with NAR is that NAR uses a higher *minsup* to mine associations once beforehand and augments these with the top-m co-occurring tags for each input tag. We will empirically compare to LATRE in Section 4.

3.3 From Many to Few Associations

Although experiments will show that NaiveAR and LATRE can improve on PairAR in terms of precision, the downside is that both methods are rather slow. Unfortunately, most users would rather skip recommendations than wait for them. The question is therefore whether a small set of associations could already improve precision, such that recommendation is still (almost) instant.

That is, we are looking for a pattern set that provides a complete description of (the associations contained in) source database $\mathcal{D}$. In previous work [13,8] we have shown that *code tables*, such as produced by e.g. the heuristic algorithm Krimp [13], provide such descriptions.

A code table CT consists of two columns and describes a dataset by encoding it. The first column contains tagsets, while the second column contains their replacement codes. To encode a transaction t, CT is scanned for the first tagset $X \in CT$ such that $X \subseteq t$. X is then replaced by its code and the encoding continues for $t \setminus X$ until $t = \emptyset$. Since a code table contains at least the singleton tagsets, the encoding of a transaction always succeeds.

Analogue to NaiveAR, the set of tagsets provided by the first column of a code table is augmented with the top-m pairwise co-occurrences for each input tag. Also, the set of candidate tags is obtained in exactly the same way: each tag that co-occurs with any of the input tags in the code table is considered a candidate tag, as are the top-m tags for each input tag.

Although the code table does not store tagset supports, we can infer information from its codes. The actual codes used are immaterial to us here. What is important is their lengths. While encoding $\mathcal{D}$ with CT we maintain a list that records how often each $X \in CT$ is used in encoding, which is called the *usage* of X, denoted by $usage_{\mathcal{D}}(X)$. The code for X is chosen such that its length is

$$-\log\left(\frac{usage_{\mathcal{D}}(X)}{\sum_{Y \in CT} usage_{\mathcal{D}}(Y)}\right).$$

The Minimum Description Length (MDL) principle states that the best code table is the one that compresses $\mathcal{D}$ best. Given this optimal code table CT, one can compute a tagset's support by summing the usages of all code table elements that are a superset of the tagset. Unfortunately, computing the optimal code table is infeasible. However, from previous research we know that the code tables computed by the heuristic algorithm Krimp are rather accurate. Hence, we estimate the support of any tagset X by

$$\hat{sup}_D(X) = \sum_{Y \in CT, X \subseteq Y} usage_{\mathcal{D}}(Y),$$

which can be easily computed from the code table.

Again, computing the scores for the candidate tags is done as by PAR and NAR, except that code table CT is consulted for any conditional probabilities that cannot be obtained from the top-m's. In these cases, the support of a tagset X is estimated by $\hat{sup}_{\mathcal{D}}(X)$.

Note that the code table itself can be computed offline. While computing recommendations, we only need to estimate supports. Given that a code table is significantly smaller than the set of all (frequent) tagsets, this algorithm should be much faster than both NAR and LATRE. We therefore dub it FastAR (or FAR for short), for Fast Association-based Recommendation.

4 Experiments

In this section we evaluate PairAR, NaiveAR, FastAR, and LATRE on three datasets collected from online collaborative tagging services.

Evaluation Criteria. To assess recommendation quality, we resort to the most commonly taken approach, i.e. 5-fold cross-validation. The dataset is partitioned into 5 equal-sized parts and for each of 5 folds, four parts are concatenated into a training database D^{train} and the remaining part D^{test} is used for testing. All results are averaged over all 5 folds (except for runtime, which is averaged per query).

D^{train} is used as source database $\mathcal{D}$ for all methods. From each tagset $X \in D^{\text{test}}$, a query $I \subset X$ is randomly selected. The remaining tags in the tagset, $X \setminus I$,

Table 1. Datasets. The number of transactions and distinct tags, as well as average and maximum transaction lengths are given. $|\mathcal{D}^k|$ represents the number of transactions used for testing after removing transactions that are too short, for $k \in \{2, 3\}$.

Dataset	*Properties*				*Effective test data*	
	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	avg($\|t\|$)	max($\|t\|$)	$\|\mathcal{D}^2\|$	$\|\mathcal{D}^3\|$
Delicious	526,490	19,037	9.6	30	338,284	308,832
LastFM	91,325	7,380	20.3	152	61,701	56,904
YouTube	169,290	7,785	8.3	74	97,591	83,450

are used for validation. The size of query I is parametrised by k and fixed for a given experiment, s.t. $k = |I|$. Since small queries are most prominent in real world situations, we consider $k \in \{2, 3\}$. Note that we do not consider $k = 1$, because the proposed methods are equivalent to PairAR for this setting. To ensure that there are always at least five validation tags that can be used to measure precision, we exclude all transactions that contain less than $k + 5$ tags from $\mathcal{D}^{\text{test}}$.

As evaluation criteria, we use precision, mean reciprocal rank and runtime, denoted by P@x, MRR and time. P@x denotes precision of the x highest ranked tags, with $x \in \{1, 3, 5\}$, and is defined as the average percentage of the first x recommended tags that occur in validation tags $X \setminus I$. Mean reciprocal rank represents the average rank of the first correctly recommended tag. It is defined as $MRR = \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{X \in \mathcal{D}^{\text{test}}} \frac{1}{first(X)}$, where $first(X)$ is the rank of the first correctly recommended tag for test case X.

To quantify computational demand we measure runtime, which reflects the time needed for the *online* computation of a recommendation for a single query (on average). Note that this excludes the time needed for generating the associations needed by PAR/NAR/FAR, as this can be done *offline* beforehand.

Datasets. We use the pre-processed datasets crawled from Delicious[1], YouTube[2], and LastFM[3] by Menezes et al. [11]. Some basic properties are presented in Table 1; see their paper for more details on the collection and pre-processing. Note that we experiment on the complete datasets, which contrasts their setup.

PAR, NAR and FAR. In all cases, $m = 50$, i.e. the top-50 most frequent co-occurring tags for each tag are collected. For NAR, tagset collection $\mathcal{F}$ is obtained by mining all closed frequent tagsets with *minsup* as given in Table 2 and $maxlen = k+1$ (as was already specified in the previous section). For FAR, Krimp is used to obtain a code table CT from all frequent closed tagsets for the specified *minsup* (no *maxlen*).

LATRE. For comparison to LATRE, we use the original implementation and the same settings as in the original paper: *minsup* = 1, *minimum confidence* = 0.00001, and $\alpha_{max} = 3$ (implying that association rules with antecedent up to the length 3 are allowed). Note that strictly speaking we compare to LATNC,

[1] `www.delicious.com`
[2] `www.youtube.com`
[3] `www.last.fm`

a variant of LATRE without 'calibration', a tag weighing scheme like PAR's promotion. We decided to refrain from such schemes to keep comparison fair. Promotion and/or calibration could be applied to all methods described in Section 3, but require fine-tuning.

Implementation. Prototypes of PAR, NAR and FAR were implemented in C++. Each experiment was run single-threaded on a machine with a quad-core Intel Xeon 3.0GHz CPU and 8Gb RAM running Windows Server 2003.

4.1 Results

A complete overview of the results is presented in Table 2. PairAR performs quite well in terms of precision and it is extremely fast, requiring up to 5 seconds for the online recommendation phase. We chose the *minsup*s for NaiveAR such that the number of resulting tagsets in $\mathcal{F}$ was in the order of the 100 000s; these amounts of tagsets could be kept in memory and recommendation was computationally feasible. Looking at the precision of the highest ranked recommended tags, we observe that NAR provides improvements up to 2.71% over PAR. Only for LastFM with $k = 3$ precision slightly decreases. Due to the large numbers of tagsets to be considered, runtimes increase up to 428.91 milliseconds per query.

To demonstrate the effect of pattern selection, we ran FastAR with the same *minsup*s as NAR. This shows that Krimp significantly reduces the number of tagsets. For LastFM with $k = 2$, for example, $\mathcal{F}$ contains 540 697 tagsets, whereas the code table used by FAR contains only 9 513 tagsets. With this decrease in the number of tagsets, runtime is reduced from 1229.32 to only 2.16 milliseconds per query, while an –admittedly small– increase in precision with respect to PAR is still attained. When we consider YouTube with $k = 2$ (or similarly $k = 3$), running NAR with a *minsup* lower than 13 was not feasible, while FAR could easily be applied with a *minsup* of 1. This resulted in a code table consisting of only 14 981 tagsets. Despite the modest number of tagsets, this gave an increase in P@1 of 4.73%, which is substantially better than the 2.71% improvement obtained by NAR.

The relative performance of LATNC varies from setting to setting: precisions and MRR are subpar for Delicious, for LastFM with $k = 2$ and for YouTube with $k = 2$. For LastFM with $k = 3$, the scores are better than those of any other method, and LATNC is only beaten by FAR for YouTube with $k = 3$. LATNC is always much slower than FAR though.

4.2 Analysis

The overall good performance of PairAR shows that the idea of summing pairwise conditional probabilities is hard to beat. It very much depends on the data at hand whether associations consisting of multiple tags can be exploited to improve recommendation quality. This is not the case for the Delicious dataset that we used, which is probably due to the relative large number of tags, of which many occur only very rarely. For the YouTube data, however, recommendation can be substantially improved using FastAR. That is, for this dataset

Table 2. Results. For each combination of dataset, query size k, method, and *minsup*, the obtained precisions, MRR and runtime are given. For NaiveAR and FastAR, the number of tagsets in $\mathcal{F}$ resp. CT are given. Runtime is given per query, on average in milliseconds.

Dataset	k	method	*minsup*	#tagsets	P@1	P@3	P@5	MRR	time (ms)
Delicious	2	PAR	-		45.93	35.19	29.20	56.95	0.01
		NAR	37	142,185	46.80	35.79	29.64	55.68	59.07
		FAR	37	25,736	46.28	35.42	29.36	55.26	6.89
		LATNC	1		39.39	29.82	24.47	50.34	129.88
Delicious	3	PAR	-		51.13	39.27	32.53	62.35	0.02
		NAR	37	219,832	51.33	39.35	32.60	60.59	428.91
		FAR	37	25,736	50.86	39.18	32.54	60.43	11.83
		LATNC	1		49.14	37.98	31.60	59.77	606.06
LastFM	2	PAR	-		52.18	41.38	35.11	63.02	0.03
		NAR	51	540,697	53.49	42.60	36.18	62.41	1229.32
		FAR	51	9,513	52.70	42.04	35.78	61.63	2.16
		LATNC	1		46.36	37.75	32.17	58.43	156.41
LastFM	3	PAR	-		52.59	42.40	36.31	64.22	0.05
		NAR	146	132,103	52.16	41.97	36.11	61.88	415.44
		FAR	146	5,111	52.46	42.84	36.86	62.59	1.02
		FAR	51	9,513	52.91	43.27	37.41	62.90	4.20
		LATNC	1		55.42	45.15	38.83	66.49	596.18
YouTube	2	PAR	-		50.10	39.86	34.05	58.69	0.04
		NAR	13	324,696	52.81	42.48	36.29	59.03	300.26
		FAR	13	10,159	52.12	41.81	35.73	58.34	3.30
		FAR	1	14,981	54.84	44.86	38.78	60.46	19.08
		LATNC	1		38.86	31.00	26.54	48.29	63.50
YouTube	3	PAR	-		54.90	43.83	37.42	63.53	0.06
		NAR	40	183,228	55.73	44.32	37.66	62.17	562.79
		FAR	40	4,933	55.63	44.37	37.72	62.29	1.69
		FAR	1	14,981	59.20	48.44	41.83	64.87	45.37
		LATNC	1		55.81	46.20	40.15	63.71	150.68

our method based on pattern selection beats the more 'exhaustive' methods with respect to both precision and runtime.

Whenever NAR and/or LATNC achieve higher precisions than PAR, FAR provides a much better trade-off between precision and runtime. The downside is that it does not always give the highest possible precision. Mining all association rules on demand, such as LATNC does, does not seem to be a good idea for realistically sized datasets, as this comes at the cost of long runtimes.

Using pattern selection, on the other hand, comes at the cost of longer offline pre-processing times. Krimp requires up to 15 minutes to obtain code tables for Delicious and YouTube, and up to 10 hours for LastFM. However, this is mostly due to the fact that it needs to generate and test a large set of candidates, which

could be avoided by using different heuristics to construct code tables [15]. In the end, performing pattern selection once beforehand is well worth the effort if this results in much faster yet high-quality recommendation.

Finally, note that we here focused on relatively small query sizes (k) because these are both realistic and the most useful. However, exploiting longer associations can clearly be even more beneficial when k grows. This explains why LATRE performed much better than PAR in Menezes et al. [11].

5 Related Work

Many variants of tag recommendation have been studied in recent years [1]. Based on the used information, we split existing methods into three categories.

The first category is the most generic, and the one we consider in this paper: methods that only assume a binary relation between resources and tags. We already introduced and experimented with the methods in this category [14,11].

The second category aims at tagging systems that allow users to individually assign their own tagset to each resource. The resulting ternary relation is often called a *folksonomy*. One of the first methods for folksonomies was Folkrank [5], based on the same principles as PageRank, and outperforms baseline approaches like collaborative filtering [6]. Also, a system composed of several recommenders that adapts to new posts and tunes its own parameters was proposed [10].

The third category is characterised by its use of resource-specific information. These systems use the resources themselves to improve recommendation. Examples include methods specifically designed for documents [9,16], web pages [4], YouTube videos [17], and songs [7]. Finally, Rae et al. proposed to use social network information to improve tag recommendation [12].

Note that we cannot compare to methods from the latter two categories, as they all make additional assumptions about the recommendation task.

6 Conclusions

Conditional probabilities based on pairwise associations allow for high-quality tag recommendation, and this can be further improved by exploiting associations between more than two tags. Unfortunately, doing this naïvely is infeasible in realistic settings, due to the enormous amounts of associations in tag data.

To overcome this problem, we propose to use the strengths of pattern selection. Using an off-the-shelf pattern selection method based on the Minimum Description Length principle, we have demonstrated that our FastAR method gives a very favourable trade-off between runtime and recommendation quality.

FastAR uses Krimp, an existing pattern selection technique, to pick a small set of tagsets. However, the used coding scheme is not specifically designed for selecting associations that are useful for recommendation. Despite this, the presented results are encouraging. By modifying the selection process to better reflect the needs of tag recommendation, e.g. by allowing overlapping tagsets, we believe that the results can be further improved.

Acknowledgments. The authors would like to thank Adriano Veloso for kindly providing the datasets and LATNC implementation. This research is financially supported by the Ministry of Communication and Information Technology of the Republic of Indonesia, and by the Netherlands Organisation for Scientific Research (NWO) under project number 612.065.822 and a Rubicon grant.

References

1. Balby Marinho, L., Hotho, A., Jäschke, R., Nanopoulos, A., Rendle, S., Schmidt-Thieme, L., Stumme, G., Symeonidis, P.: Recommender Systems for Social Tagging Systems. Springer (February 2012)
2. Begelman, G.: Automated tag clustering: Improving search and exploration in the tag space. In: Proc of the WWW 2006 (2006)
3. Han, J., Pei, J.: Mining frequent patterns by pattern-growth: methodology and implications. SIGKDD Explorations Newsletter 2(2), 14–20 (2000)
4. Heymann, P., Ramage, D., Garcia-Molina, H.: Social tag prediction. In: Proc. of the SIGIR 2008, pp. 531–538 (2008)
5. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
6. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag Recommendations in Folksonomies. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 506–514. Springer, Heidelberg (2007)
7. Law, E., Settles, B., Mitchell, T.: Learning to Tag from Open Vocabulary Labels. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 211–226. Springer, Heidelberg (2010)
8. van Leeuwen, M., Bonchi, F., Sigurbjörnsson, B., Siebes, A.: Compressing tags to find interesting media groups. In: Proc of the CIKM 2009, pp. 1147–1156 (2009)
9. Li, X., Guo, L., Zhao, Y.E.: Tag-based social interest discovery. In: Proc. of the WWW 2008, pp. 675–684 (2008)
10. Lipczak, M., Milios, E.E.: Learning in efficient tag recommendation. In: Proc. of the RecSys 2010, pp. 167–174 (2010)
11. Menezes, G.V., Almeida, J.M., Belém, F., Gonçalves, M.A., Lacerda, A., de Moura, E.S., Pappa, G.L., Veloso, A., Ziviani, N.: Demand-Driven Tag Recommendation. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 402–417. Springer, Heidelberg (2010)
12. Rae, A., Sigurbjörnsson, B., van Zwol, R.: Improving tag recommendation using social networks. In: Proc of the RIAO 2010, pp. 92–99 (2010)
13. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: Proc. of the SDM 2006, pp. 393–404 (2006)
14. Sigurbjörnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: WWW, pp. 327–336 (2008)
15. Smets, K., Vreeken, J.: Slim: Directly mining descriptive patterns. In: Proc. of the SDM 2012 (2012)
16. Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W.-C., Lee Giles, C.: Real-time automatic tag recommendation. In: Proc of the SIGIR 2008, pp. 515–522 (2008)
17. Toderici, G., Aradhye, H., Pasca, M., Sbaiz, L., Yagnik, J.: Finding meaning on youtube: Tag recommendation and category discovery. In: CVPR, pp. 3447–3454 (2010)

Identifying Anomalous Social Contexts from Mobile Proximity Data Using Binomial Mixture Models

Eric Malmi, Juha Raitio, Oskar Kohonen,
Krista Lagus, and Timo Honkela

Department of Information and Computer Science,
Aalto University, FI-00076 AALTO, Finland
{eric.malmi,juha.raitio,oskar.kohonen,krista.lagus,
timo.honkela}@aalto.fi

Abstract. Mobile proximity information provides a rich and detailed view into the social interactions of mobile phone users, allowing novel empirical studies of human behavior and context-aware applications. In this study, we apply a statistical anomaly detection method based on multivariate binomial mixture models to mobile proximity data from 106 users. The method detects days when a person's social context is unexpected, and it provides a clustering of days based on the contexts. We present a detailed analysis regarding one user, identifying days with anomalous contexts, and potential reasons for the anomalies. We also study the overall anomalousness of people's social contexts. This analysis reveals a clear weekly oscillation in the predictability of the contexts and a weekend-like behavior on public holidays.

Keywords: anomaly detection, social context, mobile proximity data, mixture models.

1 Introduction

Smartphones collect an unprecedented amount of objective information about their users' surroundings, providing a rich source of information about the users' behavior and social interactions. *Reality mining* refers to the collection and analysis of this kind of data [1]. We focus on mining Bluetooth (BT) data which serves as an indicator of the social context since it tells about the people and devices nearby which a user is potentially interacting with.

Modeling social contexts allows us to detect occasions when the user is in an abnormal social environment. The detection of this kind of anomalous events is commonly referred to as *anomaly detection* (also *outlier* or *novelty detection*). Detection of simultaneous anomalous contexts for different persons can reveal interesting collective phenomena, such as demonstrations or traffic jams. On the individual level, we may discover surprising events, such as a person getting into an accident or a mobile phone being stolen. All this contributes to the building of context-aware applications that may help us in unpredictable situations.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 195–206, 2012.
© Springer-Verlag Berlin Heidelberg 2012

For the anomaly detection, we take a statistical approach using multivariate binomial mixture models. Our objective is to address the following questions

- On which days is a user in an anomalous social context and what specifically makes this context anomalous?
- Which days are similar so that they can be clustered together?

In the next subsection, we discuss the related work. In Section 2, we describe the data and the preprocessing steps. The anomaly detection algorithm is presented in Section 3 and the experimental results in Section 4. Finally, we draw conclusions in Section 5.

1.1 Related Work

Univariate binomial mixtures have been used for density estimation, e.g., by Snipen et al. [2]. In statistical anomaly detection, mixture models have been used, e.g., for new physics searches [3], network intrusion detection [4] and for medical signal processing [5]. In the context of mobile data, anomaly detection problems have been studied by Zhang et al. [6] where the authors study anomalous collective call patterns using a method based on Voronoi diagrams. However, their study only considers univariate data. The same data set we use is analyzed in the work by Do and Gatica-Perez [7] where the authors propose a probabilistic method for discovering interaction types in BT data. Unlike their method, we model all devices in a scan at once rather than individual links between devices.

2 Data Set

The data set we analyze contains Bluetooth records from 115 users who participated the Lausanne data collection campaign [8] which spanned a period over two years. During the campaign, other records, such as GPS and application usage were also collected. However, in this work, we focus on analyzing the Bluetooth records since they carry information about the user's social context and how it changes over time.

The Bluetooth records describe scans made by the smartphones in intervals ranging typically from one to three minutes. The observed BT devices are other people's cell phones and other devices equipped with BT. For each user, there is a list of scans made by the user's smartphone. Each scan contains a time stamp and a list of MAC addresses of the observed devices. Some data is missing as people occasionally switch off their phones. More specifically, there may be longer than three minute gaps between the scans if the user has switched off his or her own smartphone or alternatively, a scan might be lacking some of the devices that are present if their owners have switched off the devices.

The Lausanne data set has similarities to the extensively analyzed Reality Mining data set [1]. In comparison, the Lausanne data set has a more heterogeneous sample of participants and it extends over a longer period [8]. Before analyzing the data, some preprocessing steps described in Section 2.1 are applied giving us daily counts of observed devices.

2.1 Preprocessing

We decided to study only the data records between October 10, 2009 and March 26, 2011, a total of more than 17 months, since outside this interval the number of active participants was very small. Furthermore, only the users with Bluetooth scans from at least 90 days within the time period are included. This leaves us with a total of 106 out of 115 users.

Our objective is to model social contexts. We define a context as device counts which are given by the following *bag-of-devices* representation

$$x_{i,t}^{j} : \text{the number of times user } j \text{ has observed device } i \text{ during day } t$$

The device indices i are unique to each user and the devices that the user has observed fewer than 20 times are ignored. This operation exludes a majority, i.e. typically several thousand devices the user has observed at least once thus making the computations faster and the estimation of probabilities more reliable.

To make the device counts comparable, we would like to have a constant number of scans being made during a day for each day and each user. To achieve this, we thin out some of the observations so that there are observations at maximum every three minutes. This gives us a total of $\frac{60(minutes/hour)}{3(minutes/observation)} \times 24(hours/day) = 480$ observations per day.

Nevertheless, the number of observations per day varies since the users may keep their phones switched off parts of the day, e.g., during the night. Therefore, we make the final preprocessing step where we accept only the days with at least 100 scans. The rejected days are regarded as missing values.

To make the comparison of likelihoods easier later on, we make the following assumption for the accepted days.

Assumption 1. *A daily count always consist of 480 scans. If in reality less scans were made during the day, we assume that no devices would have been observed in the missing scans.*

With the data selection principles described above, most of the missing scans are due to the phone being switched off during the night, which makes this assumption fairly realistic.

3 Methods

3.1 Anomaly Detection

An anomaly refers to "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [9]. In this work, we study an unsupervised problem setting where we do not have labels for observations indicating whether they are *normal* or *anomalous*. Therefore, we use the statistical definition of anomalies as observations that occur considerably less frequently than normal ones [9]. In practice, this means that if some observation, perhaps even a very peculiar one in the real world,

starts to occur often, it is by definition no longer an anomaly. This accords with the psychology's statistical definition of abnormal behavior as something that occurs "infrequently in relation to the behavior of the general population" [10].

In practice, we model the mechanism behind the normal observations with a parametric (probability) density function $p(x|\Theta)$, where x is an observation and Θ are the parameters. Then, following the statistical definition of anomalies, we say that observations occurring in low density regions are anomalies. Any new observation is fed to the estimated density function whose output, which is called evidence or simply likelihood, determines the anomalousness of the observation. A low likelihood indicates a highly anomalous sample. In case we had some labeled samples, we could use them to determine a threshold below which all samples would be labeled as anomalies. However, in our unsupervised setting we can only order the social contexts of different days according to their anomalousness and then, e.g., examine the most anomalous contexts.

3.2 Binomial Mixture Models

Mixture models provide a flexible way of modeling complex data (see e.g. [11]). We use a mixture of multivariate binomial distributions to model the density of the daily device count data.

A univariate binomial distribution $p(x|n,\theta) = \binom{n}{x}\theta^x(1-\theta)^{n-x}$ represents the probability distribution of the number of "successes" in n i.i.d. Bernoulli trials with parameter θ as the success probability. In our case, parameter $n = 480$ can be kept fixed based on the Assumption 1. We denote the number of observed devices during one day for n_d different devices as $\mathbf{x} = [x_1, x_2, \cdots, x_{n_d}]^T$. These counts can be modeled with a multivariate binomial distribution $p(\mathbf{x}|n,\theta_1,\cdots,\theta_{n_d}) = \prod_{i=1}^{n_d} p(x_i|n,\theta_i)$ assuming that the occurrences of the devices are independent. However, the independence assumption does not seem realistic in our case; e.g, there is probably a correlation between observing the phones of two colleagues. To account for some of the correlations, we take a mixture of these multivariate binomial distributions (*binomial mixture model*). The binomials are referred to as *components* of the mixture model and the resulting density function is given by a weighted sum of k components

$$p(\mathbf{x}|n,\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_k,\pi_1,\cdots,\pi_k) = \sum_{c=1}^{k} \pi_c \prod_{i=1}^{n_d} p(x_i|n,\theta_{c,i}), \tag{1}$$

where $\boldsymbol{\theta}_c = [\theta_{c,1},\theta_{c,2},\cdots,\theta_{c,n_d}]$ and π_c are the weights for which it holds that $\pi_c \geq 0,\ 1 \leq c \leq k$ and $\sum_{c=1}^{k} \pi_c = 1$.

Maximum likelihood estimates for the model parameters are learnt using the *Expectation Maximization* (EM) algorithm [12]. The EM updates for the binomial mixture models are adapted according to Banerjee et al. [13].

In addition to density estimation, mixture models can be used for clustering [11]. The idea is that each component of the mixture represents a cluster.

The clustering is achieved by giving sample $\mathbf{x}$ label l according to the cluster C with the highest posterior probability

$$l = \arg\max_C p(C|\mathbf{x}) = \arg\max_C \pi_c p(\mathbf{x}|\boldsymbol{\theta}_C, n). \quad (2)$$

4 Experimental Results

4.1 Model Selection

The number of components k in the mixture model is a free parameter. In order to avoid overfitting, we conducted several 5-fold cross-validations [14] varying k between 1 and 35. Due to a random initialization of the components, we ran ten restarts of the EM algorithm for each training of the model and always chose the model that gave the best training log-likelihood. We chose to have the same number of components for all users since manually selecting the optimal number of components for 106 users would have been laborious. In Figure 1, we have the validation log-likelihoods for five randomly selected users.

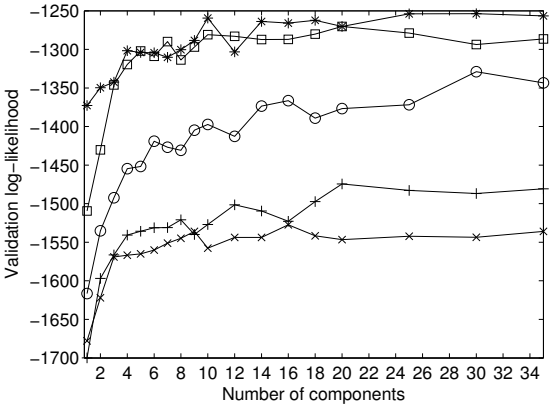


Fig. 1. Validation log-likelihoods for five randomly selected users

The validation likelihoods seem to stabilize around twenty components. Thus we selected twenty as the number of components in the model. However, for the analysis of a single user presented in Sections 4.2 and 4.3, we used only six components in order to be able to visualize the components.

4.2 Clustering of Weekdays

We selected user 62 to illustrate the clustering properties of the mixture model. User 62 was chosen as she or he was among the few who had only a couple of days with no BT scans being made. A binomial mixture model with six components

was learnt and each day was clustered into one of the six components according to Equation 2.

In Figure 2, we have plotted the weekday distribution of days in different clusters. On top of each subfigure, we have the number of samples n falling into the cluster and the average number of device observations being made per day in the cluster, defined as $d = \frac{1}{n} \sum_{t \in c} \sum_{i=1}^{n_d} x_{i,t}$, where $\sum_{t \in c}$ is the summation over days that get clustered into cluster c.

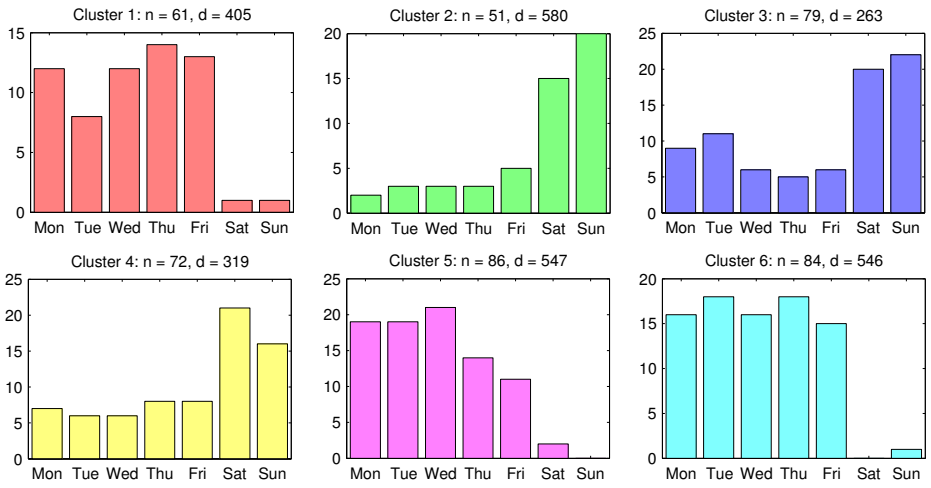


Fig. 2. Clustering of the daily Bluetooth observations (social contexts) for user 62 based on six binomial components. The histograms show how observations in a cluster are distributed over different weekdays. n is the number of samples in the cluster and d the average number of device observations being made per day in the cluster rounded to the nearest integer.

We notice that clusters 2, 3 and 4 capture mainly weekends. The difference between the weekend clusters is that cluster 2 describes an active weekend ($d = 580$) whereas clusters 3 and 4 describe quieter weekends ($d = 263$ and $d = 319$, respectively). Clusters 1, 5 and 6 capture mostly working days. In general, working days seem to be more active than weekends which is natural as people go to work or to university where they meet several colleagues and other students. In Section 4.3, we show how individual days of user 62 fall into these six clusters.

4.3 Measuring Anomalousness: Likelihood over Time

Single User. We used the same model, described in Section 4.2, to calculate the log-likelihood of each day for user 62 from October 9, 2009 to March 25, 2011. The resulting likelihood curve is shown in Figure 3. A low likelihood indicates high anomalousness. The figure also shows the clusters in which the days have fallen to. The clusters have been colored and numbered according to Figure 2.

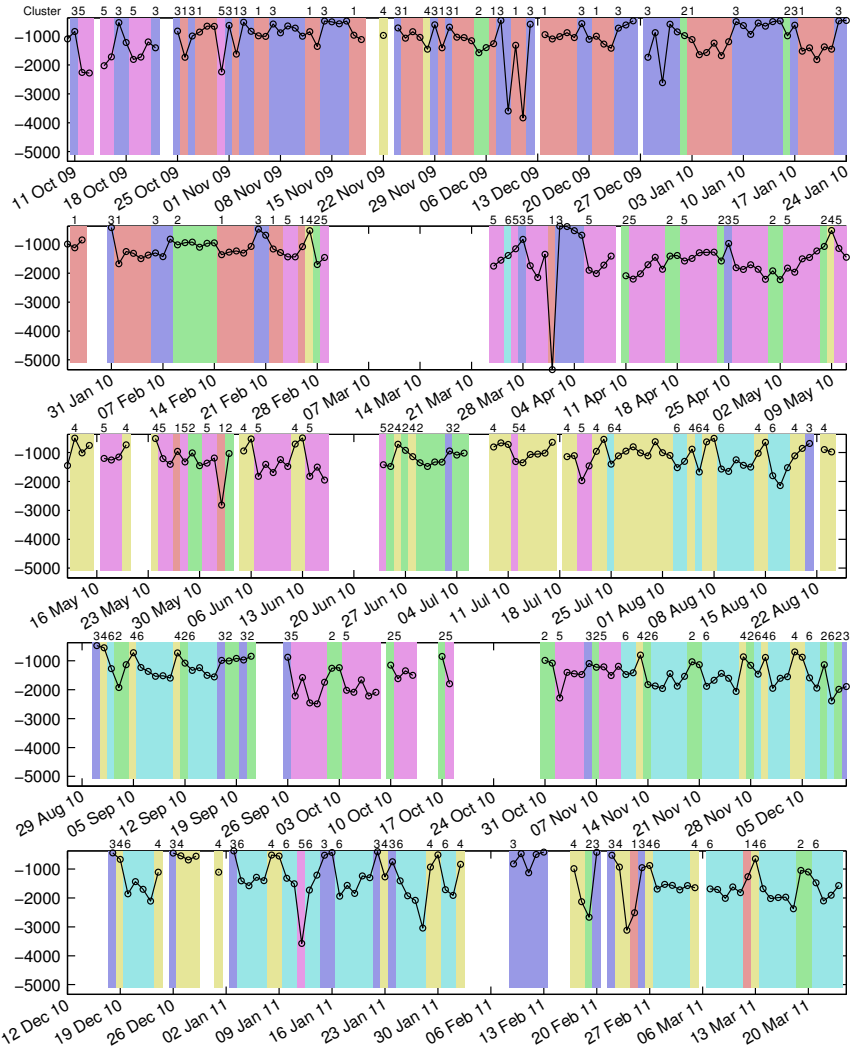


Fig. 3. Log-likelihoods of user 62's social contexts. Colors and the numbers on the top denote the clusters from Figure 2 to which the days belong. Tick marks are placed on Sundays.

The first impression regarding the clusters is that the user's social environment evolves over time and the clusters can be used to segment the timeline into periods dominated by different types of social contexts. Cluster 1 (red) is active until February 28, 2010 after which cluster 5 (violet) replaces it. The end of the timeline starting from August 3, 2010 is dominated by cluster 6 (turquoise) except for a period from September 27 to November 10 in 2010 when cluster 5 reappears. Clusters 2 (green), 3 (blue) and 4 (yellow) describe the weekends

throughout the timeline but they also capture public holidays such as Easter (from the 2nd to the 5th of April), Christmas and New Year.

In the likelihood curve we can distinguish several pits. The largest pit is on April 1, 2010 which is April Fools' Day. Let us take a closer look at this anomaly. To study which devices might be causing the anomaly, we calculate the marginal probability of each device count $p(x_i|n, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_k, \pi_c, \cdots, \pi_k) = \sum_{c=1}^{k} \pi_c p(x_i|n, \theta_{c,i})$. If the marginal probability is very low, it means that none of the components describes this single device count well. Table 1 displays the four lowest marginal log-probabilities and the corresponding device counts x_i.

Table 1. Four devices with the lowest marginal log-probabilities on April 1, 2010 which was the most anomalous day for user 62. $\log p(x_i)$ is the marginal log-probability and x_i is the observed device count.

Device (i)	5	4	2	3
$\log p(x_i)$	-17.4	-16.5	-16.0	-8.4
x_i	72	143	214	0

The marginal distributions for devices 5, 4, 2 and 3 are shown in Figure 4 where the circles denote the actual counts observed x_i.

Based on these results, we may say that part of the anomaly of April 1, 2010 for user 62 comes from the finding that devices 5, 4 and 2 have been observed exceptionally many times and device 3 exceptionally few times. However, it must be noted that part of the anomaly may result from unusual co-occurences: two device counts x_i and x_j may be normal if we look at their marginal probabilities $p(x_i)$ and $p(x_j)$ but their joint probability $p(x_i, x_j)$ may still be low if these kind of counts are never observed together. For example, i and j could be the devices of person's divorced parents. It can be normal to spend a whole day with either of them but the person rarely spends a whole day with both of them. Correlations of these kind may be captured by the mixture model but they are not shown in the marginal distributions.

All Users. We also studied the overall likelihood curve for the whole population. A model with twenty components was trained for each user separately. Each training consisted of ten restarts of the EM algorithm and the best of the ten runs was chosen as the model. To obtain an overall likelihood, we took averages of the log-likelihoods over the users[1]. The resulting overall likelihood curve is shown in Figure 5. A high likelihood means that people have been in typical social contexts whereas a low likelihood indicates an anomalous day during which the numbers of encounters have been unusually high or low.

[1] Normally, one would take the sum of log-likelihoods, which corresponds to the product of likelihoods but we chose to take the average instead since the number of active users varies from day to day. Thus the sum would depend on the number of users, whereas the average does not. The average corresponds to the harmonic mean of likelihoods.

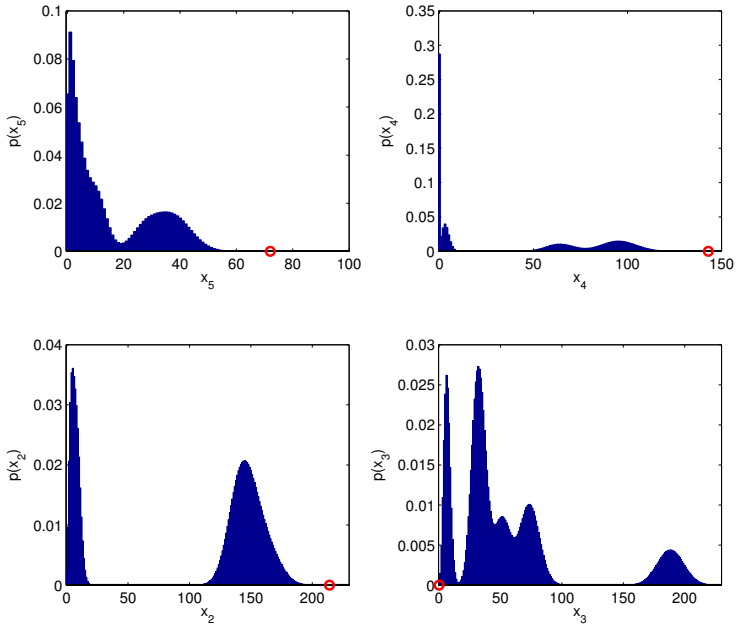


Fig. 4. Marginal distributions for the counts of devices 5, 4, 2 and 3 according to the model. The circles denote the actual counts observed on April 1, 2010 by user 62.

The log-likelihood of Figure 5 has an oscillation period of one week: Saturdays and Sundays have typically higher likelihoods than working days except for a few exceptions. The reason could be that on working days there are typically more devices present (see Section 4.2) and more variation in the encountered devices as the users go to university or to work encountering many people. On the other hand, on weekends there are typically only a few devices present, perhaps a family, meaning that most of the device counts are zero and easy to predict. The most easily distinguishable deviation from the normal week rhythm is Christmas time, highlighted from December 24 to January 1, when several consecutive days have high likelihoods. Thus, people are behaving similarly as on weekends which is natural as most people are having their holidays around Christmas. Another example is Easter. Good Friday on the 2nd of April and Easter Monday on the 5th of April in 2010 are merged with the weekend between them. These are both public holidays in Switzerland where the data has been collected.

5 Conclusions and Discussion

A statistical anomaly detection method based on multivariate binomial mixtures was applied to mobile proximity data collected from 106 users and spanning over one year. The method provided us with an anomaly measure that quantifies the typicality of people's social contexts. It also gave us a clustering of days.

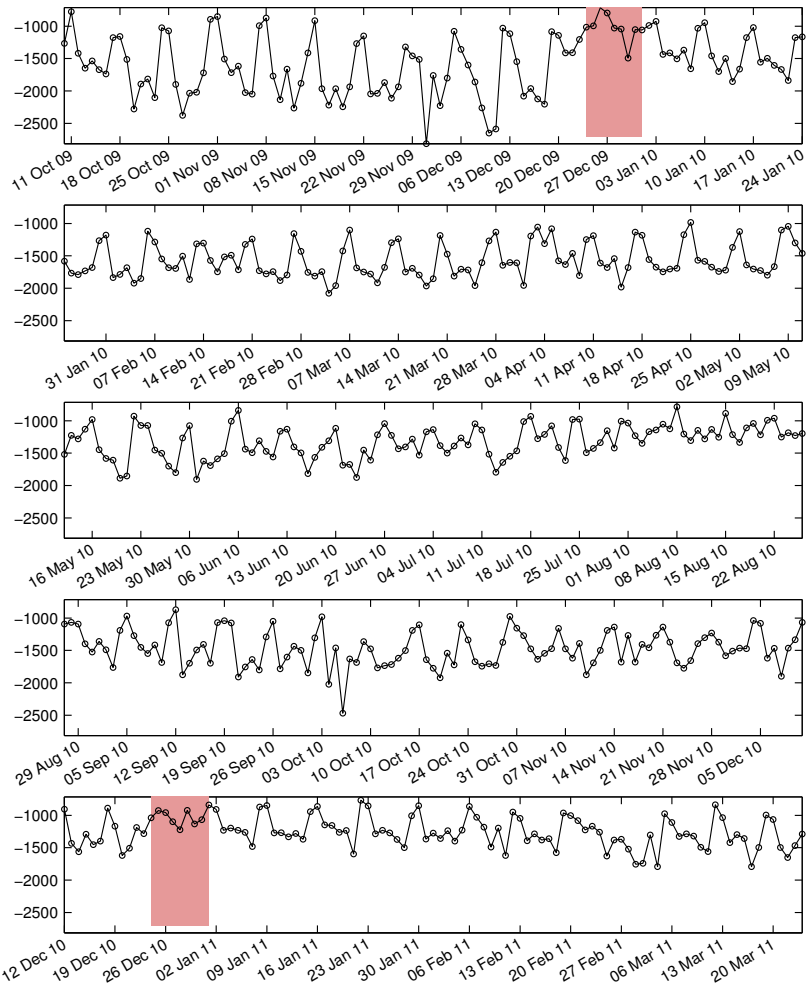


Fig. 5. Log-likelihood for each day averaged over all 106 users. Time periods from Christmas Eve (Dec 24) to New Year's Day (Jan 1) are highlighted. Tick marks are placed on Sundays.

A detailed analysis for a single user was presented. The analysis showed that the user's social contexts evolve over time, i.e., some clusters concentrated more on the beginning of the studied time period and some more on the end. Furthermore, some clusters described mostly weekends and public holidays indicating that the user's social context was different on these occasions. We took a closer look at April 1, 2010 which the algorithm identified as the most anomalous day for the user. The devices that were observed exceptionally frequently, or infrequently during that day were identified giving us insights about what is causing the anomaly.

We also analyzed the overall anomalousness of daily social contexts averaged over all users. This analysis revealed a clear weekly oscillation in the predictability of people's social contexts. Moreover, public holidays stood out even more clearly from the overall behavior than from the single user's behavior. These intuitive findings give credibility for the method. However, it remains a future challenge to assess the relevance of the found anomalies as we do not have a ground truth for them.

A criticism of the current work is that the model parameters are estimated for the whole data set, and then anomalousness is assessed globally. This is a limitation when the temporal structure is important for the analysis. Such a batch approach cannot identify anomalies caused by something being previously unseen, but later commonplace (e.g. first day of work at a new workplace). However, a batch model may be used to discover such changes, since it might lead to a previously inactive mixture component becoming active. Furthermore, a batch algorithm cannot easily differentiate an instant change (e.g. change of workplace) from slow temporal drift (people update their devices over time). If these concerns are important, then they would appropriately be addressed by fitting the model parameters in an online fashion, e.g., following the work of Stauffer and Grimson [15]. An online model addresses novelty by only using past observations during learning, and temporal drift by updating component parameters over time.

Acknowledgements. We would like to thank Nokia Research Center (NRC) Lausanne for giving us access to the data and, in particular, Jan Blom (NRC) for valuable discussions. We gratefully acknowledge funding from Tekes (VirtualCoach project), Academy of Finland and the Department of Information and Computer Science.

References

1. Eagle, N., Pentland, A.: Reality mining: sensing complex social systems. Personal and Ubiquitous Computing 10(4), 255–268 (2006)
2. Snipen, L., Almøy, T., Ussery, D.: Microbial comparative pan-genomics using binomial mixture models. BMC Genomics 10(1), 385 (2009)
3. Vatanen, T., Kuusela, M., Malmi, E., Raiko, T., Aaltonen, T., Nagai, Y.: Semi-supervised detection of collective anomalies with an application in high energy particle physics. In: Proc. IJCNN 2012, Brisbane, Australia (to appear, 2012)
4. Yamanishi, K., Takeuchi, J.I., Williams, G., Milne, P.: On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Mining and Knowledge Discovery 8(3), 275–300 (2004)
5. Roberts, S., Tarassenko, L.: A probabilistic resource allocating network for novelty detection. Neural Computation 6(2), 270–284 (1994)
6. Zhang, A., Ye, W., Wen, M.: Detection of anomaly collective call patterns. Journal of Computational Information Systems 7(10), 3614–3622 (2011)
7. Do, T.M.T., Gatica-Perez, D.: Human interaction discovery in smartphone proximity networks. Personal and Ubiquitous Computing, 1–19

8. Kiukkonen, N., Blom, J., Dousse, O., Gatica-Perez, D., Laurila, J.: Towards rich mobile phone datasets: Lausanne data collection campaign. In: Proc. ICPS, Berlin (2010)
9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys (CSUR) 41(3), 15 (2009)
10. Getzfeld, A.R.: Essentials of abnormal psychology, vol. 5. Wiley (2006)
11. McLachlan, G.J., Peel, D.: Finite mixture models, vol. 299. Wiley- Interscience (2000)
12. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 1–38 (1977)
13. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with Bregman divergences. The Journal of Machine Learning Research 6, 1705–1749 (2005)
14. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society. Series B (Methodological), 111–147 (1974)
15. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real- time tracking. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, pp. 246–252 (1999)

Constrained Clustering Using SAT

Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux,
Mehdi Khiari, and Samir Loudni

University of Caen Basse-Normandie – GREYC (CNRS UMR 6072)
Campus II, Côte de Nacre, 14000 Caen - France
{firstname.lastname}@unicaen.fr

Abstract. Constrained clustering - finding clusters that satisfy user-specified constraints - aims at providing more relevant clusters by adding constraints enforcing required properties. Leveraging the recent progress in declarative and constraint-based pattern mining, we propose an effective constraint-clustering approach handling a large set of constraints which are described by a generic constraint-based language. Starting from an initial solution, queries can easily be refined in order to focus on more interesting clustering solutions. We show how each constraint (and query) is encoded in SAT and solved by taking benefit from several features of SAT solvers. Experiments performed using `MiniSat` on several datasets from the UCI repository show the feasibility and the advantages of our approach.

1 Introduction

Clustering is one of the core problems in data mining. Clustering aims at partitioning data into groups (clusters) so that transactions occurring in the same cluster are similar but different from those appearing in other clusters [12]. The usual clustering problem is designed to find clusterings satisfying a nearest representative property while constrained clustering [3,19] aims at obtaining more relevant clusters by adding constraints enforcing several properties expressing background information on the problem at hand. Constraints deal with various types: (1) data objects' relationships (e.g., a set of objects must be (or not) in a same cluster [20]), (2) the description of the clusters (e.g., a cluster must have a minimal or a maximal size [2]), (3) both objects and clusters (e.g., a given object must be in a given cluster), (4) the characteristics of the clustering (e.g., the number of clusters),... Traditional clustering algorithms do not provide effective mechanisms to make use of this information. The goal of this paper is to propose a generic approach to fill this gap.

Recently, several works have investigated relationships between data mining and constraint programming (CP) to revisit data mining tasks in a declarative and generic way [6,14,15]. The user models a problem and expresses his queries by specifying *what* constraints need to be satisfied. The process greatly facilitates the search of knowledge and models such as clustering. The approach is enforced by the use of a constraint-based language [17]: it is sufficient to change

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 207–218, 2012.
© Springer-Verlag Berlin Heidelberg 2012

the specification in term of constraints to address different pattern mining problems. In the spirit of this promising avenue, we propose an effective constrained clustering approach handling a large set of constraints.

The paper brings the following contributions. First, we use the declarative modeling principle of CP to define a constrained clustering approach taking into account a large set of constraints on objects, a description of the clusters and the clustering process itself. By nature, clustering proceeds by iteratively refining queries until a satisfactory solution is found. Our method integrates in a natural way this stepwise refinement process based on the queries in order to focus on more interesting clustering solutions. Contrary to very numerous clustering methods that use heuristics or greedy algorithms, our method is complete. Second, we define an efficient SAT encoding which integrates features of SAT solvers (e.g., binary clauses, unit propagation, sorting networks) to solve the queries. Finally, an experimental study using `MiniSat` shows the feasibility and the effectiveness of our method on several datasets from the UCI repository.

Section 2 provides the background on the constraint-based language. Section 3 describes our method on constrained clustering with examples of constraints coming from the background information of the problem at hand. Section 4 addresses the point of how queries and constraints of the language are encoded and solved with SAT. Section 5 shows the effectiveness of our approach through several experiments. Section 6 presents related work.

2 Background: Constraint-Based Language

The constraint programming methodology is by nature declarative. It explains why studying relationships between CP and data mining has received a considerable attention to go towards generic and declarative data mining methods [6,14,15]. This section sketches our constraint-based language that enables us to specify in term of constraints different pattern mining problems [17]. This language forms the first step of our constrained clustering method proposed in Section 3. In the remainder of this section, we only focus on primitives of the language that will be used in this paper.

Let $\mathcal{I}$ be a set of n distinct literals called items, an itemset (or *pattern*) is a non-null subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$. A *transactional dataset* $\mathcal{T}$ is a multi-set of m itemsets of $\mathcal{L}_{\mathcal{I}}$. Each itemset, usually called a *transaction* or *object*, is a database entry. For instance, Table 1 gives a transactional dataset $\mathcal{T}$ with m=11 transactions $t_1, \ldots, t_{11}$ described by n=10 items. This toy dataset is inspired by the `Zoo` dataset from the UCI repository.

Terms are built from constants, variables, operators, and function symbols. Constants are either numerical values, or patterns, or transactions. Variables, noted X_j, for $1 \leq j \leq k$, represent the unknown patterns (or clusters). Operators can be set ones (as $\cap, \cup, \backslash$) or numerical ones (as $+, -, \times, /$). Built-in function symbols involve one or several terms:

- $\texttt{cover}(X_j) = \{t \mid t \in \mathcal{T}, X_j \subseteq t\}$ set of transactions covered by X_j.
- $\texttt{freq}(X_j) = \mid \{t \mid t \in \mathcal{T}, X_j \subseteq t\} \mid$ is the frequency of pattern X_j.

- $\texttt{size}(X_j) = \mid \{i \mid i \in \mathcal{I}, i \in X_j\} \mid$ is the size of pattern X_j.
- $\texttt{overlapItems}(X_i, X_j) = \mid X_i \cap X_j \mid$ is the number of items shared by both X_i and X_j.
- $\texttt{overlapTransactions}(X_i, X_j) = \mid \texttt{cover}(X_i) \cap \texttt{cover}(X_j) \mid$ is the number of transactions covered by both X_i and X_j.

Constraints are relations over terms that can be satisfied or not. There are three kinds of built-in constraints: numerical constraints (like $<$, $\leq$, $=$, $\neq$, $\geq$, $>$), set constraints (like $=$, $\neq$, $\in$, $\notin$, $\subset$, $\subseteq$), and dedicated constraints like:

- $\texttt{isNotEmpty}(X_j)$ is satisfied iff $X_j \neq \emptyset$
- $\texttt{coverTransactions}([X_1, ..., X_k])$ is satisfied iff each transaction is covered by at least one pattern $(\bigcup_{1 \leq i \leq k} \texttt{cover}(X_i) = \mathcal{T})$
- $\texttt{noOverlapTransactions}([X_1, ..., X_k])$ is satisfied iff all i, j s.t. $1 \leq i < j \leq k$, $\texttt{cover}(X_i) \cap \texttt{cover}(X_j) = \emptyset$
- $\texttt{coverItems}([X_1, ..., X_k])$ is satisfied iff each item belongs to at least one pattern $(\cup_{1 \leq i \leq k} X_i = \mathcal{I})$
- $\texttt{noOverlapItems}([X_1, ..., X_k])$ is satisfied iff $\forall\ i, j$ s.t. $1 \leq i < j \leq k$, $X_i \cap X_j = \emptyset$
- $\texttt{canonical}([X_1, ..., X_k])$ is satisfied iff for all i s.t. $1 \leq i < k$, pattern X_i is less than pattern X_{i+1} with respect to the lexicographic order.

Finally, a query is a conjunction of constraints as illustrated in the next section.

Table 1. `Animal` dataset

Species	trans	Fur	Feather	Scale	Milk	Egg	Beak	Bone	Meat	Grass	Fish
Cat	t_1	1	0	0	1	0	0	1	1	0	1
Cow	t_2	1	0	0	1	0	0	1	0	1	0
Crow	t_3	0	1	0	0	1	1	1	1	1	1
Daulphin	t_4	0	0	0	1	0	0	1	0	0	1
Dog	t_5	1	0	0	1	0	0	1	1	0	0
Goose	t_6	0	1	0	0	1	1	1	0	1	0
Platypus	t_7	1	0	0	1	1	1	1	1	0	0
Salmon	t_8	0	0	1	0	1	0	0	0	1	1
Shark	t_9	0	0	0	0	0	0	0	1	0	1
Trout	t_{10}	0	0	1	0	1	0	0	0	1	0
Vulture	t_{11}	0	1	0	0	1	1	1	1	0	1

3 Constrained Clustering: Modeling

3.1 Introduction: Modeling a Clustering Query

A clustering problem can be thought of as a scenario in which a user wishes to obtain a partition $\pi_T = (X_1, ..., X_k)$ of a dataset $\mathcal{T}$, containing m objects, into k (non-empty) clusters. A clustering problem intrinsically owns a lot of symmetrical solutions: any permutation of π_T is a solution. The $\texttt{canonical}([X_1, ..., X_k])$ constraint is used to avoid symmetrical solutions.

So, we can define the $\texttt{isClustering}([X_1, ..., X_k])$ constraint:

$$\texttt{isClustering}([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \texttt{isNotEmpty}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{noOverlapTransactions}([X_1, ..., X_k]) \wedge \\ \texttt{canonical}([X_1, ..., X_k]) \end{cases}$$

3.2 Integrating Background Information in the Clustering Process

In many application domains, background information on the domain and/or dataset is often available and the data analyst would like to integrate it in the process to improve the clustering results. Such a knowledge is usually expressed as *transaction-level* constraints (like the `mustLink` and `cannotLink` constraints [20], and as *cluster-level* constraints (like *Maximum Diameter* and *Minimum Separation* contraints [7,8]).

We start by describing how these information can be modeled thanks to the constraint-based language (see Section 2). Then, we show how our method allow to combine them to achieve more relevant queries. Let $\mathcal{T}$ be a dataset of m transactions. Let $d(t_1, t_2)$ be a distance over transactions.

Transaction-Level Constraints consist in `mustLink` and `cannotLink` constraints [20]:

- $\texttt{mustLink}(t_1, t_2)$ ensures that transactions t_1 and t_2 belong to the same cluster.
- $\texttt{cannotLink}(t_1, t_2)$ ensures that transactions t_1 and t_2 do not belong to the same cluster.

Cluster-Level Constraints. The *diameter of a cluster* X_j is the maximum distance between a pair of transactions in X_j [7,8]. The cluster-level constraint *maximum diameter* requires that the diameter of any cluster be at most a given value α. To achieve this, we must ensure that any pair of transactions (t_i, t_j) with $d(t_i, t_j) > \alpha$ are in different clusters. So, for $1 \leq i < j \leq m$, if $d(t_i, t_j) > \alpha$ then the constraint $\texttt{cannotLink}(t_i, t_j)$ must be added.

The *separation between two clusters* X_i *and* X_j is the minimum distance between a pair of transactions, one from X_i and the other from X_j. The cluster-level constraint Minimum Separation requires that the separation between two clusters be at least a given value β. To achieve this, we must ensure that any pair of transactions (t_i, t_j) with $d(t_i, t_j) < \beta$ are in the same cluster. So, for $1 \leq i < j \leq m$, if $d(t_i, t_j) < \beta$ then the constraint $\texttt{mustLink}(t_i, t_j)$ must be added. Cluster-level constraints can be combined together (query q_1).

$$q_1([X_1, ..., X_k]) \equiv \begin{cases} \texttt{isClustering}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i, t_j) < \beta}\ \texttt{mustLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i, t_j) > \alpha}\ \texttt{cannotLink}(t_i, t_j) \end{cases}$$

The method can be performed with any distance between transactions. For instance, when transactions are described with numerical values, a numerical distance such as the euclidian distance can be used.

Seeding. Background information both on transactions and clusters is easily modeled in the same way. Let t_{i_0} and t_{j_0} be two transactions and X_{j_1} a cluster. t_{i_0} and t_{j_0} must be (resp. must not be) in a same cluster is modeled by adding the constraint $\texttt{mustLink}(t_{i_0}, t_{j_0})$ (resp. $\texttt{cannotLink}(t_{i_0}, t_{j_0})$). t_{i_0} must be (resp. must not be) in the cluster X_{j_1} is modeled by adding the constraint $t_{i_0} \in X_{j_1}$ (resp. $t_{i_0} \notin X_{j_1}$).

3.3 Stepwise Refinements for Clustering

A major strength of our approach is to provide a simple and efficient way to declare and refine queries, that is usually the process conducted by a data analyst when he performs clustering tasks. Starting from an initial query (like q_1), the data analyst can express that he prefers solutions with a minimal size of the clusters, in which the sizes of clusters do not differ too much from each other, etc. In practice, the data analyst successively refines the query (deriving q_{i+1} from q_i) until he considers that relevant information has been extracted. This stepwise refinement process is easily handled by our constrained clustering approach as illustrated below.

Removing Clusterings with Small Size Patterns. A clustering including at least one cluster X_i of small size is not considered as useful because X_i does not ensure enough similarity between transactions associated to X_i. Adding a minimal size threshold solves this drawback (query q_2).

$$q_2([X_1, ..., X_k]) \equiv \begin{cases} \texttt{isClustering}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) < \beta} \texttt{mustLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) > \alpha} \texttt{cannotLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i \leq k} \texttt{size}(X_i) \geq \delta \end{cases}$$

Balanced Clustering. Clustering solutions in which the sizes of clusters do not differ too much from each other are generally preferred. For any pair of clusters (X_i, X_j), their difference of sizes must be lower than a threshold $\Delta \times m$ where Δ is a percentage (query q_3).

$$q_3([X_1, ..., X_k]) \equiv \begin{cases} \texttt{isClustering}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) < \beta} \texttt{mustLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) > \alpha} \texttt{cannotLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i < j \leq k} \mid \texttt{size}(X_i) - \texttt{size}(X_j) \mid \leq \Delta \times m \end{cases}$$

3.4 An Example of Stepwise Refinements

Let k=3 and $d(t_1, t_2)$ be the Hamming distance between transaction t_1 and transaction t_2. Using the dataset $\mathcal{T}$ described in Table 1, query q_1' provides 966 solutions for α=9 and β=1. By refining these thresholds (decreasing the maximal diameter to α=8 and enlarging the minimal separation to β=2), there remain four solutions (see Table 2).

$$q_1'([X_1, X_2, X_3]) \equiv \begin{cases} \texttt{isClustering}([X_1, X_2, X_3]) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) < \beta} \texttt{mustLink}(t_i, t_j) \wedge \\ \wedge_{1 \leq i < j \leq m, d(t_i,t_j) > \alpha} \texttt{cannotLink}(t_i, t_j) \end{cases}$$

Table 2. Set of clusterings for q'_1 (α=8 and β=2)

Sol.	X_1	X_2	X_3
s_1	$\{t_1,t_2,t_4,t_5,t_7\}$	$\{t_3,t_6,t_8,t_{10},t_{11}\}$	$\{t_9\}$
s_2	$\{t_1,t_2,t_4,t_5,t_7\}$	$\{t_3,t_6,t_{11}\}$	$\{t_8,t_9,t_{10}\}$
s_3	$\{t_1,t_2,t_4,t_5,t_7\}$	$\{t_3,t_6,t_9,t_{11}\}$	$\{t_8,t_{10}\}$
s_4	$\{t_1,t_2,t_4,t_5,t_7,t_9\}$	$\{t_3,t_6,t_{11}\}$	$\{t_8,t_{10}\}$

Clusters with a small size (e.g., $\{t_9\}$ in solution s_1) are considered irrelevant. By adding δ=2 as a minimal cluster size threshold, we get query q'_2 and there remains three solutions (s_2, s_3, and s_4, see Table 2).

$$q'_2([X_1, X_2, X_3]) \equiv \begin{cases} q'_1([X_1, X_2, X_3]) \wedge \\ \bigwedge_{1 \leq i \leq 3} \texttt{size}(X_i) \geq \delta \end{cases}$$

The user may want to indicate that the *shark* (t_9) must be in the same cluster as the *salmon* (t_8). This is done with a `mustlink` constraint. The solution s_2 is then the unique solution for the query q'_3 where the k=3 clusters respectively denote mammals, birds and fish.

$$q'_3([X_1, X_2, X_3]) \equiv \begin{cases} q'_2([X_1, X_2, X_3]) \wedge \\ \texttt{mustlink}(t_8, t_9) \end{cases}$$

3.5 Other Clustering Problems

In the same way, it is easy to express other clustering problems [5] such as soft clustering and co-clustering, the latter being well-used in bioinformatics for exploring gene expression data. **Soft clustering** is a relaxed version of the clustering where small overlaps on transactions (less than a threshold δ_T) are allowed.

$$q_4([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \texttt{isNotEmpty}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k}\ \texttt{overlapTransactions}(X_i, X_j) \leq \delta_T \wedge \\ \texttt{canonical}([X_1, ..., X_k]) \end{cases}$$

Co-clustering consists in finding k clusters covering both the set of transactions and the set of items, without any overlap on transactions or on items.

$$q_5([X_1, ..., X_k]) \equiv \begin{cases} \texttt{isClustering}([X_1, ..., X_k]) \wedge \\ \texttt{coverItems}([X_1, ..., X_k]) \wedge \\ \texttt{noOverlapItems}([X_1, ..., X_k]) \end{cases}$$

Soft co-clustering is a relaxed version of the co-clustering, allowing small overlaps on transactions (less than δ_T) and on items (less than δ_I).

$$q_6([X_1, ..., X_k]) \equiv \begin{cases} \wedge_{1 \leq i \leq k}\ \texttt{isNotEmpty}(X_i) \wedge \\ \texttt{coverTransactions}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k}\ \texttt{overlapTransactions}(X_i, X_j) \leq \delta_T \wedge \\ \texttt{coverItems}([X_1, ..., X_k]) \wedge \\ \wedge_{1 \leq i < j \leq k}\ \texttt{overlapItems}(X_i, X_j) \leq \delta_I \wedge \\ \texttt{canonical}([X_1, ..., X_k]) \end{cases}$$

4 Constrained Clustering: SAT Encoding

4.1 A Short Overview of SAT Solvers

Satisfiability (SAT) is the problem of determining if the variables of a given boolean formula can be assigned in such a way as to make the formula be evaluated to `True`. A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses, where a clause is a disjunction of literals.

SAT solvers perform filtering using unit propagation. If a CNF $\mathcal{F}$ contains a unit clause (composed of a single literal l), then $\mathcal{F}$ is satisfied, if and only if, l is assigned to `True`. So every clause containing l can be removed and $\neg l$ can be deleted in every clause it occurs. Binary clauses are well suited for unit propagation. If one of its two literals is assigned, a binary clause is either removed or becomes unitary giving raise to another filtering step (by unit propagation).

Efficient and scalable algorithms for SAT, that were developed over the last decade, have contributed to dramatic advances in the ability to automatically solve problem instances involving tens of thousands of variables and millions of constraints. That is why, we have chosen to encode a query as a CNF and then use a SAT solver to answer it.

4.2 Variables and Encoding of Partitioning Constraints

The data analyst formulates his queries by using the constraint-based language introduced above. Let $\mathcal{T}$ be the dataset to be proceeded. The CNF encoding a query q_i is the conjunction of the CNFs of the constraints involved in q_i.

Each unknown cluster is modeled using m boolean variables $T_{t,j}$ such that $(T_{t,j} = \texttt{True})$ iff transaction t belongs to cluster j. A cluster is referenced by its index between 1 and k.

- `coverTransactions`$([X_1, ..., X_k])$ ensures that each transaction $t \in \mathcal{T}$ belongs to at least one cluster. So, for each t, there exists at least one cluster X_j s.t. $t \in X_j$.
$$\bigwedge_{t \in \mathcal{T}} \Big(\bigvee_{j \in [1..k]} T_{t,j} \Big)$$
- `noOverlapTransactions`$([X_1, ..., X_k])$ ensures that each transaction $t \in \mathcal{T}$ belongs to at most one cluster. So, a transaction t belongs to (at least) two clusters iff there exist X_i and X_j s.t. $(t \in X_i) \wedge (t \in X_j)$, i.e. $\bigvee_{1 \leq i < j \leq k} (T_{t,i} \wedge T_{t,j})$. So, the negation must hold for each transaction t.
$$\bigwedge_{t \in \mathcal{T}} \Big(\bigwedge_{1 \leq i < j \leq k} (\neg T_{t,i} \vee \neg T_{t,j}) \Big)$$
- `isNotEmpty`(X_j) ensures that there exists at least one transaction $t \in \mathcal{T}$ that belongs to cluster X_j and is modeled by the clause: $\vee_{t \in \mathcal{T}} T_{t,j}$
- `canonical`$([X_1, ..., X_k])$ ensures that, for all i s.t. $1 \leq i < k$, cluster X_i is less than cluster X_{i+1}. This constraint is encoded using a binary comparator.

A constraint `coverItems`$([X_1, ..., X_k])$ (resp. `noOverlapItems`$([X_1, ..., X_k])$) is encoded in the same way as a constraint `coverTransactions`$([X_1, ..., X_k])$ (resp. `noOverlapTransactions`$([X_1, ..., X_k])$).

4.3 Encoding mustLink and cannotLink Constraints

$\texttt{mustLink}(t_1, t_2)$ ensures that the transactions t_1 and t_2 belong to the same cluster. So, for each $j \in [1..k]$, $T_{t_1,j} \Leftrightarrow T_{t_2,j}$.

$$\bigwedge_{1 \leq j \leq k} (\neg T_{t_1,j} \vee T_{t_2,j}) \wedge (T_{t_1,j} \vee \neg T_{t_2,j})$$

In the same way, $\texttt{cannotLink}(t_1, t_2)$ is encoded as:

$$\bigwedge_{1 \leq j \leq k} (\neg T_{t_1,j} \vee \neg T_{t_2,j}) \wedge (T_{t_1,j} \vee T_{t_2,j})$$

So each transaction level constraint is encoded using $(2 \times k)$ binary clauses.

4.4 Encoding Threshold Constraints Using Sorting Networks

Threshold constraints are directly modeled using cardinality constraints. So, $\texttt{size}(X_j) \geq \delta_1$ is modeled as $\#(T_{1,j}, T_{2,j}, ..., T_{m,j}) \geq \delta_1$. This cardinality constraint states that at least δ_1 variables $T_{t,j}$ must be assigned to `True`. Other threshold constraints involving function symbols are encoded in the same way.

Several efficient CNF encodings of cardinality constraints have been proposed [1,18]. Cardinality constraints are encoded thanks to unary adders in order to perform filtering by unit propagation. But, such encodings require a quadratic number of clauses [1] or they depend on the value of the threshold [18]. Moreover, for clustering, or other data mining tasks, thresholds can have rather large values, so the size of such encodings can quickly become prohibitive.

We used sorting networks to encode threshold constraints because the size of the resulting encoding does not depend on the value of the threshold. Moreover, using sorting networks to implement cardinality constraints preserves arc-consistency [11]. The odd-even *Batcher sort* [4] proved to be very efficient compared to other encodings of cardinality constraints [11].

4.5 Transitive Inference of mustLink and cannotLink Constraints

Let $G=(V, E)$ be the `mustLink` graph where $V=\mathcal{T}$. There is an edge between t_i and t_j iff there exists a constraint $\texttt{mustLink}(t_i, t_j)$ [3,20]. Let CC_1 and CC_2 be two connected components of G. *(i) If there exists a constraint* $\texttt{mustLink}(t_1, t_2)$ *with* $t_1 \in CC_1$ *and* $t_2 \in CC_2$*, then we can infer the constraints* $\texttt{mustLink}(x, y)$ *for all* $x \in CC_1$ *and* $y \in CC_2$. Contrary to `mustLink`, `cannotLink` is not an equivalence relation, but *(ii) If there exists a constraint* $\texttt{cannotLink}(t_1, t_2)$ *with* $t_1 \in CC_1$ *and* $t_2 \in CC_2$*, then we can infer the constraints* $\texttt{cannotLink}(x, y)$ *for all* $x \in CC_1$ *and* $y \in CC_2$.

Such entailments are usually performed by adding all the inferred `mustLink` and `cannotLink` constraints [3,20]. But, using our SAT encoding (see Section 4.3), there is no need to perform those addings: all inferred constraints are implicitly stated and will be taken into account by the SAT solver. In fact, `mustLink` and `cannotLink` constraints are encoded using equivalence between boolean variables (see Section 4.3).

Table 3. Dataset's characteristics

dataset	Australian	Mushroom	P.-Tumor	Soybean	Zoo
m	653	8124	336	630	101
n	125	119	36	50	36
density	0.40	0.19	0.48	0.32	0.44

4.6 Ensuring Completeness

Given a CNF, SAT solvers either find one instantiation (and only one) for the variables evaluating the formula to `True`, or prove there is no such an instantiation. In order to ensure the completeness of our approach, restarts are performed. Let $\mathcal{F}$ be the CNF modeling a query q. Resolution begins with $\mathcal{F}$. Then, after having obtained the i-th solution s_i, its negation $\neg s_i$ is added to the (current) CNF and resolution is restarted in order to look for another solution. The process ends when a failure occurs, i.e. when all solutions have been found. Using restarts may seem too naive, but in practice is efficient enough. As CNFs contain much binary clauses, filtering by unit propagation is very effective (see experiments performed in Section 5).

5 Experiments

The goal of the experiments is to provide better insights on our constrained clustering method according to several constraints and datasets. We used the `MiniSat`[1] solver [10] to implement our method. We performed experiments on several datasets from the UCI repository[2] (see Table 3). Experiments were conducted on a Core2Duo E8400 (2.83GHz) with 4GB of RAM. For each experiment, we report the CPU-times needed to compute the first and the first ten solutions[3] according to the required number of clusters k.

We used the Hamming distance[4] between transactions. The maximum diameter α has been set to $n/2$ (if two transactions differ more than 50%, they cannot belong to the same cluster) and the minimum separation β to $n/20$ (if two transactions differ less than 5%, they must belong to the same cluster).

Fig. 1 reports CPU-times needed to compute the first and the first ten solutions for query q_1 (see Section 3.2). For each dataset, the first ten solutions (if there exist) are obtained very quickly, even for the dataset `Mushroom` which is the largest one.

Fig. 2 reports CPU-times needed to compute the first solution for balanced clustering (query q_3, Section 3.2). The balancing ratio Δ has been set to 10% (Fig. 2 left) and to 20% (Fig. 2 right). Even with additional threshold constraints, our approach is still efficient. Note that with such restrictive thresholds, few queries have a solution.

[1] `http://minisat.se/`

[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[3] Symbol '-' denotes the absence of solution for a query.

[4] Any distance can be used since distance is only used to state `mustLink` and `cannotLink` constraints.

dataset	k=4	k=8	k=12	k=16	k=20
Australian	-	0.02	0.25	0.89	1.59
Mushroom	0.96	1.38	3.94	6.64	-
P.-Tumor	-	0.03	0.08	0.12	0.13
Soybean	0.01	0.03	0.12	0.16	-
Zoo	0.01	0.01	0.02	0.03	0.03

k=4	k=8	k=12	k=16	k=20
-	0.05	0.31	0.96	1.70
1.28	1.53	5.38	8.24	-
-	0.03	0.11	0.15	0.17
0.02	0.05	0.14	0.18	-
0.01	0.01	0.03	0.04	0.04

Fig. 1. (q_1) Time in s. to obtain the 1^{st} sol. (left) and the first ten sol. (right)

dataset	k=4	k=8	k=12	k=16	k=20
Australian	-	19.7	27.46	45.23	180.5
Mushroom	16.9	-	-	-	-
P.-Tumor	-	1.06	2.22	6.16	6.05
Soybean	-	-	-	-	-
Zoo	0.01	0.09	-	-	-

k=4	k=8	k=12	k=16	k=20
-	7.20	13.10	46.15	69.76
19.6	-	-	-	-
-	0.91	3.30	3.88	6.08
-	-	-	-	-
0.03	0.10	-	-	-

Fig. 2. (q_3) Time in s. for the 1^{st} sol. with Δ=10% (left) and with Δ=20% (right)

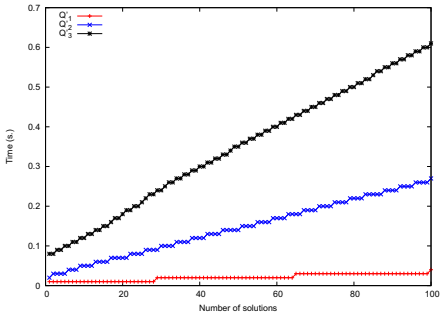


Fig. 3. Time for Zoo (k=6)

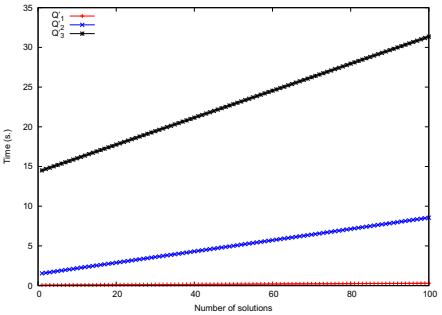


Fig. 4. Time for Australian (k=6)

Australian		$(k=2)$	$(k=6)$	$(k=10)$
q_1	#literal	2.6	10.4	18.2
	#clause	29.8	107.8	196.3
	r%	86.9	81.2	83.93
q_2	#literal	98.8	299.2	499.5
	#clause	318.7	974.5	1640
	r%	68.6	68.3	68.5
q_3	#literal	98.8	299.2	499.5
	#clause	318.7	974.5	1640
	r%	68.6	68.3	68.5

Mushroom		$(k=2)$	$(k=6)$	$(k=10)$
q_1	#literal	32.4	129.9	227.4
	#clause	1141	3651	6291
	r%	95.7	93.1	92.9
q_2	#literal	1343	4062	6781
	#clause	5075	15452	25960
	r%	73.2	72.9	73.1
q_3	#literal	1343	4062	6781
	#clause	5075	15452	25960
	r%	73.2	72.9	73.1

Fig. 5. Number of literals and clauses for the three queries (in thousands)

Fig. 3 depicts the CPU-times according to the number of solutions for dataset Zoo with k=6. A curve is associated to each of the three queries q_1 (red), q_2 with δ=$m/10$ (blue) and q_3 with Δ=20% (black). These curves *seem to be quasi-linear*. All the three queries mainly involve `mustLink` and `cannotLink` constraints which are encoded using binary clauses. As soon as a transaction is assigned to a cluster,

a lot of deductions are performed using unit propagation and transitive inferences (see Section 4.5). Nevertheless, further investigations are required to confirm this result. Fig. 4 provides similar results for dataset `Australian` with k=6.

Fig. 5 reports the size of the encodings of queries q_1, q_2, and q_3 for datasets `Australian` and `Mushroom`, as well as the ratio (r) of binary clauses constituting the CNF. The encoding of a query could be large (several millions of clauses), but the ratio r always remains very high. The size of q_2 is similar as the size of q_3 since only the bounds of the cardinality constraints are changed (see Section 4.4). **To sum up,** these experiments show that SAT solvers allow to solve efficiently this clustering task. They can find the first solutions (or prove there is none) in affordable times, even for medium scale size datasets as `Mushroom`. However, most of the clustering queries need threshold constraints which require more computational efforts.

6 Related Work

SAT for Clustering. Constraints on clusters (e.g., *Maximum Diameter* and *Minimum Separation*) into the `k-means` clustering algorithm [16] have been introduced by [7]. A formal complexity analysis of constraints on transactions and clusters is performed in [8]. Davidson *and al.* have proposed the first approach using SAT for clustering [9], but it deals only with a strong limited setting (k=2). The authors show how constraints both on transaction and cluster levels can be modeled and solved as instances of the 2-SAT problem. Gilpin and Davidson consider hierarchical constrained clustering and describe how dendograms can be modeled and solved as instances of the Horn-SAT problem [13]. We have also used SAT to implement primitives of our constraint-based language [17].

SAT for Mining Patterns. a SAT approach for enumerating all frequent patterns with wildcards in a given sequence has been proposed in [6].

7 Conclusion and Future Work

We have proposed a constrained clustering approach handling a large set of constraints. Solving is performed thanks to a SAT encoding for clustering and we have described how queries can be solved by taking benefit from features of SAT solvers. Experiments performed using `MiniSat` show the feasibility and the effectiveness of our approach. An insight of our work is that when a certain clustering task is modeled, many variants of that task can be modeled as well, changing or adding a few constraints is sufficient to allow this to happen.

As future work, we want to enrich our constraint-based language with further primitives to determine and/or constrain the cluster center locations. We also want to improve the current encoding (e.g., defining labeling orderings, exploiting backdoors, nogoods...). Another challenge is to propose an alternative encoding consuming less space and yet having relevant properties for an efficient solving. We want to conduct an in-depth study of the scalability of the approach to larger values of k and larger datasets. Finally, another promising direction is to integrate optimization criteria in our framework.

References

1. Bailleux, O., Boufkhad, Y.: Efficient CNF Encoding of Boolean Cardinality Constraints. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 108–122. Springer, Heidelberg (2003)
2. Banerjee, A., Ghosh, J.: Scalable clustering algorithms with balancing constraints. Data Min. Knowl. Discov. 13(3), 365–395 (2006)
3. Basu, S., Davidson, I., Wagstaff, K.L.: Constrained Clustering: Advances in Algorithms, Theory, and Applications. Chapman & Hall (2008)
4. Batcher, K.E.: Sorting networks and their applications. In: AFIPS Spring Joint Computing Conference. AFIPS Conference Proceedings, vol. 32, pp. 307–314. Thomson Book Company, Washington, D.C (1968)
5. Berkhin, P.: Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, USA (2002)
6. Coquery, E., Jabbour, S., Sais, L.: A constraint programming approach for enumerating motifs in a sequence. In: Workshop on Declarative Pattern Mining, ICDM 2011, Vancouver, Canada, pp. 1091–1097 (December 2011)
7. Davidson, I., Ravi, S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: SDM (2005)
8. Davidson, I., Ravi, S.: Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min. Knowl. Discov. 18(2), 257–282 (2009)
9. Davidson, I., Ravi, S., Shamis, L.: A SAT-based framework for efficient constrained clustering. In: SDM, pp. 94–105. SIAM (2010)
10. Eén, N., Sörensson, N.: An Extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
11. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. JSAT 2(1-4), 1–26 (2006)
12. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning 2(2), 139–172 (1987)
13. Gilpin, S., Davidson, I.: Incorporating SAT solvers into hierarchical clustering algorithms: an efficient and flexible approach. In: KDD 2011, pp. 1136–1144 (2011)
14. Guns, T., Nijssen, S., De Raedt, L.: Itemset mining: A constraint programming perspective. Artif. Intell. 175(12-13), 1951–1983 (2011)
15. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint Programming for Mining n-ary Patterns. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 552–567. Springer, Heidelberg (2010)
16. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
17. Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M., Loudni, S.: A constraint-based language for declarative pattern discovery. In: 27th Annual ACM Symposium on Applied Computing (SAC 2012), March 2012, pp. 438–444 (2012)
18. Sinz, C.: Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 827–831. Springer, Heidelberg (2005)
19. Tung, A.K.H., Han, J., Lakshmanan, L.V.S., Ng, R.T.: Constraint-Based Clustering in Large Databases. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 405–419. Springer, Heidelberg (2000)
20. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: ICML 2000, pp. 1103–1110. M. Kaufmann (2000)

Two-Stage Approach for Electricity Consumption Forecasting in Public Buildings

Antonio Morán[1], Miguel A. Prada[1], Serafín Alonso[1], Pablo Barrientos[1], Juan J. Fuertes[1], Manuel Domínguez[1], and Ignacio Díaz[2]

[1] SUPPRESS Research Group, Esc. de Ing. Industrial e Informática, Campus de Vegazana s/n, 24071, León, Spain
{a.moran,ma.prada,saloc,pbarf,jj.fuertes,mdomg}@unileon.es
http://suppress.unileon.es

[2] Dept. de Ing. Elétrica, Electrónica, de Computadores y Sistemas, Universidad de Oviedo, Campus de Viesques s/n, Ed. Departamental 2, 33204, Gijón, Spain
idiaz@isa.uniovi.es

Abstract. Many preprocessing and prediction techniques have been used for large-scale electricity load forecasting. However, small-scale prediction, such as in the case of public buildings, has received little attention. This field presents certain specific features. The most distinctive one is that consumption is extremely influenced by the activity in the building. For that reason, a suitable approach to predict the next 24-hour consumption profiles is presented in this paper. First, the features that influence the consumption are processed and selected. These environmental variables are used to cluster the consumption profiles in subsets of similar behavior using neural gas. A direct forecasting approach based on Support Vector Regression (SVR) is applied to each cluster to enhance the prediction. The input vector is selected from a set of past values. The approach is validated on teaching and research buildings at the University of León.

Keywords: Time-series prediction, electricity consumption forecasting, feature selection, clustering, regression.

1 Introduction

The prediction of electricity consumption is a problem that has been extensively studied in the literature. Many methods including Generalized Regression Neural Networks (GRNN), Autoregressive Integrated Moving Averages (ARIMA), Self-Organizing Maps (SOM), Bayesian networks, Multi-Layer Perceptrons (MLP), Support Vector Regression (SVR), etc. [1] have been applied to the prediction of large-scale consumption of countries or, to a lesser extent, cities [2] [3]. However, despite the intensive research, few applications of these consumption forecasting algorithms to buildings have been presented [4].

Electricity consumption in buildings presents specific characteristics that make a direct application of large-scale approaches difficult. The problem does not only come from the abrupt changes and the periods with very low consumption. The consumption is also extremely influenced by the activity in the building, which is not directly observed and depends on several conditions. This circumstance causes that buildings with

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 219–229, 2012.
© Springer-Verlag Berlin Heidelberg 2012

apparently similar characteristics present very different behaviors. Nevertheless, the estimation of short-term future consumption in buildings is interesting because it can lead to substantial savings. Predicting consumption allows to take advantage of power auction markets, where electricity can be purchased cheaper, or to forecast in advance the occurrence of consumption peaks that are penalized by the supplier. For this reason, an approach for the short-term prediction of electricity consumption in public buildings is presented in this work.

Among the numerous forecasting algorithms, the ones that build models only from the past evolution of consumption stand out. To achieve that, a regression algorithm is trained taking a sample as output and a vector of past values of the observed series as input [5]. Many other algorithms additionally include information from environmental variables with an influence on the consumption in order to improve prediction results [6] [7]. In any case, a key factor for the performance of these approaches is the proper selection of the variables that are included to the input of the model [8] [9].

In this paper, a two-stage approach is proposed to forecast the consumption of a building in the next 24 hours. It uses information from both environmental variables and past samples. In the first stage, the external variables are used to partition the data in several subsets of similar consumption behaviors. A key factor in the success of this phase is the selection of the environmental variables that have an effect on the consumption. The second stage of the proposed approach involves the actual forecasting algorithm, which follows a direct strategy, uses SVR and is applied to each of the clusters obtained in the previous step. The input vectors to the forecasting algorithm only include information about past consumption and will be selected to maximize the accuracy. This approach is similar to the one proposed in [2] to forecast the power consumption. However, our work introduces modifications that improve its applicability to the previously stated purpose and puts the emphasis on feature processing and selection.

The approach has been validated in teaching and research buildings of the University of León. The University of León comprises a large number of buildings, so the ability of forecasting electricity consumption is useful to achieve energy and cost savings. The paper is organized as follows: Section 2 describes the analysis of variables that influence the consumption and the clustering stage which partitions the consumption data. In section 3, the selection of the regressor vector is explained and the approach proposed to forecast electricity consumption in buildings is presented. Section 4 shows the prediction results for two campus buildings at the University of León. Finally, conclusions are discussed in section 5.

2 Clustering Consumption Patterns with Respect to the Environmental Variables

Some variables, especially those related to the expected activity in a building, have a strong effect on the electricity consumption of a public building in the next 24 hours. As a consequence, the short-term evolution of the consumption can change drastically when it faces different conditions. For that reason, it seems reasonable to train separate models for different conditions. First, it is necessary to analyze the variables that are closely correlated with the consumption. Later, they are used to partition the space as a basis for the forecasting stage.

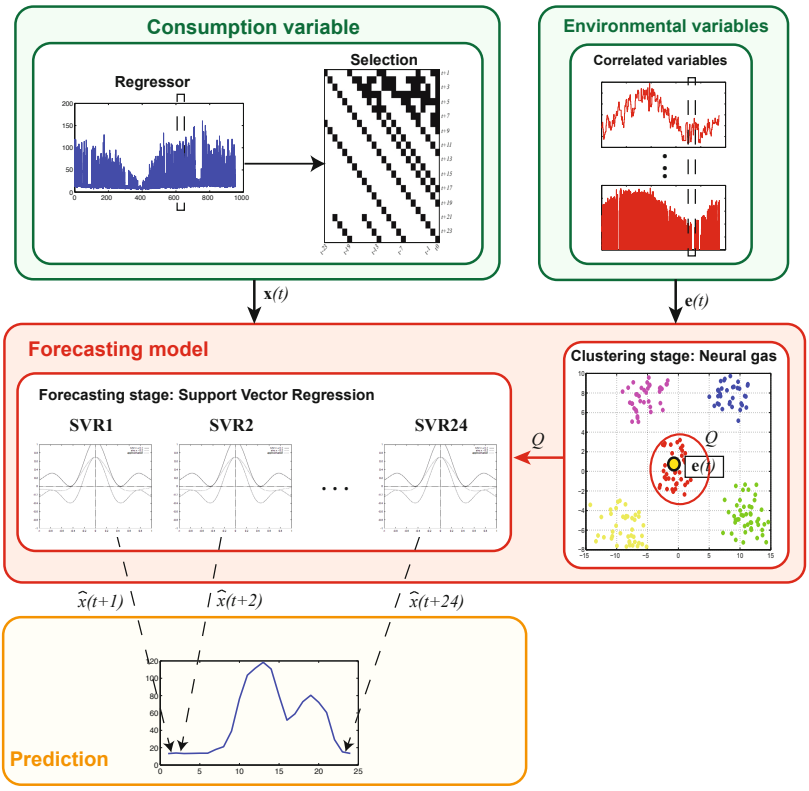


Fig. 1. Two-stage approach for forecasting

2.1 Variables Related to Electricity Consumption in Buildings

As well as in larger-scale electricity load forecasting, the environmental-related variables such as the daily average temperature, humidity or solar radiation are clear candidates to be considered to distinguish different consumption patterns. Their relevance can be directly assessed through a correlation analysis.

However, a certain degree of data processing is necessary to consider the expected activity of a building. This non-observable variable depends mainly on social constructs such as working or non-working days, holiday seasons, schedules, etc. whose expected value is known in advance. Most of these variables are binary, its effect is overlapped and often the magnitude of its influence on consumption is not straightforward. Their inclusion in the set of variables that is clustered might lead to suboptimal results. For this reason, the definition of a continuous *activity index* that brings together all the related information is proposed. A similar approach is proposed in [3]. For that purpose, vector quantization is performed on the data set of activity-related variables and the result is associated to a suitable indicator of the occupancy of a building. As an example, for the particular problem considered in this paper, i.e., forecasting the consumption in

university buildings, the indicator could be proportional to the overall power consumption of the campus.

Finally, the other class of features that should be included in this analysis are the time-related variables. Variables such as the day of the week or the hours can be used to consider the seasonality of the power consumption. However, these variables are characterized by the discontinuity in its definition. It is easy to see that the nature of consumption at 23 p.m. is closer to the one at 0 a.m. than what the distance between both values suggest. It is advisable to decompose this type of variables as two coordinates, X and Y, corresponding to the hour hand in a clock-like representation to avoid discontinuity between maximum and minimum values. Therefore, the hour is encoded as: $H_x = \sin\left(\frac{2\pi H}{24}\right)$ and $H_y = \cos\left(\frac{2\pi H}{24}\right)$.

The analysis of the correlation between these variables and the consumption dictates which ones constitute the vectors which are clustered. This process is described in the following section.

2.2 Clustering Stage: Neural Gas

The first stage of the proposed approach uses a *clustering* algorithm in order to partition data into several subsets with similar consumption behaviors (see Fig. 1). The input vectors $\mathbf{e}$ to this process are only the variables that affect the consumption, which are preprocessed and selected as discussed in the previous subsection. A different forecasting model will be associated to each cluster.

The method which inspires the proposed approach [2] used a self-organizing map to cluster day types. However, since visualization is not an aim of this stage, the *neural gas* algorithm [10] is used in this work. The topological constraint of this algorithm is not as rigid as the one imposed by the fixed grid in SOM, so it can perform better for clustering or vector quantization.

The main difference between neural gas and SOM is that the adaptation step does not use a neighborhood function defined in an output grid, but a neighborhood ranking of the codebook vectors with regard to its distance in the input space. The adaptation rule is therefore defined as:

$$\mathbf{q}_i(t+1) = \mathbf{q}_i(t) + \alpha(t)\Delta e^{-i/\lambda}\Delta[\mathbf{e}(t) - \mathbf{q}_i(t)], \tag{1}$$

where $\alpha(t)$ is the learning rate, λ determines the number of units significantly affected by the update and $i = 0, \ldots, N-1$ is the index of the codebook vector $\mathbf{q}$.

The algorithm is used as a partitive clustering where each of the resulting codebook vectors is used as the centroid of a cluster Q. The nearest codebook vector $\mathbf{q}$ to each input sample determines the cluster it belongs to. This method achieved better convergence than k-means in the preliminary results.

3 Forecasting Electricity Consumption

Forecasting models are trained for each cluster obtained in the previous step. First, the selection of the *input or regressor vector* is discussed, since the accuracy of the prediction results depends largely on the choice of regressor. Later, the algorithm proposed to forecast directly the next 24 hours is presented.

3.1 Regressor Selection

The regressor vector which is used as an input to the forecasting algorithm only contains past values of consumption. Nevertheless, we can distinguish two different classes of past values: the sequential part, which covers the past values immediately preceding the current one (e.g., the values corresponding to the previous 24 hours), and the periodic or partitioned part, which includes past consumptions observed the previous days at exactly the same time. Depending on the consumption pattern is better to use one type of information or the other one [9]. In this work, a regressor that combines information from both parts is used to improve the results:

$$\begin{aligned}\hat{x}\left(t+s\right) = f\left[x\left(t\right), x\left(t-1\right), \ldots, x\left(t-M_1+1\right),\right. &\rightarrow \text{Sequential} \\ \left. x\left(t+s-24\right), \ldots, x\left(t+s-24M_2\right)\right] &\rightarrow \text{Periodic}\end{aligned} \tag{2}$$

where M_1 is the number of the previous hours which are used to create the sequential part, M_2 is the number of previous days considered in the periodic part and s is the hour to be predicted from the present time.

However, the regressor vector should not include the whole set of variables. Many authors emphasize the need for input selection to guarantee high accuracy, efficiency and scalability in the prediction [8]. Different strategies can be adopted in this process. They are usually classified as wrappers, filters or embedded methods [11]. Each of the two parts of the input vector will be selected with a different method.

The inputs corresponding to the periodic part are selected using a wrapper approach by including an input at a time, up to reduced number of past days. The chosen inputs are the ones that minimize the error of the actual forecasting algorithm. Preliminary results show that 6 days back seems an appropriate maximum size of the candidate periodic part.

Once the size of the periodic part is selected, a filter approach is used to select the variables included in the sequential part to decrease the error. Filters select subsets of variables as a pre-processing step, so they do not require model training and, therefore, they are faster. When applying the filtering method, it is also necessary to find out the inputs dependent on the regressor. To select the regressor, several assessment methods, such as k-NN, mutual information and nonparametric noise estimation have been proposed [8]. The k-NN selection method provides comparable results to the other methods, which are computationally more complex, so it is the one chosen. It is based on the idea that samples with similar inputs must have similar output values. The output values corresponding to the k nearest neighbors of a sample x_i, denoted as $y_j(i)$ to simplify the notation, are averaged to obtain the approximated output $\hat{y}_i$: $\hat{y}_i = \sum_{j=1}^{k} y_j(i)/k$. The selected set of variables are those that minimize the error. The free parameter k can be selected by means of 10-fold cross validation [12]. The selection method can be applied exhaustively to all the possible combination of inputs, but this is computationally expensive. For that reason, several strategies such as forward selection, backward elimination and forward-backward selection have been proposed [13]. In this work, one run of forward selection and one of backward elimination are performed. The selected inputs are the ones that provide the overall minimum error.

3.2 Forecasting Stage: Support Vector Regression

This stage applies the actual forecasting techniques on the representative data obtained by the clustering of the first stage (see Fig. 1). Since the aim is to forecast the consumption in the following 24 hours, i.e., a multi-step ahead prediction, there are two choices to conduct the prediction: direct or recursive. In this work, a *direct method* is used, to avoid the accumulation of errors that happens in recursive methods due to the use of predicted values as known data to predict the new ones [14]. The direct method improves forecasting accuracy but increases the complexity of prediction, because a different model must be trained for each step ahead. Therefore, 24 models are computed for each cluster to predict the consumption for each hour of the next 24-hour window.

A nonlinear prediction algorithm, the *Support Vector Regression* (SVR), is proposed in this work, since it is regarded as a state-of-the-art method for time-series prediction [15]. SVR derives from the well-known Support Vector Machine (SVM) [16], generally used for classification, and its basic idea is to map nonlinearly the data into a high-dimensional space and to perform linear regression in that space.

SVR estimates the regression function, which relates the input of the regressor x_i and the output y_i, by means of the following minimization problem:

$$E = \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i + \xi_i^*), \tag{3}$$

where the minimization cost function is subject to the following constraints

$$\begin{array}{c} y_i - \left(w^T \phi\left(x_i\right) + b\right) \leq \varepsilon + \xi_i^* \\ \left(w^T \phi\left(x_i\right) + b\right) - y_i \leq \varepsilon + \xi_i \,, \\ \xi_i^*, \xi_i \geq 0, i = 1, \ldots, n. \end{array} \tag{4}$$

The inputs x_i are mapped to a higher-dimensional space by the kernel function Φ. The upper ξ_i^* and lower ξ_i slack variables cope with deviations larger than ϵ. The constant $C > 0$ determines the trade-off between the flatness of the function and the amount of errors outside the ϵ-insensitive tube $\left(w^T \phi\left(x_i\right) + b\right) - y_i \leq \varepsilon$ that are tolerated. The imposed constraints guarantee that most of the input data are smaller than ϵ.

4 Results

The approach is validated with buildings at the campus of the University of León. The campus is composed of thirty buildings which can be divided into two types according to their activity. On one hand, there are the buildings intended solely for teaching which are characterized by a strong dependence between the class and exam schedules and the power consumption. On the other hand, there are research buildings which present a higher level of the normal consumption. To illustrate the operation of the prediction approach, a representative building of each group has been chosen for the tests, specifically the School of Arts and the Veterinary building. Data from 396 days, sampled every hour, are used. The training set includes 9120 samples, whereas 384 samples are used for testing. The results of the most unfavorable prediction, i.e. the one that forecasts

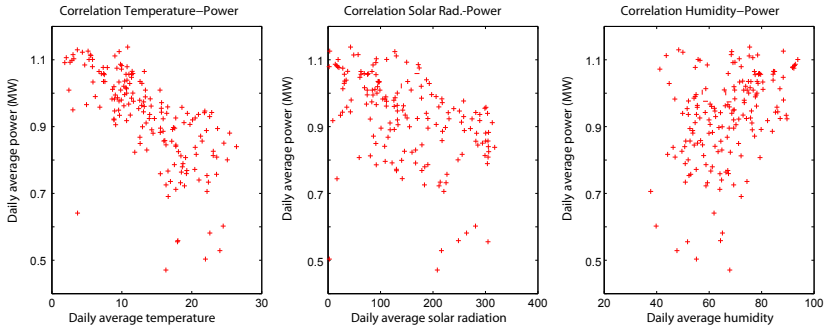


Fig. 2. Correlation between the continuous environmental variables and the power consumption of the campus

the consumption 24-hours ahead, are compared with the measured consumption. They are also compared with the 24-hour ahead prediction of a SVR that uses all the candidate inputs and does not benefit from the pre-processing and clustering proposed in this approach.

4.1 Clustering Results

The variables that influence the consumption are selected and processed as explained in section 2.1. A correlation analysis is performed to select among the weather-related variables. The candidate variables are humidity, temperature and solar radiation. Scatter plots of the correlation between the daily average value of the aforementioned variables and the daily average consumption of the whole campus are shown in Figure 2. It can be seen that the correlation presents a high negative value for temperature (-0.6947) and solar radiation (-0.486), i.e. the colder and darker a day is, the higher is the power consumed. This fact is intuitively explained by a more extensive use of lighting and heating. Humidity has a slightly lower correlation with consumption (0.3989) and is, in turn, closely correlated to temperature (-0.443). For that reason, the variable can be considered redundant and is discarded for subsequent calculations.

The remaining feature included in the vector is the activity index, which is computed from the variables contained in table 1. The activity-related variables are quantized using neural gas and associated to the overall power consumption of the campus, which is a good indicator of the effect of a combination of calendar events. The optimal number of clusters, according to the Davies-Bouldin index was fixed as 25. The calculation of a common activity index alleviates the correlation that would be introduced by using the individual power of each building.

The two coordinates of the hour, as defined in section 2.1 are also added to the vectors $\mathbf{e}$ that are going to be clustered, along with the power consumed both in the 24 and 48 previous hours. The information of the power consumption is incorporated to limit further environmental conditions and since the correlation of the temperature is high it also considered past values of it [17].

Table 1. Variables used to compute the activity index

Label	Variable	Variable type
D_x	X coordinate of the weekday	Senoidal
D_y	Y coordinate of the weekday	Senoidal
H	Holiday season	Binary
E	Exams	Binary
F	Non-working day	Binary
X	Miscellaneous event	Binary

Table 2. Variables selected for the clustering stage

Label	Description
H_x	X coordinate of the current time
H_y	Y coordinate of the current time
$I_a c$	Activity index expected for the next 24 hours
$\bar{T}_0$	Average temperature forecasted for the next 24 hours
$\bar{T}_{-24}$	Average temperature measured during the 24 previous hours
$\bar{R}_0$	Average solar radiation forecasted for the next 24 hours
$\bar{P}_{-24}$	Average power consumed in the 24 previous hours
$\bar{P}_{-48}$	Average power consumed in the 48 previous hours

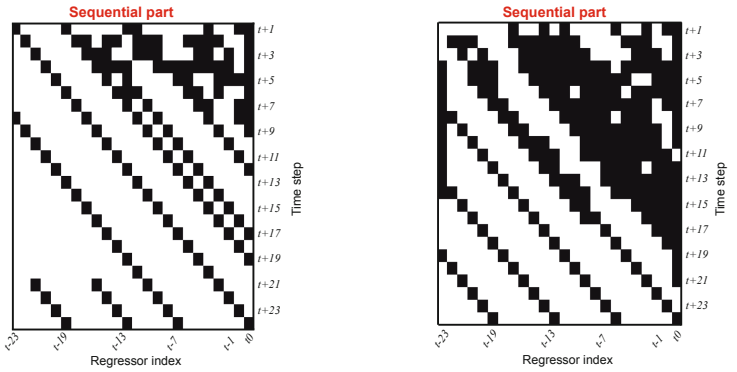
In short, the environmental variables selected for the clustering that constitutes the first stage of the proposed approach are shown in table 2. The neural gas clusters different environmental conditions to generate specific regression models for these days. The SOM Toolbox [18] has been used for this purpose. The number of units of the neural gas is determined to 44 for both buildings using the Davies-Bouldin index.

4.2 Forecasting Results

For each of the clusters obtained by grouping the data with regard to the variables that influence the consumption, 24 models are trained through the application of SVR for direct multiple-step ahead prediction.

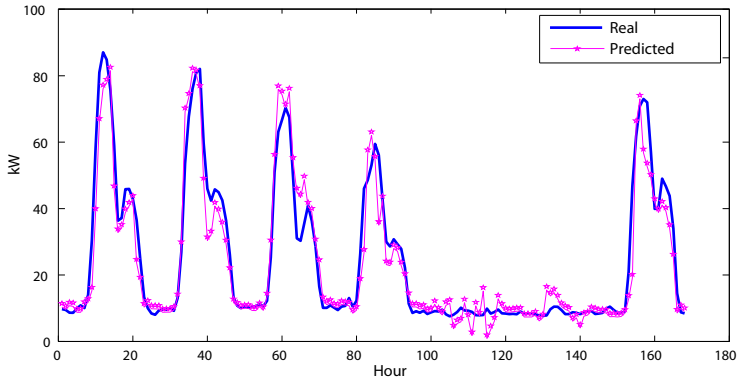
The regressor vector is selected as explained in section 3.1. With regard to the periodic part of the regressor, a forward method searches the regressor that minimizes the prediction error up to 6 days back. The inclusion of the previous 6 days minimizes the prediction error for the teaching building, whereas only the previous 5 days are added in the case of the research building. The inputs of the sequential part, i.e. the ones from the previous 24 hours, that minimize the k-NN error after one forward and one backward search are the ones selected as part of the input vector. The figure 3a shows the inputs selected in the teaching building. It can be seen that there is high periodicity in the selection of inputs. For the research building (see Fig. 3b) the periodicity is not so clear.

On the other hand, the 24 SVR models per cluster are trained using the LS-SVM Matlab toolbox [19]. The free parameters are tuned by means of a 10-fold cross-validation.

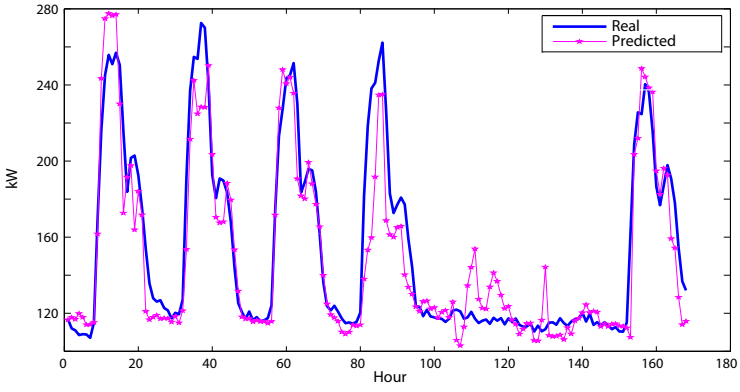


(a) Teaching building (b) Research building

Fig. 3. Results of the k-NN selection of the sequential part of the regressor vector



(a) Teaching building



(b) Research building

Fig. 4. Prediction results using the two-stage approach

In order to obtain the 24-hours ahead prediction from $\mathbf{x}(t)$, we compare the vector $\mathbf{e}(t)$ with the codebook vectors that result from the application of the neural gas. The best matching unit defines the cluster it belongs to and, consequently, which models are used to compute the forecasted consumption.

For conciseness purposes, the presented results are the ones corresponding to the most unfavorable prediction, i.e. the one that forecasts the consumption 24-hours ahead. The 24-hour ahead forecasted consumption is compared with the measured consumption and with the 24-hour ahead prediction of a SVR that uses all the candidate inputs and does not benefit from the pre-processing and clustering steps proposed in this approach. The results are computed for a week of data.

Figure 4a shows the results for the teaching building. It can be seen that the forecasted consumption matches the real one to a large extent. The NRMSE (Normalized Root Mean Square Error) is 0.0440, whereas the error rises up to 0.1219 for the simple SVR approach. On the other hand, Figure 4b shows the results of the research building. It can be seen again that the predictions are quite accurate. They provide a NRMSE of 0.0411, clearly better than the error provided by the plain SVR prediction: 0.1025.

5 Conclusions

In this paper a two-stage forecasting approach that clusters consumption patterns and train separated models for each group is presented. It uses neural gas for clustering and Support Vector Regression for forecasting. The approach also relies strongly on the selection of both the set of environmental variables that influence power consumption and the optimal set of past consumptions that optimize the regressor vector used as input to the forecasting method. The proposed approach is used to estimate short-term electric consumption in buildings of the University of León.

The results show that the prediction results achieve a good fit to the original data and the NRMSE is much lower using this methodology than using SVR directly. The University of León comprises a large number of buildings, so the ability of forecasting electricity consumption is useful to achieve energy and cost savings.

Acknowledgments. This work was supported in part by the Spanish *Ministerio de Ciencia e Innovación* (MICINN) and with European FEDER funds under grant DPI2009-13398-C02-02. A. Morán was supported by a grant from the *Consejería de Educación de la Junta de Castilla y León* and the European Social Fund. We thank the research group *Parsimonious Modelling* of the Helsinki Institute for Information Technology for their support and collaboration in the preliminary stages of this work.

References

1. Tiao, G.C., Tsay, R.S.: Some advances in non-linear and adaptive modelling in time-series. Journal of Forecasting 13(2), 109–131 (1994)
2. Fan, S., Chen, L.: Short-term load forecasting based on an adaptive hybrid method. IEEE Transactions on Power Systems 21(1), 392–401 (2006)

3. Sideratos, G., Vitellas, I., Hatziargyriou, N.: A load forecasting hybrid method for an isolated power system. In: 2011 16th International Conference on Intelligent System Application to Power Systems (ISAP), pp. 1–5 (September 2011)
4. Cao, L.: Support vector machines experts for time series forecasting. Neurocomputing 51(0), 321–339 (2003)
5. Weigend, A.S., Gershenfeld, N.A.: Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley, Reading (1993)
6. Gupta, P., Yamada, K.: Adaptive short-term forecasting of hourly loads using weather information. IEEE Transactions on PAS-91(5), 2085–2094 (1972)
7. Fan, S., Chen, L., Lee, W.J.: Short-term load forecasting using comprehensive combination based on multimeteorological information. IEEE Transactions on Industry Applications 45(4), 1460–1466 (2009)
8. Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., Lendasse, A.: Methodology for long-term prediction of time series. Neurocomputing 70(16-18), 2861–2869 (2007)
9. Fay, D., Ringwood, J.V., Condon, M., Kelly, M.: 24-h electrical load data-a sequential or partitioned time series? Neurocomputing 55(3-4), 469–498 (2003)
10. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: Neural-gas network for vector quantization and its application to time-series prediction. IEEE Trans. on Neural Networks 4(4), 558–569 (1993)
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
12. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), vol. 2 (1995)
13. Thompson, M.L.: Selection of variables in multiple regression: Part I. a review and evaluation. International Statistical Review 46(1), 1–19 (1978)
14. Sorjamaa, A., Hao, J., Lendasse, A.: Mutual Information and *k*-Nearest Neighbors Approximator for Time Series Prediction. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 553–558. Springer, Heidelberg (2005)
15. Müller, K., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with Support Vector Machines. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 999–1004. Springer, Heidelberg (1997)
16. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
17. Chen, J., Deng, S.J., Huo, X.: Electricity price curve modeling and forecasting by manifold learning. IEEE Transactions on Power Systems 23(3), 877–888 (2008)
18. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: SOM toolbox for Matlab 5 (2000)
19. Suykens, J., Gestel, T.V., Brabanter, J.D., Moor, B.D., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)

Online Predictive Model for Taxi Services

Luís Moreira-Matias[1,2], João Gama[2,3], Michel Ferreira[4,5], João Mendes-Moreira[1,2], and Luís Damas[5]

[1] Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal
[2] LIAAD-INESC TEC. Rua de Ceuta, 118, 6º, 4050-190 Porto – Portugal
[3] Faculdade de Economia, Universidade do Porto Rua Dr. Roberto Frias, s/n 4200-465 Porto – Portugal
[4] Instituto de Telecomunicações, Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto, 4169-007 Porto, Portugal
[5] Geolink, Pct. Adelino Amaro Costa, 772 4º Dto, 4050-012 Porto – Portugal
{luis.matias,jmoreira}@fe.up.pt, jgama@fep.up.pt, michel@dcc.fc.up.pt, luis@geolink.pt

Abstract. In recent years, both companies and researchers have been exploring intelligent data analysis to increase the profitability of the taxi industry. Intelligent systems for online taxi dispatching and time saving route finding have been built to do so. In this paper, we propose a novel methodology to produce online predictions regarding the spatial distribution of passenger demand throughout taxi stand networks. We have done so by assembling two well-known time series short-term forecast models: the time-varying Poisson models and ARIMA models. Our tests were performed using data gathered over a period of 6 months and collected from 63 taxi stands within the city of Porto, Portugal. Our results demonstrate that this model is a true major contribution to the driver mobility intelligence: 78% of the 253745 demanded taxi services were correctly forecasted in a 30 minutes horizon.

Keywords: ARIMA, Time-Varying Poisson Model, Taxi Services, Time Series, Data Streams.

1 Introduction

In the last decade, real-time vehicle location systems have attracted the attention of both companies and researchers for a new kind of rich spatio-temporal information. Taxi networks have largely been affected by this phenomenon, as such networks produce multiple data streams that can be explored using intelligent data analysis. Online systems for efficient taxi dispatching [1] and time-saving route finding [2] (among others) have already been developed to improve taxi service reliability.

The taxi driver mobility intelligence is a crucial feature to maximize both profit and reliability. There are few works which focus on this topic and use real GPS data. The work presented in [3] uses spatial-based clustering applied to the GPS historical data of a taxi network running in a large urban zone. Their goal consisted on

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 230–240, 2012.
© Springer-Verlag Berlin Heidelberg 2012

discovering passenger demand patterns over a period of time, by building a departure-destination-time cubic matrix. Recently, an innovative study has been presented in [4] to validate the triplet Time-Location-Strategy as the key feature to build a good passenger finding strategy. Here the L1-Norm-SVM was used as a feature selection tool to discover both efficient and inefficient passenger finding strategies based on a large-scale GPS dataset, from a taxi network running in a large city in China. An empirical study on the impact of the selected features was conducted and its conclusions were validated by the feature selection tool.

Both works represent recent studies about the selection of the best route (over a period of time) to maximize the number of passengers picked-up by each driver. However, there are three important issues that are not convincingly handled by these authors: (1) conditions which are not common to every urban area are assumed (e.g.: drivers are able to choose their routes freely and without any regulation); (2) their insights consist of offline patterns, which do not fulfill the ubiquitous potential of the data to provide decision support information in real-time; (3) the rising cost of fuel is clearly disallowing the economic viability of online or offline cruising strategies to get passengers.

In our work, we focus on the choice problem concerning which is the best taxi stand to go to after a passenger drop-off at a given location and time. One of the most important factors for this is the passenger demand at each taxi stand over a time span. In this paper, we present a streaming model to predict the number of services of a taxi network over a space span (taxi stand), for a short-time horizon of P-minutes. Based on both historical and real time GPS location and service data (passenger drop-off and pick-up) transmitted by the telematics installed in each vehicle, time series histograms are built for each stand containing the number of services with an aggregation of P-minutes. The predictive model was developed by adapting well-known time series forecasting techniques, such as the time varying Poisson model [5] and ARIMA (Autoregressive Integrated Moving Average) [6] to our problem. Our goal is to predict at an instant t how many services will be demanded during a period $[t, t+P]$ at each existent taxi stand, reusing the real service count on $[t, t+P]$ which has been extracted from the data to do the same for the instant $t+P$ and so on (i.e. the framework runs continuously in a stream). To the best of our knowledge, such approach has no parallel in the literature.

Our model was applied to data collected from a large-sized taxi network which contains a total of 63 taxi stands and has 441 vehicles running in the city of Porto, Portugal. In this city, the existing regulations force taxi drivers to pick a route to one of the existing stands after a passenger drop-off. Our study just uses the services obtained directly at the stands or which were automatically dispatched to the parked vehicles as input/output. This was done because the passenger demand at each taxi stand is the main feature to aid the taxi drivers' decision (since it represents 76% of the total number of services).

Our experiments use the real-time data arriving in a stream to produce predictions about the expected number of services demanded in each stand. Firstly, 20 weeks' worth of data was acquired to build our historic. Secondly, the demand for the next 8 weeks was predicted, by updating the historic time series over time. The results obtained were promising: our model predicted more than 78% of the services that actually emerged using an average computational time (i.e. average time per prediction) of 99.77 seconds.

The remainder of the paper is structured as follows. Section 2 formally describes our model. Section 3 describes how the dataset used was acquired and preprocessed, as well as some statistics about it. Section 4 outlines the testing of the methodology in a concrete scenario. Firstly, we introduce the evaluation metrics of our model, along with the experimental setup. We then present the obtained results. Section 5 concludes the paper and describes future work we intend to carry out.

2 The Model

Consider $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_N}\}$ to be the set of $\boldsymbol{N}$ taxi stands of interest and $\boldsymbol{D} = \{\boldsymbol{d_1}, \boldsymbol{d_2}, \ldots, \boldsymbol{d_j}\}$ to be a set of $\boldsymbol{j}$ possible passenger destinations. Our problem consists of choosing the best taxi stand at the instant $\boldsymbol{t}$ according to our forecast about passenger demand distribution over the time stands for the period $[\boldsymbol{t}, \boldsymbol{t+P}]$, as is illustrated in Fig. 1.

Consider $\boldsymbol{X_k} = \{\boldsymbol{X_{k,0}}, \boldsymbol{X_{k,1}}, \ldots, \boldsymbol{X_{k,t}}\}$ **to** be a time series for the number of demanded services at a taxi stand $\boldsymbol{k}$. Our goal is to build a model to determine the set of service count $\boldsymbol{X_{k,t+1}}$ for the instant $\boldsymbol{t+1}$ and for all taxi stands $\boldsymbol{k \epsilon \{1, N\}}$**.** To do so, we propose three distinct short-term prediction models and a well-known data stream ensemble framework to use them all. We formally describe those models along this section.

2.1 Time Varying Poisson Model

The following section presents a model firstly proposed in [5]. The demand of taxi services exhibit, like other transportation means [7], a periodicity in time on a daily basis that reflects the patterns of the underlying human activity, making the data appear non-homogeneous [5]. Fig. 2 illustrates a one month taxi service analysis extracted from our dataset that illustrates this periodicity (the dataset is described in detail in Section 3).

Consider the probability to have *n* taxi assignments in a determined time period – ***P(n)*** - following a *Poisson distribution*. We can define it using the following equation

$$\mathrm{P}(\mathrm{n};\lambda) = \frac{\mathrm{e}^{-\lambda}\lambda^{\mathrm{n}}}{\mathrm{n!}}, \tag{1}$$

where λ represents the rate (averaged number of the demand on taxi services) in a fixed time interval. However, in this specific problem, the rate λ is not constant but time-variant. As a result, we adapt it as a function of time, i.e. $\lambda(t)$, transforming the Poisson distribution into a nonhomogeneous one. Consider $\lambda(t)$ to be defined as follows

$$\lambda(\mathrm{t}) = \lambda_0 \delta_{\mathrm{d(t)}} \eta_{\mathrm{d(t),h(t)}}, \tag{2}$$

where $d(t)$ represents the weekday {1=Sunday, 2=Monday,…}; $h(t)$ the period in which time *t* falls (e.g. the time 00:31 is contained in period 2 if we consider 30-minute periods).

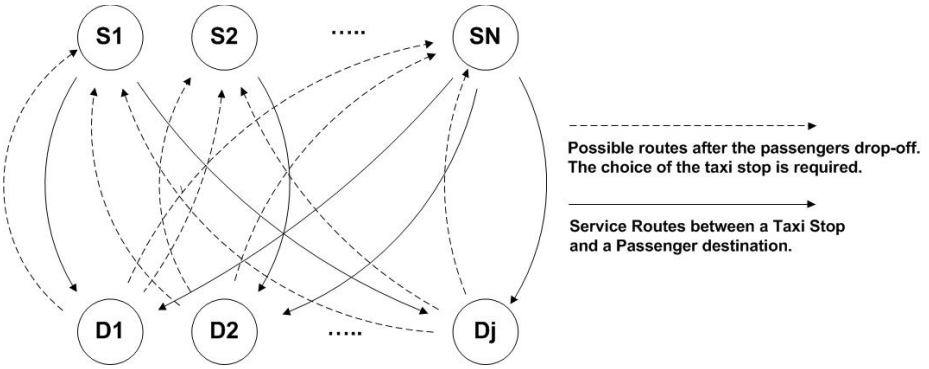


Fig. 1. A schema to illustrate our problem

It requires the validity of both equations

$$\sum_{i=1}^{7} \delta_i = 7, \tag{3}$$

$$\sum_{i=1}^{D} \eta_{d,i} = D \quad \forall d, \tag{4}$$

where *D* is the number of time intervals in a day. To ease the interpretation of these equations, we can define the remaining symbols as follows:

- λ_0 is the average (i.e. expected) rate of the Poisson process over a full week;
- δ_i is the relative change for the day ***i*** (Saturdays have lower day rates than Tuesdays);
- $\eta_{j,i}$ is the relative change for the period ***i*** on the day ***j*** (the peak hours);
- $\lambda(t)$ is a discrete function representing the expected demand of taxi services distribution over a period of time for a taxi stand of interest ***k.***

2.2 Weighted Time Varying Poisson Model

The model previously presented can be viewed as a time-dependent average. However, it is not guaranteed that every taxi stand will have a highly regular passenger demands: in fact, the demand at many stands can often be *seasonal*.

To face this specific issue, we propose a weighted average model based on the one already presented: our goal is to increase the relevance of the demand pattern observed in the previous week, by comparing it with the patterns observed several weeks ago (e.g. what happened on the previous Tuesday is more relevant than what happened two or three Tuesdays ago). The weight set ω is calculated using a well-known time series approach to these kind of problems: the Exponential Smoothing [8]. We can define ω as follows

$$\omega = \alpha * \{1, (1-\alpha), (1-\alpha)^2, \dots, (1-\alpha)^{\gamma-1}\}, \tag{5}$$

where γ is the number of historical periods considered in the initial average, α is the smoothing factor (i.e. a user-defined parameter) and $0 < \alpha < 1$.

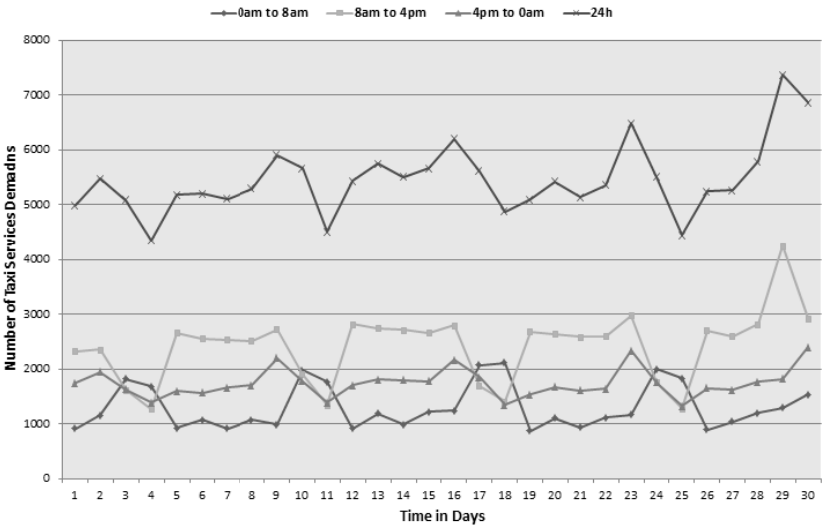


Fig. 2. One month data analysis (total and per driver shift)

2.3 Autoregressive Integrated Moving Average Model

The two previous models assume the existence of a regular (seasonal or not) periodicity in the taxi service passenger demand (i.e. the demand at one taxi stand on a regular Tuesday during a certain time period will be highly similar to the demand verified during the same time period on other Tuesdays).

However, the demand can present distinct periodicities for different stands. The ubiquitous characteristics of this network force us to rapidly decide if and how the model is changing. The Autoregressive Integrated Moving Average Model (ARIMA) [9] is a well-known methodology to both model and forecast univariate time series data such as traffic flow data [10], electricity pricing [11] and other short term prediction problems like our own. The ARIMA main advantage when compared to other algorithms is its versatility to represent very different types of time series: the autoregressive (AR) ones, the moving average ones (MA) and a combination of those two (ARMA). A brief presentation of one of the simplest ARIMA models (for non-seasonal stationary time series) is enunciated below following the existing description in [12] (however, our framework can also detect both seasonal and non-stationary ones). For a more detailed discussion, the reader should consult a comprehensive time series forecasting text such as Chapters 4 and 5 in [13].

In an autoregressive integrated moving average model, the future value of a variable is assumed to be a linear function of several past observations and random errors. The underlying process that generates the time series (taxi service over time for a given stand $\boldsymbol{k}$) can be formulate as

$$\begin{aligned} R_{k,t} = \theta_0 + \phi_1 X_{k,t-1} + \phi_2 X_{k,t-2} + \cdots + \phi_p X_{k,t-p} \\ + \varepsilon_{k,t} - \theta_1 X_{k,t-1} - \theta_2 X_{k,t-2} - \cdots - \theta_q X_{k,t-q} \end{aligned} \quad (6)$$

where $R_{k,t}$ and $\varepsilon_{k,t}$ are the actual value and random error at time period t, respectively; $\phi_l (l = 1,2, \dots, p)$ and $\theta_m (m = 0,1,2, \dots, q)$ are the model parameters/weights while p and q are positive integers often referred to as the order of the model. Both order and weights can be inferred from the historical time series using both the autocorrelation and partial autocorrelation functions as introduced by Box and Jenkins in [9]. They are useful to detect if the signal is periodic and, most important, which are the frequencies of these periodicities.

2.4 Sliding Window Ensemble Framework

In the last decade, regression and classification tasks on stream data have attracted the community attention due to its special characteristics: the traditional algorithms had computational efforts which were not compatible with the time available to make a decision. Fast and adaptive ensembles of these were also built. One of the most popular models is the weighted ensemble [14]. The model proposed below is also based on this one.

Consider $M = \{M_1, M_2, \dots, M_z\}$ to be a set of z models of interest to model a given time series and $M_t = \{M_{1t}, M_{2t}, \dots, M_{nt}\}$ to be the set of forecasted values to the next period on the interval t by those models. The ensemble forecast E_t is obtained as

$$E_t = \sum_{i=1}^{z} \frac{M_{it}}{\beta}, \qquad \beta = \sum_{i=1}^{z} \rho_{iH} \tag{7}$$

where ρ_{iH} is the forecasting error obtained for the model M_1 in the periods contained within the time window/period $[t - H, t]$ (H is a user-defined parameter to define the window size). As the information is arriving in a continuous manner for the next periods $\{t, t+1, t+2, \dots\}$ the window will also *slide* to determine how the models are performing in the *last H periods*. To calculate such error, we used a well-known time series forecasting error metric: the Symmetric Mean Percentage Error (*sMAPE*) [15].

3 Data Acquisition and Preprocessing

In this work, we studied the taxi driver mobility intelligence of one company operating in the city of Porto, Portugal. This city is the center of a medium size urban area (1.3 million inhabitants) where the passenger demand is lower than the number of running vacant taxis, provoking a huge competition between both companies and drivers. The existing regulations force the drivers to not run *randomly* in search of passengers but to choose a specific taxi stand out of the 63 existing ones in the city – see Fig. 3 to observe its spatial distribution - to go park and wait for the next service immediately after the last passenger drop-off. There are three main ways to pick-up a passenger: (I) a passenger goes to a taxi stand and picks-up a taxi – the regulations also force the passengers to pick-up the first taxi in the line (First In, First Out); (II) a passenger calls the taxi network central and demands a taxi for a specific location/time –parked taxis have priority over running vacant ones in the central taxi dispatch system; (III) a passenger picks a vacant taxi while it is going to a taxi stand, on any street.

Fig. 3. Taxi Stand spatial distribution over the city of Porto, Portugal

In this section, we describe the studied company and the data acquisition process and the preprocessing applied to it.

3.1 Data Acquisition

The data was continuously acquired using the telematics installed in each one of the 441 running vehicles (GPS and wireless communication) of the studied company. These vehicles usually run in one out of three 8h shifts: midnight to 8am, 8am-4pm and 4pm to midnight. Each data chunk arrives with the following six attributes: (1)

TYPE –relative to the type of event reported and has four possible values: busy – the driver picked-up a passenger; assign – the dispatch central assigned a service previously demanded; free – the driver dropped-off a passenger and park – the driver parked at a taxi stand. The attribute (2) STOP is an integer with the ID of the related taxi stand. The attribute (3) TIMESTAMP is the date/time in seconds of the event and the attribute (4) TAXI is the driver code; the attributes (5) and (6) refer to the LATITUDE and the LONGITUDE corresponding to the acquired GPS position.

This data was acquired over a non-stop period of 28 weeks. Our study just uses as input/output the services obtained directly at the stands or those automatically dispatched to the parked vehicles (more details in the section below). This was done as the passenger demand at each taxi stand is the main feature to aid the taxi drivers' decision.

3.2 Preprocessing and Data Analysis

As preprocessing, a time series of taxi demand services aggregated for a period of P-minutes was continuously built. There are three types of accounted events: (1) the *busy* directly set at a taxi stand; (2) the *assign* directly set to a taxi parked in a taxi stand and (3) the *busy* set while a vacant taxi is cruising. Type 1 events are accounted directly as type 2 events. However, for each type 2 event, the system receives a *busy* event a few minutes later – as soon as the driver effectively picked-up the passenger – this is ignored by our system. Type 3 events are ignored unless they occur in a radius of W meters from a taxi stand (where W is a user defined parameter). If it does, it is considered as being a type 1 event related with the nearest taxi stand according the

defined criteria. This was done because many regulations disallow to picking-up passengers in a pre-defined radius of a stop (in Porto a 50m radius is defined).

Table 1 details the number of taxi services demanded from any taxi stand per daily shift and day type. Table 2 has information about all services (independently of the pick-up place) per taxi and duration. The *service* column in Table 2 represents the number of services picked-up by taxi drivers, while the second one is related to the time distance of each service done.

Additionally, it could be stated that the central service assignment is 24% of the total service (*versus* the 76% of the one demanded directly in the street) while 77% of the service is demanded directly to taxis parked at a taxi stand (and 23% is picked-up while they are cruising). The average waiting time (to pick-up passengers) of a taxi parked at a taxi stand is 42 minutes while the average time distance for a service is only 11 minutes and 12 seconds.

The data presented in the tables highlight this, despite the regularity exhibited in the service (especially on the weekends), there are huge mobility intelligence discrepancies between the drivers (i.e. a large variance in both number and time distance of the services).

4 Experimental Results

In this section, we firstly describe the experimental setup developed to test our model on the available data. Secondly, we present and discuss the results achieved.

4.1 Experimental Setup

Our model produces an online forecast for the demand of taxi services at all taxi stands at each period of P-minutes. The scripts used were developed using the R statistical software [16].

The data was continuously acquired and processed through messages sent over a socket. The pre-defined functions used and the values set for the models parameters are detailed along this section.

An aggregation period of 30 minutes was set (i.e. a new forecast is produced each 30 minutes; P=30) and a radius of 100 meters (W = 100 > 50 defined by the existing regulations). This aggregation was set according the average waiting time at a taxi stand, i.e. the forecast horizon lower than 42 minutes.

Data was acquired over a period of 28 weeks. 20 weeks' worth of data was initially stored just to train the model. Then the number of services for each stand was continuously forecasted every 30 minutes over a total of 8 weeks. This framework continuously runs on a single computer using just one core (i.e. without any parallelization).

The ARIMA model (p,d,q values and seasonality) was firstly set (and updated each 24h exactly at 3am) by learning/detecting the underlying model running on the historical time series curve for each considered taxi stand. This was done by using an automatic time series function in the [forecast] R package [17] - `auto.arima` - with the default parameters. The weights/parameters are specifically fit for each period using the `arima` function from the built-in R package [stats].

Table 1. Taxi Services Volume (per daytype/Shift)

	Total Services Occurred	Averaged Service Demand per Shift		
		0am to 8am	8am to 4pm	4pm to 0am
Workdays	781398	1263	2227	1719
Weekends	289364	1048	2085	1534
Total	1070762	**2311**	**4312**	**3253**

Table 2. Taxi Services Volume (per Taxi/Duration)

	Services	Time Running Busy (minutes)
Máx.	5174	50605
Min.	38	384
Mean Total	1811	23424
Std. Dev. Total	816	9996

The time-varying Poisson averaged models (both weighted and non-weighted) were updated every 24 hours. A sliding window of 4 hours (H=8) was considered in the ensemble. The error of each model was measured using the metric also proposed to weight each model in the ensemble – sMAPE. Distinct results for two distinct values (0.4 and 0.5) of the parameter alpha (α in the weighted average) are presented below.

4.2 Results

The main results are presented in Table 3. A percentage is presented for each model corresponding to a similarity measure between the real service time series and the predicted one (i.e.: like a distance between the two time series calculated using the *sMAPE*). The results are firstly presented per shift and then globally. The values presented below are calculated through an average weight of the error obtained through each one of the time series (i.e. the forecast error of the demand for taxi services at each one of the 63 taxi stands). Each error rate was weighted according to the number of services demanded at the corresponding taxi stand along the whole test period.

Table 3. Models Success Rate per Shift

MODEL	PERIODS							
	alpha (α) = 0.4				*alpha (α) = 0.5*			
	00->08	08->16	16->00	24h	00->08	08->16	16->00	24h
Poisson Mean	79,03%	76,25%	76,79%	77,34%	79,03%	76,25%	76,79%	77,34%
W. Poisson Mean	78,15%	74,51%	75,25%	75,53%	77,05%	73,20%	73,99%	74,29%
ARIMA	79,01%	73,37%	76,79%	75,72%	79,01%	73,37%	76,79%	75,72%
Ensemble	**81,06%**	**76,87%**	**78,64%**	**78,36%**	**81,01%**	**76,73%**	**78,58%**	**78,26%**
Nr. Of Events	54.276	114.344	85.125	253.745	54.276	114.344	85.125	253.745

Each model returns a time series which has a similarity measure above 74% (the Weighted Poisson Mean and the Ensemble are only affected by the changes of the *alpha* (α) parameter) for the real one. The sliding window ensemble is always the best model for every shift and period considered, with a similarity measure greater than 78% (197921 of the 253745 total taxi services were correctly forecasted in both time and space using an aggregation of 30-minutes). Our framework also had a satisfactory performance in another important aspect: the computational time. It was implemented using an iterative process (i.e. a loop) forecasting the next value for each existing time series (for a total of 63). Such process took, in average, 99.77 seconds of processing time. The ARIMA model update (it is done every 24 hours) on average lasted 48.12 seconds.

5 Conclusions and Future Work

Sensor networks are providing a growing number of challenges to researchers along with an explosive number of opportunities for companies. In the last decade, the taxi industry has already been exploring intelligent data analysis on a daily basis (efficient taxi dispatching, time saving route finding, among others). This paper presents a novel application work where well-known time series forecasting models are adapted to predict the spatial distribution of the passenger demand for taxi services in a short term horizon (30 minutes periods). To do so, two distinct models have been assembled - the Time-Varying Poisson Model [5] and the ARIMA [9] one – using a Sliding Window Weighted Ensemble model [14] (i.e. it uses the error of each model through a pre-determined time window as their weight in the mean calculation). These models were chosen due to their well-known efficiency in short-term point forecast problems like our own. At our best knowledge, such an online predictive model for the passenger demand is a true novelty in this specific industry.

We tested it using the data received from a company network in the city of Porto, Portugal. The test ran continuously for the duration of 28 weeks over the existing 63 taxi stands.

The results are promising: the model correctly predicted 78% of the 253745 received services. Our iterative implementation took, an average, 99.77 seconds to produce a prediction (1.58 seconds for each taxi stand) for the next period. These numbers demonstrate that this model can be a major contribution to this industry, improving and aiding the drivers mobility intelligence in real time.

In the near future, we will improve this model in two ways: 1) using parallel computation to reduce the current processing time - the time series produced by each stand are independent from the remaining ones; 2) modeling the weather changes as a factor to change our predictions (as it is addressed by using Hidden Markov Models to handle the occurrence of bursty events in [5]).

It will also be used as a feature of a recommendation system (to be developed) which will produce smart live recommendations to taxi drivers about which taxi stand they should head to after a drop-off. This decision support framework will also address other features like the distance or live traffic conditions, among others.

Acknowledgements. This work is part-funded by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for

competitiveness), by the Portuguese Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701.

References

1. Glashenko, A., Ivaschenko, A., Rzevski, G., Skobelev, P.: Multiagent real time scheduling system for taxi companies. In: Proceedings of 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems, Budapest, Hungary (2009)
2. Lee, J., Park, G.-L., Kim, H., Yang, Y.-K., Kim, P., Kim, S.-W.: A Telematics Service System Based on the Linux Cluster. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part IV. LNCS, vol. 4490, pp. 660–667. Springer, Heidelberg (2007)
3. Junghoon, L., Inhye, S., Park, G.: Analysis of the Passenger Pick-Up Pattern for Taxi Location Recommendation. In: Conference on Networked Computing and Advanced Information Management, vol. 1, pp. 199–204 (2008)
4. Bin, L., Daqing, Z., Lin, S., Chao, C., Shijian, L., Guande, Q., Qiang, Y.: Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 63–68 (2011)
5. Ihler, A., Hutchins, J., Smyth, P.: Adaptive Event Detection with Time-Varying Poisson Processes. In: 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, pp. 207-216 (2006)
6. Box, G., Jenkins, G., Reinsel, G.: Time series analysis. Holden-day, San Francisco (1976)
7. Matias, L., Gama, J., Mendes-Moreira, J., Sousa, J.F.: Validation of both number and coverage of bus Schedules using AVL data. In: Proceedings of IEEE Conference on Intelligent Transportation Systems, Funchal, Portugal, pp. 131–136 (2010)
8. Holt, C.: Forecasting seasonals and trends by exponentially weighted moving averages. International Journal of Forecasting 20, 5–10 (2004)
9. Box, G., Pierce, D.: Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. Journal of the American Statistical Association 65, 1509–1526 (1970)
10. Williams, B., Hoel, L.: Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. Journal of Transportation Engineering 129, 664–672 (2003)
11. Contreras, J., Espinola, R., Nogales, F.J., Conejo, A.J.: ARIMA models to predict next-day electricity prices. IEEE Transactions on Power Systems 18, 1014–1020 (2003)
12. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 50, 159–175 (2003)
13. Cryer, J., Chan, K.: Time Series Analysis with Applications in R. Springer, USA (2008)
14. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235. ACM, Washington, DC (2003)
15. Makridakis, S., Hibon, M.: The M3-Competition: results, conclusions and implications. International Journal of Forecasting 16, 451–476 (2000)
16. R Development Core Team, R: A Language and Environment for Statistical Computing, Vienna, Austria (2005)
17. Yeasmin, K., Hyndman, R.: Automatic Time Series Forecasting: The *forecast* Package for R. Journal of Statistical Software 27 (2008)

Analysis of Noisy Biosignals for Musical Performance

Joonas Paalasmaa[1], David J. Murphy[2], and Ove Holmqvist

[1] Beddit.com Ltd., Espoo, Finland,
Department of Computer Science, University of Helsinki, Finland
joonas.paalasmaa@beddit.com
[2] Meld Ltd, Helsinki, Finland
david.murphy@meld.fi,
ove@ove.fi

Abstract. Biosignal sensors are now small, affordable, and wireless. We desire to include such sensors (e.g. heart rate, respiration, acceleration) in a live musical performance, which sets requirements on the reliability and variability of the data. Unfortunately the raw signals from such devices are unable to meet these requirements. We contribute our solutions for overcoming the shortcomings of these sensors in two parts. The first is an online data processing and analysis system, including on-line generative models that describe the signals but add consistency. The second is the end-to-end system for capturing wireless signal data for the analysis system and integrating the resulting output into a popular digital audio workstation in a very flexible manner conducive to live performance. We also explore the role of "analysis supervisor"—a member of the performing act who ensures that the results of biosignal analysis fall within the desired ranges to contribute to the music effectively.

Keywords: ECG, respiration, accelerometer, music, performance.

1 Introduction

This paper presents a novel way to integrate biosignal measurement into live musical performance. We present a complete system from signal acquisition, signal analysis, data routing, to integrating the signals into instruments and effects in a digital audio workstation (DAW).

The motivation for our work comes from the desire to utilize inexpensive and convenient biosignal sensors in musical performance. This raised two technical challenges. The sensors themselves provide noisy and unreliable measurement that cannot be directly utilized in musical use. For example, directly using the heartbeat markers from a noisy electrocardiography (ECG) signal to trigger notes is not possible, because missing and spurious heartbeats will disturb the rhythm. The second challenge is that biological data has high-variability in both timing and magnitude, and incorporating biosignals into a composition can entail massaging the data. We needed an online solution to both of these challenges to facilitate live performance.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 241–252, 2012.
© Springer-Verlag Berlin Heidelberg 2012

The contribution of this paper is twofold. First, we present biosignal analysis methods that turn noisy measurements of irregular biosignals into consistent and uninterrupted outputs, which are suitable for musical use. Second, we present a practical implementation for integrating the analysis into live musical performance.

We have tried to build the analysis system so that it fullfills the following requirements. First, it should be usable by a musician: it should have a sufficient range of output to be expressive, it should be robust in the face of noisy biosignals, and its working mechanism must be learnable, with a clear mental model of operation.

Second, it has to be possible to compose for it, such that a performer can communicate the composer's intent faithfully. There is a tendency to use physiological signals in electronic music as audio generators, that is, the signals themselves are directly translated into sounds as synthesized audio or triggered samples. It is very difficult to purposefully convey intent within that framework. We see a distinction between the direct sonification of biosignals, and their integration into a musical performance. If a conflict arises between expressing the biosignals with high fidelity, or achieving the intent of the composer with high fidelity, our bias is towards the latter.

Finally, biosignal measurement should contribute towards the musical performance, not dictate it: we have elected to make the output of signal analysis malleable, and envisage one member of the group in the role of the "analysis supervisor", carefully monitoring and if necessary shaping the physiological data to suit the desired expression of the piece. For example, signals may be of higher magnitude than anticipated due to the excitement of performance, and require scaling down to fall within the required range for modulating a parameter such as a synthesizer cut-off frequency.

A video showcasing our approach with a solo performance is available online[1]. In the video, it is shown how a synthesizer can be controlled with biosignal measurements.

2 Background

Generating audio and music from physiological signals has been the subject of research since the invention of the encephalophone in the 1940s at the University of Edinburgh [1]. It generated audio from measured electroencephalography (EEG) signals. Various artists have since used brainwaves in music. Krzysztof Penderecki's *Polymorphia* (1963) used pitch notation derived from EEG data from patients listening to a recording of his *Threnody for the Victims of Hiroshima* (1961).

[1] `http://vimeo.com/42032074` — The performer triggers the synthesizer with heartbeats and modulates the filter envelope amount with respiration. Tilting the body back and forth adjusts the low frequency oscillation rate, while bending sideways controls the arpeggiator distance.

Alvin Lucier's *Music for Solo Performer* (1965) is considered to be the first music performance to use real-time biosignals, where EEG triggered percussive instruments through sympathetic resonance [2]. Richard Teitelbaum's *In Tune* and *Organ Music* (1967/1968) used EEG, ECG and respiration measurement [3]. The research in biomusic by Manford L. Eaton [4] inspired Erkki Kurenniemi to build instruments based on galvanic skin response (GSR) and EEG in the early 1970s [5]. In 1973, Pierre Henry performed his EEG-based piece *Cortical Art III*. David Rosenboom used neurofeedback and algorithmic composition in *Brainwave Music* (1976) and *On being invisible* (1977), which compared performers' brain activity with previously stored patterns in real time [6].

The first commercial biomusic production environment, *BioMuse*, was introduced in 1992 [7,8].

Various contemporary projects have used EEG and ECG in music generation. *The Multimodal Brain Orchestra* measures EEG event related potentials, with an "emotional conductor" controlling the "affective expression" of the performance [9]. Brain computer interfaces (BCI) enable disabled people to perform music with EEG [10]. In *DECONcert*, EEG and ECG of 48 participants were monitored and methods such as signal averaging and detection of collective alpha synchronization were used in order to facilitate collaborative biofeedback in regeneration of the music [11]. A traditional chamber music performance has been augmented by EEG and ECG signals [12].

With the interactive cinema environment *Biosuite*, the audience's emotional responses are measured using ECG and GSR in order to influence cinematic events [13]. In another project, ECG measurement from healthy and diseased hearts are mapped into musical notes, in order to highlight their differences[2]. In [14], respiration and cardiac events are translated into synthesized sound. The timbral brightness of the sound responds to respiratory sinus arrhythmia.

Apart from the academic research, there are performing musicians who utilize biosignals to varying degrees. *Lucky Dragons* use a GSR-touch interface to connect with their audience. *The Heart Chamber Orchestra* measures the orchestra's ECG and presents the data to them as notation, which they subsequently perform. The dance and performance company *Manifold Motion* uses real-time ECG biofeedback in some of their shows.

3 Biosignal Analysis System

3.1 Signal Acquisition

Our system captures biosignals with two Bluetooth chest sensor belt models from Zephyr Technology Corporation[3]. Zephyr BioHarness provides a range of data including heartbeat events, a respiration effort signal and a 3-axis accelerometer signal. Zephyr HxM provides only heartbeat events and a rough estimate of activity level, but is more affordable. The signals from the sensors are transmitted

[2] `http://polymer.bu.edu/music/`
[3] `http://www.zephyr-technology.com/`

over Bluetooth to a computer where they are processed and forwarded to the DAW. We have implemented drivers for the Bluetooth protocols of the devices in Python (source code is available for download[4]). The implementation turned out to be surprisingly complex, because issues such as varying latency and connection breaks need to be taken into account.

The Zephyr devices transmit the signal data in packets. For example, the acceleration signal is sampled at 50 Hz and transmitted over Bluetooth in 20-sample packets. That creates a 0.4-second measurement latency for acceleration data. The Bluetooth connection can occasionally drop out completely, which has required us to implement reconnection logic, so that gaps in measurement during a performance would be as short as possible.

The latency problem could be avoided with a device that has a smaller packet size. However, the latency is in practice tolerable so we have not tried to find alternative devices. Replacing Bluetooth with a wired connection would improve the reliability of the connection, but would also impair the usability of the system in musical performance use too much.

3.2 Dealing with Noise and Missing Data

The way the physiological parameters are used in a performance motivates a signal analysis approach where noise and missing values are filled in with generated information.

For example, heartbeat events are primarily used as sample (e.g. drum) triggers. If the measurement of a heartbeat is missed, a triggering of the drum will be missing, and that disturbs the performance. Therefore, missing and corrupted measurements need to be filled in in a way that is musically sensible. That process is mostly automatic, but may at times require human intervention.

The mediation of signal analysis is performed by an "analysis supervisor", who is one of the musicians in the performance and knows a musically optimal way to massage the data, having practiced the performance with the other musicians, and being keenly aware of the moment-by-moment intent.

The flow of physiological measurement information during a performance is summarized in Figure 1. It can be described in three steps:

1. Biosignals are measured (ECG, respiration effort and acceleration) from one or more people.
2. The biosignals are analyzed so that noise and missing data is taken into account. If needed, the analysis is mediated by the analysis supervisor. This produces consistent results for the physiological parameters (e.g. heart rate, respiration signal and activity level).
3. A musician uses the physiological parameters in real-time as part of the performance.

[4] `https://github.com/jpaalasm/zephyr-bt`

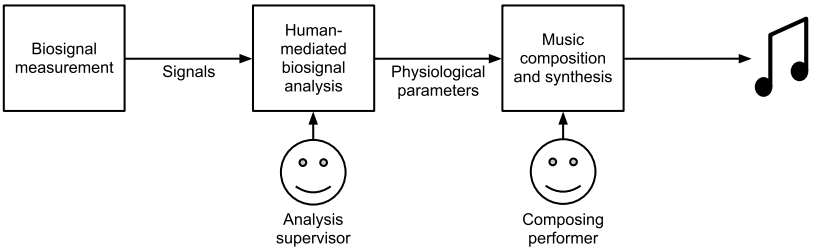


Fig. 1. The flow of information during a performance

3.3 Heartbeat Interval Analysis

A healthy human heart beats at intervals that vary relatively little. For example, at rest, the interval between successive heartbeats might vary, say, between 0.8 and 1.2 seconds, corresponding to around 60 beats per minute (BPM). When heartbeats are detected from a noisy ECG signal, there can be incorrectly detected heartbeats and several-second long gaps in heartbeat detection. When a chest heart rate belt is used, measurement errors are frequent since signal quality is relatively low.

The aim is to output a consistent sequence of heartbeat events from ECG measurement, so that the sequence of events does not have incorrect heartbeats or measurement gaps. The analysis is mostly automatic, and the analysis supervisor just sets upper and lower bounds for allowable heartbeat intervals. That information is needed in cases where signal quality is bad.

The heartbeat interval analysis procedure is visualized in Figure 2. First, heartbeat intervals are analyzed by an automatic ECG heartbeat detection algorithm, which tries to find heartbeat intervals from the signal. The result of that analysis may contain some erroneous values, but the detected heartbeat intervals should mostly be correct.

Then, the intervals are manually limited to a range by the analysis supervisor. Two erroneous short heartbeat intervals are shown in the second row of Figure 2 at time 13 seconds. They are discarded by the manual limitation, because they are outside the specified range. If the previous step in the analysis works correctly, no manual limitation is needed.

Last, a consistent heartbeat sequence is generated to fill the gaps in the heartbeat interval data. A consistent heartbeat sequence is created by generating synthetic heartbeat positions to the gaps in the heartbeat sequence from the previous step. The heartbeat positions are generated from an Inverse gaussian distribution with a manually specified variability parameter λ [15]. The end result is in Figure 2d.

With a good-quality measurement, the heartbeat events go through the three steps of analysis unchanged, as no heartbeat intervals need to be discarded and no gaps need to be filled.

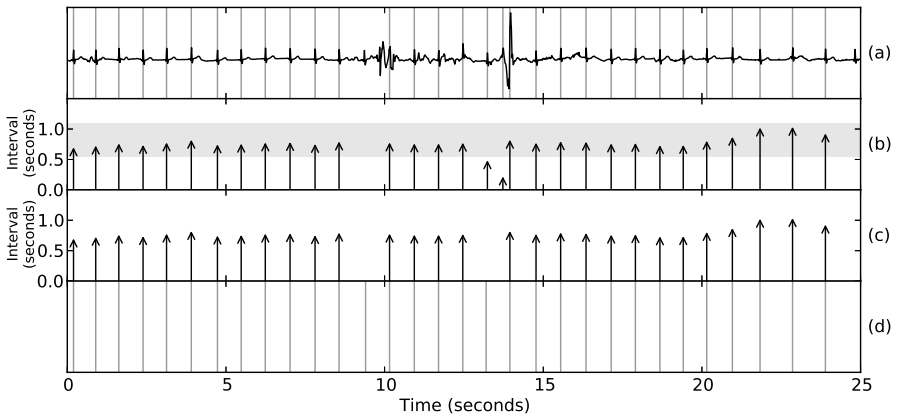


Fig. 2. The analysis of heartbeat intervals. In (a), an ECG signal with detected heartbeat positions is shown. The corresponsing heartbeat intervals are in (b). The analysis supervisor limits heartbeat intervals to range 0.55...1.1 seconds, to discard erroneous heartbeat intervals, and the result of that is in (c). The resulting consistent heartbeat event sequence is in (d).

3.4 Respiration Signal Analysis

The BioHarness sensor measures a chest respiration effort signal. Variation in signal quality is large. The signal sometimes tracks respiration in a sinusoidal manner, but, at other times, has no clear structure. We have not found out why the signal quality varies so much.

The analysis of the respiration signal consists of defining the output as a linear combination of the measured respiration effort signal and a generated synthetic sinusoidal. The amplitude, frequency and phase of the synthetic signal are estimated from the measured signal, so that is is possible to transition as smoothly as possible between the measurement and the synthetic signal.

The task of the analysis supervisor is to continuously adjust a weight parameter that defines the linear combination. The manually chosen weight is a number between 0 and 1, and values in the output signal are simply generated as

$$output = measured \times weight + synthetic \times (1 - weight).$$

This means that the measurement is output when $weight = 1$ and the sinusoidal when $weight = 0$. Values between 0 and 1 output a mixture of the signals.

Manual weighting between the real measurement and a synthetic signal enables smooth swithing between them during the performance, when signal quality changes. That is visualized in Figure 3.

3.5 Accelerometer Signal Analysis

The three-axis accelerometer signals from BioHarness include information about the posture and activity level of the measured person. Posture is measured as two

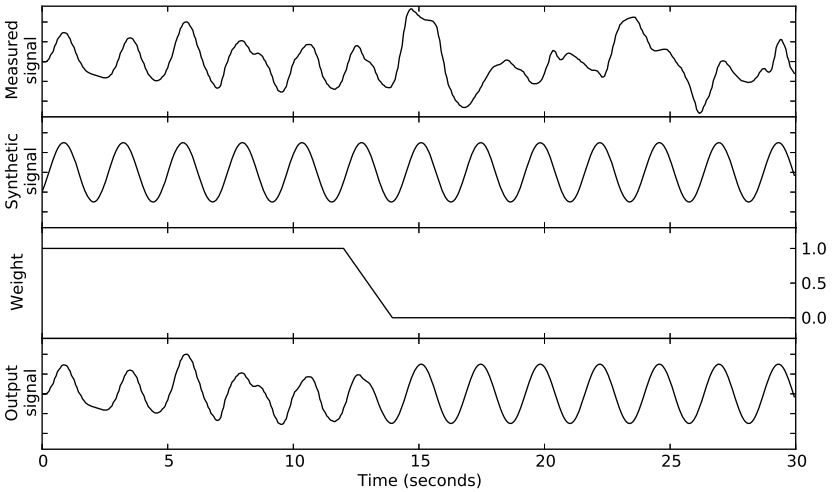


Fig. 3. A measured respiration signal is shown in the topmost plot, and a corresponding synthetic sinusoidal in the second plot. The output of the analysis is a weighted sum of the signals. Smooth switching between the signals happens at time 12...14 seconds.

parameters: tilt (back and forth) and sway (sideways), by simply calculating the angles of the accelerometer's axes in space. The activity level is calculated with a moving average of the high-pass filtered and squared accelerometer signal. The high-pass filtering and squaring is done for each axis separately, and the resulting three values are summed to give the activity level. The length of the moving average is controlled by the analysis supervisor, and can vary between 0.1 seconds to 10 seconds. The activity analysis is visualized in Figure 4.

3.6 Networking

The biosignal analysis and music workstation software may run on different computers, so there needs to be a reliable way to transmit the physiological parameters between them. Our goal was to create a system where all parameters are available to all music workstations in the performance, regardless of the number of devices involved in capturing the sensor data.

The solution we have settled on is broadcasting the data as Open Sound Control (OSC) messages [16] over User Datagram Protocol. In the composition and synthesis environment, performers "tune in" to a physiological parameter of interest based on OSC message tags. Using OSC requires us to create a wired or wireless network for our participating computers. Basic multi-computer performance functionality such as synchronizing MIDI clocks depends on being networked, so this does not introduce a new requirement to the performing environment.

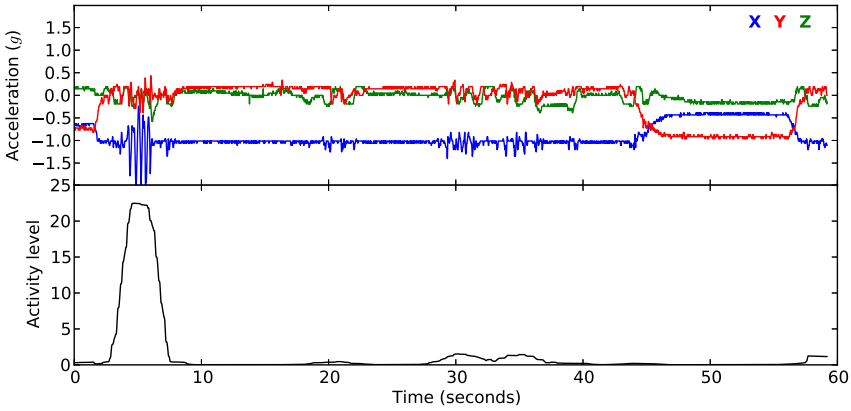


Fig. 4. Activity analysis example. The upper plot contains the three-axis acceleration signals, with orthogonal measurement directions X, Y and Z. An activity value signal calculated with a three-second moving average is shown in the lower plot.

4 Performance

The performance is the ultimate goal of our work. As digital technology makes recorded music ubiquitous, the importance of the performance as a unique event becomes elevated. We hope that linking the music to the musician's physiological signals will result in a performance that feels intimate, unique and a little magical.

A challenge with performances utilizing biosignals is the evolving and unpredictable nature of the signals, and the burden that monitoring for bad signals places on a performer. Viewed over an entire performance, the rates and magnitudes of a given signal can change drastically, or the signal may be lost entirely. Our solution to the problem of managing the data flow in the service of the performance is the role of the analysis supervisor, who ensures that other musicians do not have to worry about problems related to the signals.

We feel it is important to make the link between the physiological signals and the resulting sounds as direct as possible, so an audience is not mystified by how the sound is being produced, which has led us to favour straightforward translations from signal to sound. Mappings that we consider intuitive include:

- triggering drums with heartbeat events
- matching the performance tempo to heart rate
- modulating synthesizer parameters to reflect respiration phase
- modulating synthesizer parameters such as pitch-bend using pose
- modulating the frequency spectrum of the performance using activity level

We have created a functional music production environment that allows us to make performances based around the mappings we have defined as being intuitive to us. The system uses Cycling 74's MaxForLive[5] plug-ins for the popular

[5] `http://cycling74.com/products/maxforlive/`

Ableton Live digital audio workstation. Ableton Live organizes music performances into tracks, with one track per instrument. Each track can have a rack of modulating effects. Using MaxForLive, we can create instruments that can be embedded in Live's effect racks, and we can stack as many effects per rack as we like. Thanks to our "tunable" broadcast network architecture, any performer's signals can be used by any instrument.

The most basic instrument we have simply assigns the BPM of the track to the BPM of a measured heart. In this manner it is very trivial for us to control the tempo of the performance using one of the performers' heart rates. By carefully controlling their exertion, a performer is able to raise or lower the tempo within reasonable bounds.

The Modulator instrument accepts data from our signal sources and performs some basic mapping: a higher and lower bound for the output can be set, and a squashing function is provided, which allows a signal to be mapped to all or part of the MIDI range for a selected parameter, and furthermore be amplified or diminished depending on the desire of the performer. The output can be dynamically assigned to any suitable parameter in the rack. This is very flexible—we can use it to assign a setting value to any exposed user interface element on any instrument or effect in any track, and we can have multiple instances per track. In this way we have been able to create instruments where performer body position can be used to alter the cut-off frequency on a modelled analog synthesizer, and passages where instrument volume is effected by performer activity level. See Figure 5.

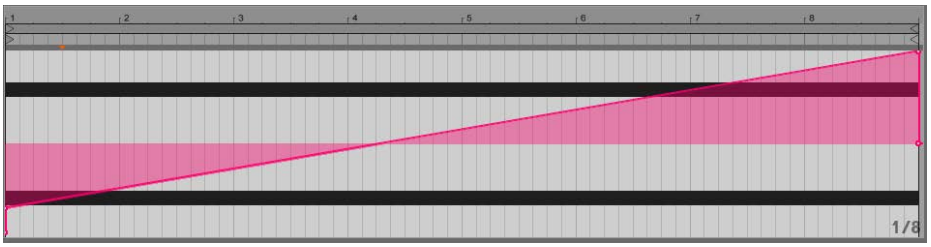


Fig. 5. An example phrase composed for keyboard and biosensor - the horizontal lines indicate two held notes, and the rising graph indicates the composed value of the parameter "activity level". The parameter is linked to instrument volume. In the performance the performer must try to increase their activity for the duration of this phrase, so the instrument swells in. The analysis supervisor must scale the output of "activity level" so it measures from -50% to +50%.

The Beat Train instrument is designed to work with heartbeat events and generates an assignable MIDI event when a heartbeat event is received. If the heartbeat events are not aligned with the "beat grid" of the track, the input will not be "in time" (in a musical sense). To compensate for this, the Beat Train has a mode for automatically quantizing the incoming events to the nearest

measure in the track. The value given for measure effectively sets the width of the quantization grid: assigning a quantization measure of one measure when the piece is being performed in Common or 4/4 time (4 quarter beats to a bar) means events will be delayed to occur "on the beat". Progressively finer quantization grids allow more freedom. We find using a grid of 8th or 16th notes pleasing, as at that resolution the natural heart rate variability introduces some randomness and the heartbeat events still feel biological rather than robotic, yet they stay aligned with the track.

The Selector instrument allows us to choose between pre-recorded phrases of music depending on a given input. We have designed this instrument for alternating between looped backing tracks based on the most easily controlled posture parameters. In one instance we have made two variations of a phrase, one with an ascending intonation at the end, the other that descends and resolves. By assigning phrase selection to whether a performer is leaning left or right, a performer can freely improvise on another instrument using other physiological signals, and control whether the phrase repeats, or resolves based on body position at the end of the phrase, in a manner that is readily observable by fellow performers.

The biggest single issue in performing with the system is latency. Due to the Bluetooth protocol, there is an unavoidable minimum latency of about a second. Normally, latencies in musical inputs are measured in tenths of seconds, so at first glance a latency of this magnitude would seem to be a major difficulty. However, we have found that by performing in a roundelay manner, where the same pattern or a similar pattern is repeated at intervals, it is possible (and natural) for musicians to get "in the groove" using the system. In effect the musician plays the same phrase, making the bodily movements necessary for the physiological instrument to produce the desired intent, with the signals being effectively delayed one whole phrase. Posture and activity data are therefore used as "ambient" biosignals, as opposed to being an effort towards gestural control.

Finally, we have found that it is good to structure the performance to introduce the physiological elements one by one, to educate an audience, typically leading from heartbeat-triggered samples to activity to respiration.

5 Conclusion

Our work lies at the intersection of biosignal analysis, music performance, and music theory. We find this a very exciting area to explore. There is potential for signal analysis techniques to aid us in our quest to compose and perform music informed by live physiological data.

In summary, we have created a working system comprising multiple body-sensors, data-analysis components, and software that integrates the signals into Ableton Live for live performance. Our system functions wirelessly, and is built in as decoupled a manner as possible. The data analysis is driven entirely by the motivations of live performances, which has led to novel approaches to the problem of turning raw sensor data into data streams that are robust, consistent

and suitable for musical performance, while balancing a faithfulness to the composition with fidelity to the source signals. Working with the system in practice has led us to consider a role for one of the performers, the analysis supervisor, in shaping and editing the data in the service of the performance.

Future development may involve the addition of other wireless sensors, such as galvanic skin response sensors (for emotional responses) and gyroscopes (for more accurate posture analysis), but availability and costs have been limiting factors. Including some existing technologies, such as Kinect or video based systems, would allow for immediate gestural expressions to complement existing biosignals.

Our initial concept sketches envisaged monitoring audience members, for use as another input or as a feedback mechanism. The focus to date has been on making the system work for the performers, so we have neglected this aspect of the system. It is interesting to consider what parameters could be extracted from an audience using the same recording equipment as the performers.

References

1. Henry, T.K.: Invention locates hurt brain cells. New York Times, 21 (March 2, 1943)
2. Lucier, A.: Statement on: Music for solo performer. In: Rosenboom, D. (ed.) Biofeedback and the Arts, Results of Early Experiments. Aesthetic Research Centre of Canada Publications (1976)
3. Teitelbaum, R.: In tune: Some early experiments in biofeedback music. In: Rosenboom, D. (ed.) Biofeedback and the Arts, Results of Early Experiments. Aesthetic Research Centre of Canada Publications (1976)
4. Eaton, M.L.: Bio-music: biological feedback experimental music systems. Something Else Press (1971)
5. Ojanen, M., Suominen, J., Kallio, T., Lassfolk, K.: Design principles and user interfaces of Erkki Kurenniemi's electronic musical instruments of the 1960's and 1970's. In: NIME 2007: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 88–93 (2007)
6. Rosenboom, D.: Extended musical interface with the human nervous system. Leonardo Monograph Series, vol. 1. International Society for the Arts, Sciences and Technology (1990)
7. Knapp, R.B., Lusted, H.S.: A bioelectric controller for computer music applications. Computer Music Journal 14(1), 42–47 (1990)
8. Knapp, R., Jaimovich, J., Coghlan, N.: Measurement of motion and emotion during musical performance. In: ACII 2009: 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, pp. 1–5 (2009)
9. Le Groux, S., Manzolli, J., Verschure, P.F.M.J.: Disembodied and collaborative musical interaction in the multimodal brain orchestra. In: NIME 2010: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 309–314 (2010)
10. Miranda, E.R., Brouse, A.: Interfacing the brain directly with musical systems: On developing systems for making music with brain signals. Leonardo 38(4), 331–336 (2005)

11. Mann, S., Fung, J., Garten, A.: DECONcert: bathing in the light, sound, and waters of the musical brainbaths. In: ICMC 2007: International Computer Music Conference (2007)
12. Barrass, S.: Sonifications for concert and live performance. AI & Society 27(2), 281–283 (2012)
13. Pérez, M.A.O., Knapp, R.B.: BioTools: A Biosignal Toolbox for Composers and Performers. In: Kronland-Martinet, R., Ystad, S., Jensen, K. (eds.) CMMR 2007. LNCS, vol. 4969, pp. 441–452. Springer, Heidelberg (2008)
14. Neumark, N., Khut, G.: Cardiomorphologies: An inner journey through art. IEEE Multimedia 14(4), 5–7 (2007)
15. Barbieri, R., Matten, E.C., Alabi, A.A., Brown, E.N.: A point-process model of human heartbeat intervals: new definitions of heart rate and heart rate variability. American Journal of Physiology - Heart and Circulatory Physiology 288(1), H424–H435 (2005)
16. Freed, A., Schmeder, A.: Features and future of Open Sound Control version 1.1 for NIME. In: NIME 2009: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 116–120 (2009)

Information Theoretic Approach to Improve Performance of Networked Control Systems

Marko Paavola, Mika Ruusunen, Aki Sorsa, and Kauko Leiviskä

University of Oulu, Control Engineering Laboratory
P.O. Box 4300, FIN-90014 University of Oulu, Finland
{marko.paavola,mika.ruusunen,aki.sorsa,kauko.leiviska}@oulu.fi

Abstract. Networked control systems (NCS) could be utilised in several industrial applications. However, the variable time delays introduced by the network impair the NCS performance, resulting even in the instability of the controlled process. To mitigate the delay problems, the advantage is taken from model-based, adaptive controllers. This calls for an efficient approach for on-line analysis of measurements applied to update the controller state in NCS. The paper introduces a new adaptive Model Predictive Controller (MPC) capable of compensating for variations in measurement and actuating delays. Weighting factors for delayed measurements and actuators are adjusted based on normalised version of mutual information that is calculated using a procedure described in the paper. The method is superior compared with other, more usual, metrics.

Keywords: information theory, adaptive control, model predictive control.

1 Introduction

Networked Control Systems (NCS) are spatially distributed systems composed of the process to be controlled together with its actuators, sensors, and controllers [1,2]. The communication between system components is carried out via a shared band-limited digital communication network. Networked control, be it wired or wireless, sets some new challenges for the design and performance of the control systems, network delay among others. Depending on the network structure, media and protocol, the delay can be constant, time varying or even random, and it occurs when sensors, actuators, controllers and humans exchange data over the network [3]. Available constant time-delay control methodologies may not be directly suitable for controlling a system over the network since the delays are usually time-varying, especially in the Internet [4] and in wireless systems, such as WSNs [5]. Namely, the variable delays may impair the performance of a control loop and can even destabilise the system.

The strategies for coping with the variable time delay can be divided to the averaging, identification and adaptation methods [6]. As a drawback of these approaches, time stamp data which may not always be available, especially in case of actuator signals (see Fig. 1), is required. Moreover, assumptions about the underlying distributions of the data or parametric models may be needed.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 253–263, 2012.
© Springer-Verlag Berlin Heidelberg 2012

This paper describes an approach incorporating a novel, advanced data stream analysis method and new adaptation algorithms for improving the performance of the NCS in the presence of variable time delay. To overcome the above mentioned drawbacks, this paper proposes a probabilistic, non-parametric index that is based on calculation of information theoretic quantities. The index quantifies the deterioration level of NCS performance and it is applied in tuning the adaptive weights on the measured outputs and manipulated variables. The controller behaviour is then adjusted according to the adaptation mechanism in order to ensure end product quality and process stability. Combined with a conventional MPC, the proposed algorithm is superior compared with other approaches.

The paper is organised as follows. In Section 2, the studied system is presented together with the proposed information theoretic index named the distance index and its usage in a controller adaptation case. Section 3 gives the results from a simulation case study and compares the proposed index to other indices used for similar purposes. Section 4 concludes the paper.

2 Materials and Methods

2.1 System Architecture and Process Description

Fig. 1 presents the NCS architecture applied in the case study (for more detailed description about the different architectures, the reader is referred to [4]). In the system, the process measurements to the controller and the actuator signals to the process are transferred via a network. During this, a random time delay is introduced to measurements. The delay causes disturbances to the measurement and actuator signals, which in turn impair the controller performance and may even lead to process instability.

To avoid degradation of the system performance, adaptive controller is utilised. The key idea in controller adaptation in this study is to quantify the effect of the disturbances to the process control applying the distance index described later. Based on the index, the parameters of the MPC are updated to cope with the current operating conditions. The controller then calculates the actuator signals.

The case study utilises a model predictive control demo available in Matlab© (R2011b): the control of a thermo mechanical pulp (TMP) plant. A more detailed description about the process can be found in [7]. The process has five actuator signals (manipulated variables, MV) and six measurement signals (measured outputs, MO). This study uses only one MV (primary dilution flow set point, PDFSP) and two MOs (primary motor load, PML, and the primary consistency, PC) as shown in Fig. 2. The variable time delay simulating the networked communications is added to the PML and PDFSP channels. The random delays for the communication channels were drawn from uniform distribution and were: PDFSP 0-2 s (low disturbance) and 0-20 s (high disturbance), PML 0-30 s (both cases) and none for all the other process input and output variables. The adaptation is applied for the variables that are transferred through the network. This is carried out with the weights (w_i and w_j) (see Fig. 2) based on the proposed index.

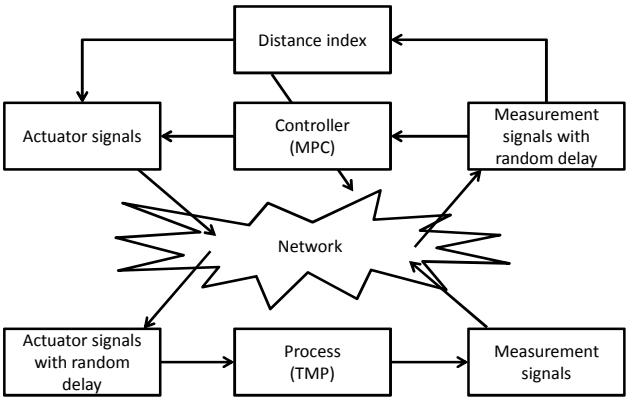


Fig. 1. The networked control system applied in the case study. The network introduces variable time delay both to the measurement and actuator signals, resulting into controller performance degradation. To overcome the problem, the weights of the controller are tuned based on the distance index.

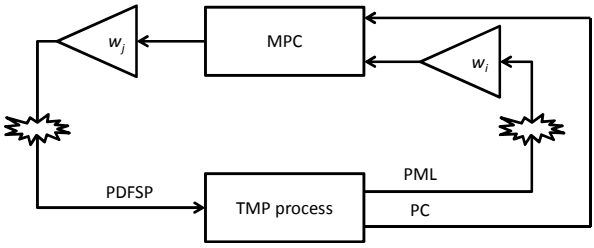


Fig. 2. The studied TMP process control scheme. The variables PDFSP and PML are transferred through the network and are thus subjected to a variable time delay.

2.2 Derivation of the Distance Index

The realised data stream from the network was analysed in sliding windows. There, an information theoretic metric was applied. The presented distance index is based on calculation of a normalised version of the mutual information $I(x(t);x(t\text{-}1))$ with the signal x itself at successive time instances. Specifically, an information theoretic interpretation of Jaccard index [8] was applied in windowed data analysis

$$D(x(t), x(t-1)) = 1 - \frac{I(x(t); x(t-1))}{H(x(t), x(t-1))}, \tag{1}$$

where $D(x(t),x(t\text{-}1))$ is the distance index for describing the similarity of vectors $x(t)$ and $x(t\text{-}1)$, presented with the mutual information and joint entropy $H(x(t),x(t\text{-}1))$, and $0 \leq D(x(t),x(t\text{-}1)) \leq 1$. The bivariate joint entropy is calculated with the signal entropies and the mutual information.

$$H(x(t), x(t-1)) = H(x(t)) + H(x(t-1)) - I(x(t); x(t-1)). \quad (2)$$

The approach is motivated by its properties, namely being a model-free framework to monitor any, possibly nonlinear, changes in correlation between successive data stream instances. The normalisation is applied in order to provide bias compensation of the mutual information and for achieving comparable index values [9]. The joint entropy (Eq. 2) describes here the uncertainty related to data set as a whole. Importantly, the normalised mutual information (Eq. 1) fulfils the requirements of a metric that has non-negative values at range of [0,1]. Data analysis in relative short and overlapping time windows was applied in order to avoid bias introduced to calculation of signal entropies [10] and to assume the i.i.d. random variables. This way also the stationary property was argued to mainly hold as indicated in [11]. In turn, this assumption is connected to the requirement that there should be present an ergodic process, where statistical properties of a data set can be derived from a single subsample. Even if the assumptions about data properties would be partly violated, the information theory may give indications about the strength of the signal dependency [12]. The distance analysis focusing on sequential data stream instances is inspired by time-delayed mutual information (TDMI) by Fraser & Swinney [13], where the calculation of mutual information between the sequential instances is performed for the whole data set.

The construction of the index requires estimation of the bivariate joint probability density functions (pdf's) and marginal probabilities of the signal and its delayed value. For this, histograms were utilised. Then, the calculation of the distance index proceeded as follows: calculation of signal entropies $H(x(t))$ together with $H(x(t\text{-}1))$ and the joint entropy $H(x(t),x(t\text{-}1))$, and finally the index according to (Eq. 1) in each time window that may overlap. The results are comparable index values along the data stream.

2.3 Adaptation Algorithms

As mentioned above, the adaptation is based on the ability of the proposed index to detect patterns in the signal. The index is applied in tuning the weights on the measured outputs and manipulated variables. A low index value indicates patterns in the signal resulting from excessive control actions or corrections carried out by the controller (such as after a step change in the set point). A high index value, on the other hand, indicates that the signal includes only uncontrollable random process noise. The adaptation of the weights also takes into account the control error between the measured outputs and their set points. Based on these, the reasoning behind the adaptation procedure is as follows:

- Measured outputs: The weights are increased, if the index value increases and decreased with decreased index values in order to reduce the controller actions. The increasing control error also increases the weight value, thus adapting the tuning towards the tighter control of the output.

- Manipulated variables: Contrary to the measured outputs, the weights on the manipulated variables are increased when the index value gets smaller to limit the excessive control actions and vice versa. The increasing control error decreases the rate weights of the manipulated variables in order to allow faster control e.g. in the case of set point changes.

Finally, adaptation scaling factors are applied to maintain the original magnitudes of the weights. For example, the scaling factors could be defined so that when no disturbance is present, the weights calculated by the adaptation algorithm would be approximately the same as the default ones. The resulting equations for the weights of the measured outputs (Eq. 3) and the rate weights of the manipulated variables (Eq. 4) are as follows:

$$w_i(t) = F_i D_i(t)\left(1 + \left|e_i(t)\right|\right) \text{ and} \tag{3}$$

$$w_j(t) = F_j\left\{\left(1 - D_j(t)\right) - \frac{1}{n}\sum_{i=1}^{n} D_i(t)\left|e_i(t)\right|\right\}. \tag{4}$$

In Eq. 3 and Eq. 4, w refers to weights, F to scaling factors, e to the control error between the set point and the measured output and D to the proposed distance index. Subscripts i and j refer to measured outputs and manipulated variables, respectively. In Eq. 4, the contribution from the error terms is divided by n, where n refers to the number of measured outputs that are dependent on the measured variable. The division is carried out to limit the effect of the error terms on the weights when n grows larger than 1.

3 Results and Discussion

3.1 Parameter Adaptation in the Presence of Disturbances

The study focuses on a single MO case, using the PML as an example (Fig. 2). The weight of the PML was adapted based on its distance index value and the control error (Eq. 3). Correspondingly, the weight of a single MV, PDFSP is tuned, utilising the error term from PML and PC according to Eq. 4.

The performance of the controller (Dist w/ Adapt) at low and high levels of communication disturbance is presented in Fig. 3 and Fig. 4, respectively. Also the case of no disturbance (w/o Dist) is presented in the figures. As shown in Fig. 3, the controller is able to follow the set point even without the adaptation (Dist w/o Adapt). However, compared to the “without disturbance” -case, the variation of the PML is significant, and the process could even become unstable (Fig. 3a). The effect of the low level disturbance is also clearly visible in the actuator signal (Fig. 3b), for which the variability seems to depend on the process state: at the end of the sequence, where the PML has a higher set point, the effect of disturbance on the PDFSP increases.

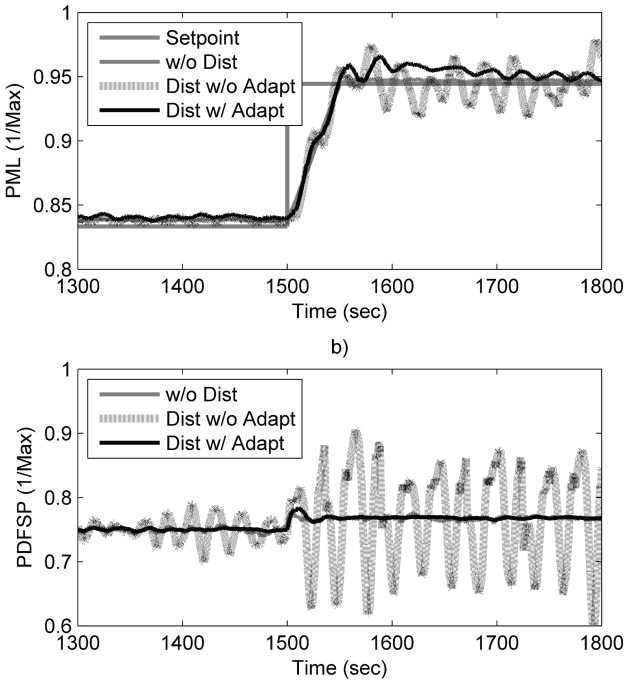


Fig. 3. Control performance in the presence of the low communications disturbance in the actuator signal and some variation in the delay of the process output measurement. a) Measured output (PML) b) Manipulated variable (PDFSP).

The adaptation clearly reduces the variation in both the process output and in the actuator signal (Fig. 3). Compared to the "without disturbance" -case, the settling time of the PML is, however, longer due to the relaxed weight. After the step change, the adaptation of the weight also reduces the variability and the control error in the PML being almost able to compensate for the effect of the disturbances.

Fig. 4 presents the case with high disturbance in the PDFSP. As a result, the PML seems to have large variation and deviation from the set point. Additionally, the PDFSP has large variation independent on the process state. Nevertheless, the controller with the parameter adaptation is able both to follow the set point and compensate for the majority of the variance of the PML. Similarly to the low disturbance level, the effect of disturbance on the actuator signal is nearly totally compensated.

Fig. 5 presents the adaptation of both the MV and MO weights. During the step response, the MO weight becomes larger due to the increase in the error terms (at t = 1500). Correspondingly, the MV weight decreases to allow faster tracking of the set point. Once the new set point is reached, the adaptation term based on the index value starts to dominate. Therefore the MO weight is decreased and input weights increased in order to attenuate the structural patterns in the process outputs. Finally, the weights converge nearly to the same values as before the step change in the process input.

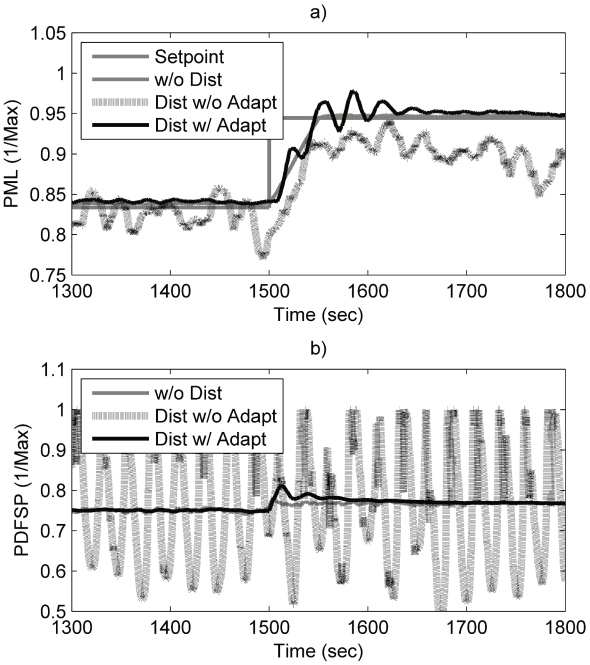


Fig. 4. Control performance in the presence of high communications disturbance in the actuator signal and some variation in the delay of the process output measurement. a) Measured output (PML) b) Manipulated variable (PDFSP).

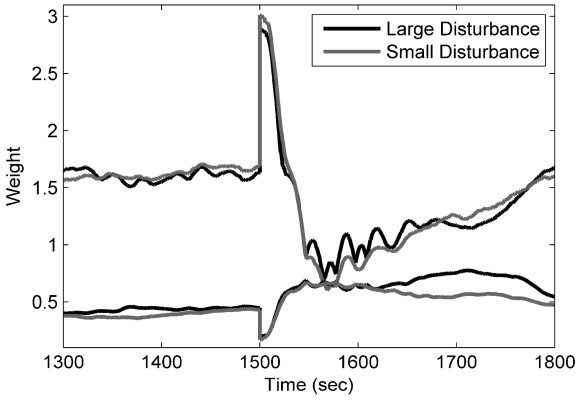


Fig. 5. Adaptation of the MO (upper two) and MV (lower two curves) weights

3.2 Comparison with Other Approaches

The other statistical measures that could be applied similarly for parameter adaptation include autocorrelation, correlation and standard deviation, especially in the linear case. The performance of the proposed index and these statistical measures in detecting the presence of disturbance is presented in Fig. 6. Based on the results, the proposed index and correlation seem to perform best in the detection because the signal levels of the "disturbance / no disturbance" cases are not overlapping (Fig. 6b). Moreover, their separation performance seems not to be degraded by a step in the PML set point at t = 2000 in this case.

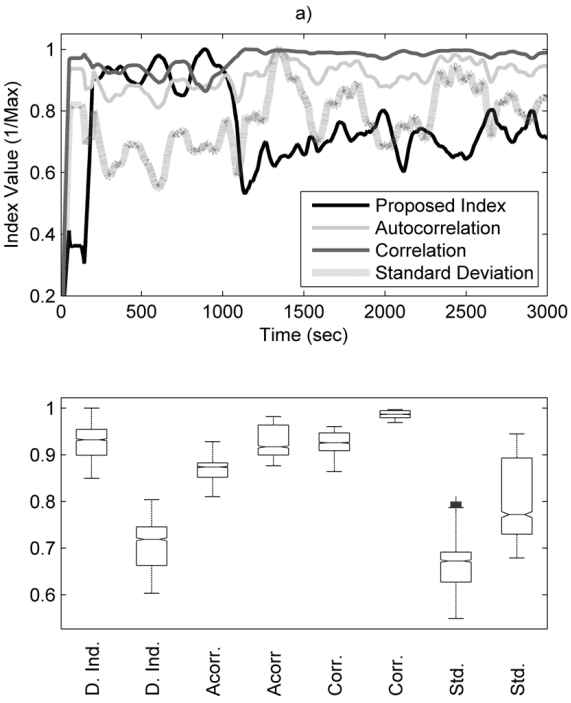


Fig. 6. a) Scaled values, each signal divided by its maximum (1/Max), of the different indices applied in disturbance detection. During the time span 0-1000 s there are no disturbances. Between the interval 1000-3000 s a major disturbance is present both in the manipulated variables and measurement communication channels. A step in the PML set point takes place at t = 2000 s. A good index can separate the two operating conditions (with and without disturbance). b) Boxplots of the different indices presented in the Fig. 6a without (left box in each category separated by the dashed line) and in the presence of the disturbance (right box). If the boxes are not overlapping, an index can separate the different operating conditions. The data to boxplots were taken from 200-1000 s and 1900-2700 s.

The use of the proposed distance index was compared with the use of correlation in the adaptation procedure. Correlation was modified to the adaptation index by reducing its value from one and saturating the resulting value to the interval [0.2 0.8]. The modifications were done in order to be able to utilise Eq. 3 and Eq. 4 as such. Moreover, the scaling factors of the adaptation algorithm were tuned to produce the same initial controller weight values in "no disturbance" -case as with the proposed index. Additionally, instead of adapting the controller weights, Kalman filter applying PML set point as a model was also applied in the measurement disturbance rejection. The comparision of these three approaches is shown in Fig 7.

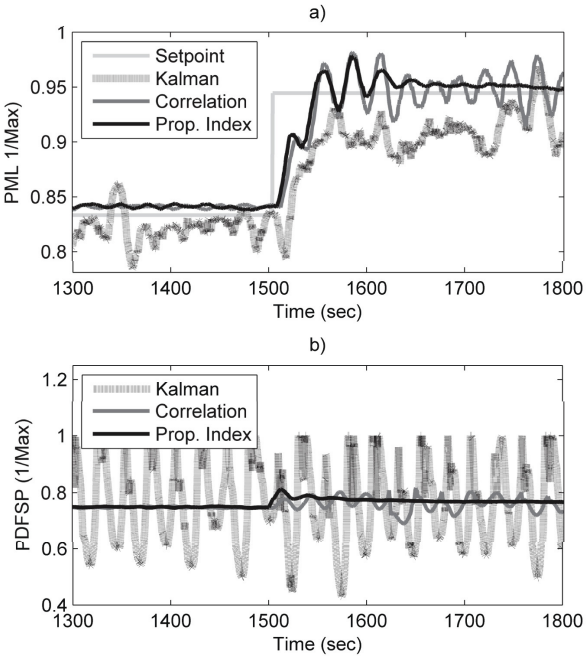


Fig. 7. Control performance in the presence of the high communication disturbance, using Kalman filter in disturbance rejection and adaptation based on the correlation and proposed index. a) Measured output (PML) b) Manipulated variable (PDFSP)

Compared to the case "disturbance without adaptation" (Fig. 4), the correlation-based adaptation seems to perform well on the lower PML set point (interval 1300-1500 s). However, contrary to the proposed adaptation algorithm, instability can be detected in both the measured and manipulated variables at the higher PML set point. A more detailed investigation revealed that also the controller weights varied correspondingly. The Kalman filter approach could not improve the control performance notably compared to "without adaptation" -case presented earlier (Fig. 4).

The observations about the superiority of the proposed adaptation procedure are also seen from Table 1 which shows the integral squared error (ISE) values of the control performances. The performance of the proposed adaptation scheme is almost

equal to the case with no disturbance and adaptation. The efficiency of the proposed algorithm becomes very obvious with high disturbances as the ISE value without adaptation increases significantly.

Table 1. The ISE values for comparing the control performances with different adaptation schemes

	Low disturbance	High disturbance
Without adaptation	44.0353	171.9272
Proposed adaptation	39.5877	40.6448
Correlation-based adaptation	-	46.5783
Kalman filter	-	139.5092

* The ISE value without disturbance and adaptation is 34.0466.

4 Conclusions

In industrial control, the effect of disturbances could be minimised by adjusting the controller based on the confidence on the data. This is done by the adaptive procedure described first time in this paper. The used metric based on the calculation of distance index in a sliding window is superior to other alternative indices and Kalman filter. The key advantages of the presented distance index include:

- Pre-assumptions about the type of the dependency or systems dynamics are not required. Therefore, the index value could be applicable also for streams with non-linearity.
- The index is non-parametric, model-free and purely probabilistic.
- The presented index seems to be especially applicable for controller parameter tuning with the developed adaptation algorithm.

The actual control is carried out by a standard MPC algorithm and adaptation takes place by adjusting input and output weights. The future research will concentrate on expanding the framework including the distance index and proposed adaptation algorithms to multivariable, multilayer and networked control schemes. Additionally, the distance index value is possibly applicable also for different types of data mining applications.

References

1. Antsaklis, P., Baillieul, J.: Special Issue on Technology of Networked Control Systems. Proceedings of the IEEE 9, 5–8 (2007)
2. Hespanha, J.P., Naghshtabrizi, P., Xu, Y.: A survey of recent results in networked control systems. Proceedings of the IEEE 95, 138–162 (2007)
3. Yliniemi, L., Leiviskä, K.: Process control across wired network. In: IASTED International Conference on Parallel and Distributed Computing and Networks, February 14-16. Acta Press, Innsbruck (2006)

4. Tipsuwan, Y., Chow, M.-Y.: Control methodologies in networked control systems. Control Engineering Practise, 1099–1111 (2003)
5. Paavola, M.: An efficient entropy estimation approach. Doctoral Thesis C394, Acta Universitatis Ouluensis (2011)
6. Caruntu, C.F., Lazar, C.: Networked predictive control for time-varying delay compensation with an application to automotive mechatronics system. Control Engineering and Applied Informatic 13, 19–25 (2011)
7. The Mathworks, Inc, http://www.mathworks.com
8. Jaccard, P.: Étude comparative de la distribution floraledansune portion des Alpeset des Jura. Bulletin de la SociétéVaudoise des Sciences Naturelles 37, 547–579 (1901)
9. Estévez, P.A., Tesmer, M., Perez, C.A., Zurada, J.M.: Normalized mutual information feature selection. IEEE Transactions on Neural Networks 20, 189–201 (2009)
10. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
11. Kaiser, A., Schreiber, T.: Information transfer in continuous processes. Physica D 166, 43–62 (2002)
12. Vu, V.Q., Yu, B., Kass, R.E.: Information in the nonstationary case. Neural Computation 21, 688–703 (2009)
13. Fraser, A.M., Swinney, H.L.: Independent coordinates for strange attractors from mutual information. Physical Review A 33, 1134–1140 (1986)

Learning Pattern Graphs for Multivariate Temporal Pattern Retrieval

Sebastian Peter[1], Frank Höppner[2], and Michael R. Berthold[1]

[1] Nycomed-Chair for Bioinformatics and Information Mining
Dept. of Computer Science, University of Konstanz
Box 712, D-78457 Konstanz, Germany
[2] Ostfalia University of Applied Sciences
Dept. of Computer Science, D-38302 Wolfenbüttel, Germany

Abstract. We propose a two-phased approach to learn pattern graphs, a powerful pattern language for complex, multivariate temporal data, which is capable of reflecting more aspects of temporal patterns than earlier proposals. The first phase aims at increasing the understandability of the graph by finding common substructures, thereby helping the second phase to specialize the graph learned so far to discriminate against undesired situations. The usefulness is shown on data from the automobile industry and the libras data set by taking the accuracy and the knowledge gain of the learned graphs into account.

1 Introduction

As the number of (mobile and/or wireless) sensor networks increases (e.g. health care, climate, earthquakes, traffic), more and more data is gathered periodically and the interest in analyzing temporal data rises. The recorded data usually includes various dimensions and the user is often interested in *typical* or *characteristic* situations. To grasp or encompass these situations, various notions of *multivariate temporal patterns* are employed in the literature. Example applications for multivariate temporal patterns include the discovery of dependencies in wireless sensor networks [1], the exploration of typical (business) workflows [3] or classification of electronic health records [2].

We present a new approach to derive classification rules automatically from multivariate, temporal data. Rather than enumerating all patterns (and forcing an expert to have a look at (too) many of them), we invite the expert to actively work on the patterns – by constructing them from scratch, by extending the patterns proposed by the algorithms presented in this paper or by alternating between both steps. To enable such a successful interaction, it is crucial that the pattern language itself is expressive enough to include the kind of constraints the expert wants to express. We use *pattern graphs* [8] as they provide the necessary flexibility to include not only the order of events, but also different kinds of parallelism, absence of events, as well as constraints on their duration.

The outline of the paper is as follows: We discuss notions of multivariate temporal patterns, including the pattern graph, in the next section. A two-phased

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 264–275, 2012.
© Springer-Verlag Berlin Heidelberg 2012

approach to construct rich pattern graphs for classification tasks, consisting of the identification of common substructures and its specialization towards class predictability, is presented in Sect. 3. Results from a real world application are shown in Sect. 4. Finally Sect. 5 concludes the paper.

2 Notions of Multivariate Temporal Patterns

Preliminaries. Due to the fact that we consider a classification task, a case consists of a multivariate data sequence S plus a class label. Rather than building the patterns upon raw data (e.g. the time series itself), we formulate them by means of various conditions (temporal abstractions), such as A≡'speed is above 50 km/h' or B≡'clutch pedaled' etc. We use these abstractions to pose *constraints* over certain periods of time. Besides these constraints on the *values* of the sequences, we also consider *temporal* constraints on their duration. For instance, we may require that there is some period of time in which 'A is present and B is absent for 5-10 time units' (which will be abbreviated by '$A, \neg B$ $[5, 10]$').

Related Work. Many approaches, such as [2], describe multivariate temporal patterns by specifying the relationships between all observed intervals like '*A before B*', '*A overlaps C*' and '*C overlaps B*' in the spirit of [5]. This notation is quite strict and somewhat ambiguous [7], because the qualitative relationship does not carry quantitative information about the degree of overlap or size of a gap. Other approaches contain such information [3], but only consider events without *duration* and thus offer no means to express concurrency as in '*A and B must co-occur for 5-10 time units*'. In practice, we frequently want to forbid some events (e.g. '*no connection to host while running batch*'). Sometimes negative constraints are used to *filter* a large set of enumerated patterns [1], but the pattern language itself seldomly offers the means to express the *absence* of events. For this paper, we settled on pattern graphs, because they offer all these features.

Notion of a Pattern Graph. A *pattern graph* $(V, E, C_{\text{val}}, C_{\text{temp}})$ is an acyclic, directed graph (V, E) with one source ($\top \in V$) and one sink ($\bot \in V$). Associated with each node $v \in V$, we have *value constraints* $C_{\text{val}}(v)$ on the necessary observations while in node v and *temporal constraints* $C_{\text{temp}}(v)$ on the duration of node v. A sequence matches (fits) the pattern graph if it is possible to assign subsequences from S to all nodes of the graph, such that the subsequences satisfy all constraints of the assigned nodes simultaneously. The assignment has to be complete in the sense, that (1) the empty prefix of S is assigned to the source, the empty suffix of S to the sink, (2) a non-empty sub-sequence is assigned to all other nodes, and (3) the subsequences to connected nodes are contiguous.

Interpretation. In Fig. 1 we see an example pattern graph with two parallel paths which is read as follows:

1. The temporal constraint of a node is represented above the node. A star represents an unlimited duration.
2. The value constraint(s) of a node is (are) shown inside the node.
3. A node without any value constraints is labeled '?' (*don't care*).

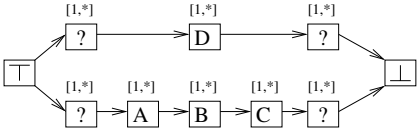


Fig. 1. Example pattern graph with two parallel paths from $\top$ to $\bot$

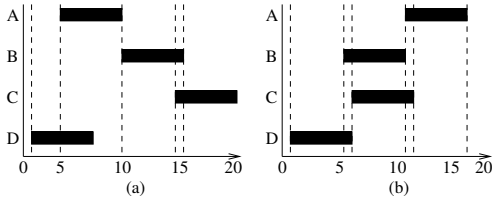


Fig. 2. Two example sequences with four binary properties A-D

To match a given sequence to a pattern graph, it has to be subdivided into several parts (on the time axis), such that each part can be assigned to a node of the graph. The nodes therefore represent a part of a sequence and pose constraints on the values and the duration of the subsequence: The temporal constraints restrict the length and the value constraints restrict the behavior of the respective subsequence. The edges between the nodes enforce the order of the parts. If a node has an outgoing edge it means that there has to be another part directly after the associated subsequence that fulfils the constraints of the successor node. If a node has two or more outgoing edges, all parts belonging to the following nodes have to begin at the same time. On the other hand, if a node has two or more incoming edges all parts belonging to the preceding nodes have to end at the same time and the part of the node has to begin immediately afterwards. Thus, to express that two conditions A and B occur simultaneously, we include both, A and B in *one* node (as value constraints). In contrast, to express that A and B are concurrent but independent from each other, we introduce parallel paths for A and B. Any condition A may continue to hold before or after the node unless a condition $\neg A$ forbids this explicitly.

Mapping sequences. Fig. 2 shows two sequences where the vertical axis shows a number of value constraints A-D that hold over certain periods of time (black bars, time on horizontal axis). We now discuss whether these sequences can be mapped validly to the pattern graph in Fig. 1. The graph shown in Fig. 1 can be decomposed into two different paths: For the lower path the sequence has to be divided in five contiguous parts, so that the first part satisfies the '*don't care*' constraint, during the second part the property A has to hold, the property B in the third, etc. The last part is again a '*don't care*'-part. All of these five parts require a duration of at least one time unit (but have no upper bound on the duration). Parallel to the lower path, the upper path requires '*don't care*', 'D' and '*don't care*' again with durations ≥ 1 time unit.

The sequence shown in 2(a) can be mapped to the graph, because we can clearly see that A is before B and B before C. And D is present during this subsequence as well. Due to the fact that B and C are overlapping, it is possible to assign different subsequences to the B and C node (the overlapping part may be assigned to any of the respective nodes or may even be split up into two parts). This means that the pattern graph has more than one valid mapping.

On the other hand we cannot find a valid mapping for the sequence shown in Fig. 2(b), because there, A does not occur before B. If A were true within $[6, 9]$ (rather than $[10, 15]$), we would have another valid mapping.

For a more formal definition of a pattern graph and a valid mapping, as well as an efficient algorithm that decides if a sequence matches a given pattern graph or not, we refer the interested reader to [8]. This algorithm may either simply state whether a match is possible, or it provides us with detailed information about possible matches in the following sense: For each edge $e = (u, v) \in E$ in the graph, we obtain a set of valid positions $p(e)$ in case of a valid mapping, that is, a set of positions t that satisfy all value constraints of node u for $t' < t$ and all value constraints of node v for $t' \geq t$. For the graph in Fig. 1 and the sequence in Fig. 2(a) for the edge e from node A to B we have only one valid location $p(e) = \{10\}$, but for the edge e' from B to C we have $p(e') = \{[14, 15]\}$.

3 Learning Pattern Graphs

Next we propose a two-phased approach to learn pattern graphs from labeled data. In the first phase we build a pattern graph that is mappable to *all* instances of the target class. This is done for the following two purposes: Firstly, the resulting pattern graph describes the structure shared by all instances of the class – even if they were not necessary for the purpose of classification – and thus supports the interpretability of the class. Secondly, the graph represents a good initialization for the second phase of the algorithm, which is a pattern graph refinement via beam search. The beam search is tailored towards a quick improvement in terms of discrimination capabilities, but with real-world problems it is impossible to discriminate classes with a pattern graph if it consists of a few nodes only. If we were starting the beam search from scratch (empty graph), it would waste considerable time to build up a pattern graph that is complex enough to eventually include the important aspects for the classification task. In [6] we have experimentally shown that a two-phased approach for learning (a restricted set of) temporal patterns may not only increase the understandability but additionally may increase the accuracy of the patterns as well. Here, we have extended this approach to the case of full-fledged pattern graphs (whereas the earlier work used only 'limited patten graphs', which consisted of a single path from $\top$ to $\bot$ only).

3.1 First Phase: Identifying the Shared Structure (Within a Class)

As already mentioned the primary goal of this phase is to find key aspects of the target class. As we expect to obtain quite complex graphs from this step already, we do not employ frequent sequence mining algorithms (e.g. [9]), as they would waste considerable time on the enumeration of a huge number of frequent subgraphs. Furthermore, such methods are usually not suited for including *absent* items (or constraints). The problem of finding a pattern common to all instances (of one class) is closely related to the alignment of multiple sequences, which is

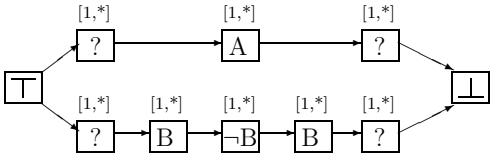


Fig. 3. Resulting pattern graph from step 1 where A occurs once and B two times

known to be NP-complete [10]. As we are aware that those parts of the graph, which are important for the classification task, will be inserted during the refinement phase anyway, there is no need to find some kind of *best graph* in the first phase, therefore we settle for a heuristic approach. The following proposal exploits the possibilities of the pattern graph to incorporate *temporal-*, *present-*, *absent-* and '*don't care*' -constraints into the pattern graph. It consists of three steps: In the first step the relevant nodes of the graph are computed, then we add relations between the nodes and finally remove redundant parts.

Univariate Paths. As a preprocessing step, we scan through all instances of the target class once and determine the (contiguous) intervals in which a constraint holds. For every constraint C the minimum number n_C of intervals per instance is determined. From this information we create the first pattern graph: For every constraint C we create a path from $\top$ to $\bot$ always starting and ending with a '*don't care*'-node, consisting of an alternating sequence of n_C *present-* and *absent-* nodes. For example, if constraint A holds at least once and constraint B twice, the resulting graph is shown in Fig. 3. The resulting pattern graph has at least one valid mapping on all sequences of the target class, as we only state the minimal number of occurrences and require no specific relation between different constraints, because they are aligned in different paths.

Linking Paths. The second step inserts connections between nodes from different paths. As already mentioned, the matching algorithm [8] returns for an edge $e \in E$ all valid edge locations $p(e)$ in a given sequence. For two nodes u and v, let $A = \bigcap_{(u,w)\in E} p(u,w)$ contain all valid 'end positions of node u' and let $B = \bigcap_{(w,v)\in E} p(w,v)$ contain all valid 'start positions of node v'. If there are $a \in A, b \in B$ such that $0 < b - a \leq t$, the gap between node u and v is at most t time units wide. If it is possible to satisfy this condition for every individual sequence, we may safely introduce a new '*don't care*'-node between nodes u and v with a temporal constraint $[1, t]$ without risking the match of any of the sequences to the extended pattern (because its feasibility has already been checked). The function '`check(u,v,t,S)`' indicates by its return value whether this condition is satisfied for all sequences S. To support the understandability of the graph we do not allow the connection of '*don't care*' nodes. Furthermore, an edge connecting u to v is not added if a (possibly longer) path already exists from u to v. A sketch of this procedure is shown in Alg. 1.

Removing Redundancy. The last step removes those edges and nodes from the graph that are no longer needed because they do not provide additional

Algorithm 1. Linking paths

Require: start pattern graph $G = (V, E, C_{\text{val}}, C_{\text{temp}})$, set S of target sequences, minlength, maxlength, stepsize
Ensure: return extended pattern graph

```
t ← minlength
repeat
   for all (v1, v2) ∈ V × V do
      if check(v1, v2, t, S) then
         extended G by a '?'-node between v1 and v2 with duration [1, t]
         re-run pattern matcher to update sets p(e), e ∈ E
      end if
   end for
   t ← t+stepsize
until t >maxlength
return G
```

information (or may be derived from transitivity). In particular, the algorithm checks for each '*don't care*'-node v, with exactly one incoming and one outgoing edge, if a path exists from node v_p directly preceding it to the node v_f directly following it. If this is the case the '*don't care*'-node and the connecting edges are removed. This ensures us, that no synchronization between nodes is removed and all constraints are preserved: synchronization requires at least two incoming or outgoing edges and no value constraints are removed (the parallel path subsumes the '*don't care*' constraint). As an example, consider the case that in step two the pattern graph in Fig. 3 has been extended by an edge connecting A and $\neg B$. This would render the '*don't care*' node following A useless because we can follow the path $A \to \neg B \to B \to ? \to \perp$ to reach $\perp$ as well.

3.2 Second Phase: Discrimination (Between Classes)

Next, we propose a method to explore the space of pattern graphs to discriminate differently labeled sequences. The search algorithm implements a general-to-specific search: it begins with the pattern graph from phase one, which matches all instances (of the class), and tries to specialize it further to improve some chosen measure of interestingness (e.g. the J-measure). While a propositional rule can only be specialized by an additional condition (like *outlook=sunny*), there are various ways to specialize a pattern graph: we can examine it in a finer resolution (by splitting a node into two or three nodes), we can change or add a value constraint (for some node), or introduce or change an existing temporal constraint. Furthermore we can add new nodes or connect two existing nodes with an additional edge to express a temporal dependency. We have settled on five different specialization operators to address each of these aspects. The outline of the beam search algorithm is given in Algorithm 2.

The general idea for all refinement operators is to search for specializations that improve the measure of interestingness, which basically requires that the

Algorithm 2. Outline of the beam search

Require: start pattern graph G_S, set S of labeled sequences
Ensure: set of k best pattern graphs (acc. to some measure of interestingness)

1: initialize the set T with the single start pattern G_S
2: **repeat**
3: $B \leftarrow T$
4: **for all** $G \in B$ **do**
5: apply all refinement operators to G and insert resulting graphs into T (but keep only the k best graphs in T)
6: **end for**
7: **until** k^{th} best graph in T is not better than the k^{th} best graph in B
8: **return** B

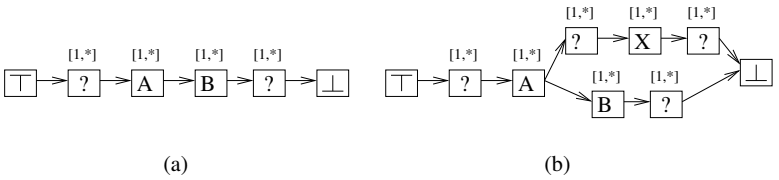


Fig. 4. Example of how the pattern graph (a) is refined by the partial order refinement operator to the pattern graph (b)

specialized pattern still matches the positive instances but less negatives. Due to lack of space, we will describe only two operators briefly.

Link Refinement. This refinement operator is almost identical to step two of phase one, except that the criterion for path-inclusion is now an increase in some chosen measure of interestingness rather than matching all instances of the considered class. The connection that discriminates best will be included.

Path Refinement. This refinement operator determines for all nodes n of the graph, whether some condition (temporal abstraction) occurs frequently before or after the subsequence assigned to node n. As an example, consider the node labeled A in the pattern graph of Fig. 4(a) as n. Suppose that in some instances of the same class we observe a presence of condition X after n. If the inclusion of 'X after n' increases the measure of interestingness most (among all other possible combinations), we add a short sequence of nodes: $? \rightarrow X \rightarrow ?$ between the node n and $\bot$ (or $\top$ in case the presence of X is observed *before* n). In our example, we may obtain the pattern graph shown in Fig. 4(b).

Note that this refinement operator does not add the constraint between the nodes labeled A and B, because it did not determine the relationship between X and the node labeled B. We thus insert a path to $\top$ or $\bot$ and leave the determination of a possible temporal relationship to B open for further refinements.

Complexity. The number of possible pattern graphs grows quickly with the number of nodes and possible constraints in the data set. But we restrict the

number of actually explored patterns with the size of the beam. Apparently we cannot guarantee that the best graph will eventually be found, because beam search is a heuristic search technique. For each main iteration, the patterns in the beam are extended by the five mentioned operators. All of the operators directly work on the result of the matcher, so each instance has to be matched against a pattern graph only once. The complexity of the refinement operators differ: for example, the worst case complexity of finding the best partial order refinement is in $\mathcal{O}(v * n * (m + c * c_n))$ and that of link refinement is in $\mathcal{O}(n * v^2 * m)$, resp. Here v denotes the number of nodes in the graph, n the number of sequences, c the number of different conditions, m the number of matches and c_n the number of occurrences of the conditions in the sequences.

4 Experimental Evaluation

We evaluated the proposed algorithm on real world data from a German car manufacturer. Several cars were equipped with recording devices that captured various measurements, such as current speed, gear, pedal state and angles, etc. The goal is to identify driving cycles with a specific duration in the data, which appears pretty simple at first glance. In a test-bed situation, a driving cycle may be defined as a sequence of acceleration, constant speed and deceleration. However, if we define 'cycle' by such a pattern, it matches far more situations than the experts actually had in mind. Fig. 5 shows one test drive, where the first plot shows the current speed of the car over time and the lower plot the intervals during which various conditions hold. These conditions were derived

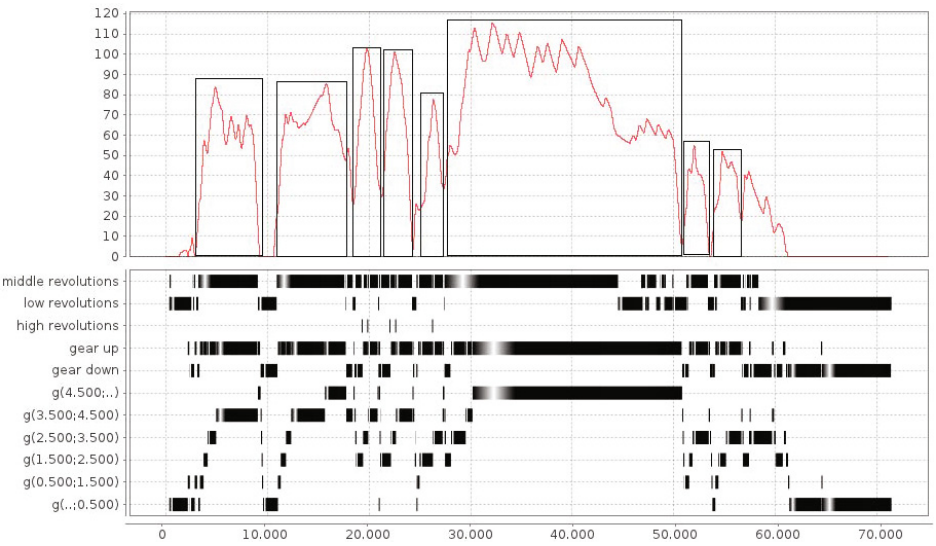


Fig. 5. Possible driving cycles marked in the sequence, which could be used as instances for the target class

from the numerical time series data using a priori defined thresholds. We used the following labels:

- gear up/down: indicates whether the last gear shift was up- or down-shift
- g: represents the current gear of the car: no gear (g(..-0.500)), gear one (g(0.500-1.500)), ..., gear 4 or higher (g(4.500-...))
- revolutions: represents the engine revolutions (low, middle, high)
- coupling: clutch is pedaled (coupling(0.500-..)) or not (coupling(..-0.500))

Learning the Pattern Graph. In [8] we created a pattern graph from scratch by iteratively enhancing the graph with the help of expert knowledge to retrieve the desired driving cycles. In the following we want to show how the above mentioned approach could be used to help the expert specifying the pattern graph or how a pattern graph could be learned if no expert is available.

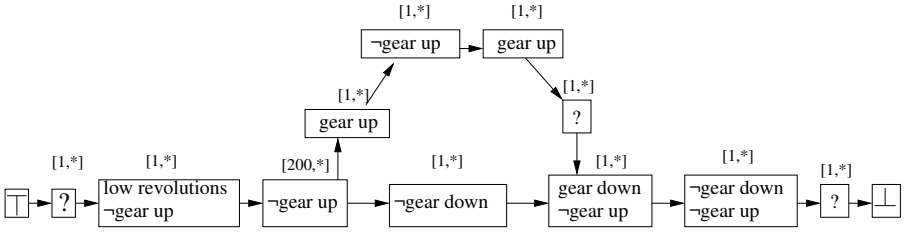


Fig. 6. Pattern graph to query driving cycles

To derive a pattern for the target class, as with any other classifier we need examples that contain the target class as well as examples that do not (or contain other classes). For illustration purposes, some 'driving cycle' examples are marked by rectangles in Fig. 5. In many applications the sequence labels will be readily available, but in our case we may ask an expert to label some examples manually. However, as we have already specified *a pattern* (rather than marked individual sequences) for 'driving cycles' by means of expert knowledge in [8], which does a pretty good job at identifying cycles, we have decided to use this pattern (shown in Fig. 6) to label example sequences: We have extracted various (random) subsequences and label them 'driving cycle' if the manually constructed pattern matches. The appeal of this procedure is that it allows us to compare the learned graph directly to the manually constructed graph.

The pattern graph without the dashed nodes shown in Fig. 7 was found during the first phase of the approach. This graph is already very similar to the graph used for extraction (cf. Fig. 6), the sequence of two up-shifts is prominent in both graphs. The location of the *low revolutions* constraint is different, it appears to occur somewhat later in the extracted graph. However, both parallel paths in the extracted graph synchronize at the *coupling*-node. While the upper part (*gear-up*) is closely connected to this node (time constraint [1,50]), this is not

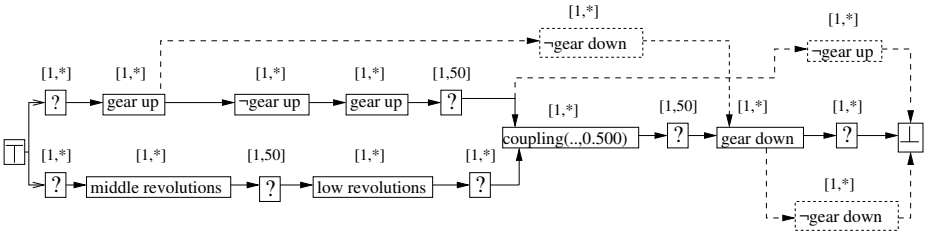


Fig. 7. Pattern graph learned for the driving cycles. The graph without the dashed nodes shows the pattern after phase one and the 3 dashed nodes are added during phase 2.

necessarily the case for the lower part. Due to the unlimited temporal constraint of the '*don't care*'-node, the *low revolutions* are allowed to appear before the first *gear up* (as in the original graph). The fact that no temporal dependency is introduced in the extracted path may be explained by the simple fact, that it did not help to discriminate the classes. Furthermore the graph requires *middle revolutions* before *low revolutions*, but of course this is always the case when slowing down before the next cycle starts (except for the first cycle starting from a parking position). An interesting aspect is that between the last *gear up* and the *gear down* node *coupling(..,0.500)* has to occur, which is a stronger constraint than ¬*gear down* because a gear shift requires coupling. If we apply the learned pattern to all instances we retrieve the confusion matrix $\left(\begin{smallmatrix} 300 & 0 \\ 174 & 249. \end{smallmatrix}\right)$, where we can see that all 'cycles' are matched correctly however we also obtained 174 false positives. This was to be expected as we were not trying to separate the 'cycles' from other sequences so far.

This is the task of the subsequent phase, which tries to reduce the number of false positives by adding constraints to discriminate the target class from all other classes. The resulting pattern is also shown in Fig. 7 if we include the dashed nodes. The refinement operators added three constraints to the graph: The first constraint is the ¬*gear down*-node between the first *gear up*- and *gear down*-node, thus stating that no down-shift is allowed during the 'cycle'. The second refinement is the ¬*gear up* node connected from the *don't care*-node after the last gear up node to $\bot$, which ensures that after the last up-shift no further up-shift can occur. The last additional constraint states that after the *gear down*-node no further change in a lower gear is permitted (additional ¬*gear down* node connected between *gear down* and $\bot$).

By comparing the learned pattern (after phase 2) in Fig. 7 to the pattern used for the extraction in Fig. 6 we can see that the patterns describe a nearly identical 'driving cycle'. The beam search step only inserted constraints that helped to separate the randomly selected sequences from the 'cycles'. All these constraints are also found in the manually defined graph that was used for labeling the sequences. As the graphs are nearly the same it is not surprising that the learned pattern performs perfectly as indicated by the confusion matrix: $\left(\begin{smallmatrix} 300 & 0 \\ 0 & 439. \end{smallmatrix}\right)$.

Libras Movement. We also applied our algorithm to the libras movement data set from the UCI repository [4]. It contains 15 different signs described by their characteristic hand movement over 45 frames, where the current x- and y-positions of the hand were recorded. We extracted features to address the speed of the hand movement in the x- and y-direction only. For each sign we learned a pattern graph on 66% of the data and then matched all pattern graphs to the unseen data. We predict a class only if just one graph matches (and is 'undecided' otherwise). We arrive at 98 correct, 2 false and 23 unclassified instances, resulting in 79.675% accuracy and 20.325% error-rate. For the 23 unclassified instances (where no pattern graph matches), we can switch back to manual mode and inspect and change the pattern manually to further improve the classification rate. The dataset contains some cases that deviate greatly from the original hand movement, such that even a human is not able to classify them. By removing these outliers the approach improves to 88.235% accuracy and 11.765% error rate.

5 Conclusion

We consider pattern graphs as useful for capturing multivariate patterns in temporal data, because they are capable of expressing most of the constraints a human expert may want to use when describing a specific situation (e.g. absence of events, durations, different kinds of parallelism). These graphs are thus well-suited for manual construction [8], but in this paper we have demonstrated that they can also be learned automatically from data. We have presented a two-phased approach to construct pattern graphs for classification tasks. The first phase identifies the common structure in all sequences of the same class and the second phase refines this structure further to discriminate between the different classes. Early results are encouraging, the approach was able to successfully re-discover hand-coded pattern graphs from a set of labeled examples.

Acknowledgements. We would like to thank Dr. Werther from Volkswagen AG for kindly providing the data.

References

1. Basile, T.M.A., Di Mauro, N., Ferilli, S., Esposito, F.: Relational Temporal Data Mining for Wireless Sensor Networks. In: Serra, R., Cucchiara, R. (eds.) AI*IA 2009. LNCS(LNAI), vol. 5883, pp. 416–425. Springer, Heidelberg (2009)
2. Batal, I., Valizadegan, H., Cooper, G.F., Hauskrecht, M.: A pattern mining approach for classifying multivariate temporal data. In: Proc. IEEE Int. Conf. Bioinformatics BioMed, pp. 358–365 (2011)
3. Berlingerio, M., Pinelli, F., Nanni, M., Giannotti, F.: Temporal mining for interactive workflow data analysis. In: Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD 2009, pp. 109–118 (2009)

4. Frank, A., Asuncion, A.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2010)
5. Höppner, F.: Discovery of temporal patterns – learning rules about the qualitative behaviour of time series. In: Proc. of the 5th Europ. Conf. on Principles of Data Mining and Knowl. Discovery, pp. 192–203. Springer (2001)
6. Höppner, F., Peter, S., Berthold, M.R.: Enriching Multivariate Temporal Patterns with Context Information to Support Classification. In: Moewes, C., Nürnberger, A. (eds.) Computational Intelligence in Intelligent Data Analysis. SCI, vol. 445, pp. 195–206. Springer, Heidelberg (2013)
7. Mörchen, F.: Unsupervised pattern mining from symbolic temporal data. ACM SIGKDD Explorations Newsletter 9(1), 41–55 (2007)
8. Peter, S., Höppner, F., Berthold, M.R.: Pattern graphs: A knowledge-based tool for multivariate temporal pattern retrieval. In: Proc. IEEE Conf. Intelligent Systems. IEEE (2012)
9. Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: Int. Conf on Data Engineering, pp. 79–90 (2004)
10. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology 1(4), 337–348 (1994)

GET_MOVE: An Efficient and Unifying Spatio-temporal Pattern Mining Algorithm for Moving Objects

Phan Nhat Hai[1,2], Pascal Poncelet[1,2], and Maguelonne Teisseire[1,2]

[1] IRSTEA Montpellier, UMR TETIS - 34093 Montpellier, France
{nhat-hai.phan,maguelonne.teisseire}@teledetection.fr
[2] LIRMM CNRS Montpellier - 34090 Montpellier, France
pascal.poncelet@lirmm.fr

Abstract. Recent improvements in positioning technology have led to a massive moving object data. A crucial task is to find the moving objects that travel together. Usually, they are called spatio-temporal patterns. Due to the emergence of many different kinds of spatio-temporal patterns in recent years, different approaches have been proposed to extract them. However, each approach only focuses on mining a specific kind of pattern. In addition to the fact that it is a painstaking task due to the large number of algorithms used to mine and manage patterns, it is also time consuming. Additionally, we have to execute these algorithms again whenever new data are added to the existing database. To address these issues, we first redefine spatio-temporal patterns in the itemset context. Secondly, we propose a unifying approach, named *GeT_Move*, using a frequent closed itemset-based spatio-temporal pattern-mining algorithm to mine and manage different spatio-temporal patterns. GeT_Move is implemented in two versions which are GeT_Move and Incremental GeT_Move. Experiments are performed on real and synthetic datasets and the results show that our approaches are very effective and outperform existing algorithms in terms of efficiency.

Keywords: Spatio-temporal pattern, frequent closed itemset, trajectories.

1 Introduction

Nowadays, many electronic devices are used for real world applications. Telemetry attached on wildlife, GPS installed in cars, sensor networks, and mobile phones have enabled the tracking of almost any kind of data and has led to an increasingly large amount of data that contain moving objects. Therefore, analysis on such data to find interesting patterns is attracting increasing attention for applications such as movement pattern analysis, animal behavior study, route planning and vehicle control.

Recently, many spatio-temporal patterns have been proposed [1, 3, 4, 6, 2, 10, 9]. In this paper, we are interested in the querying of patterns which capture *'group'* or *'common'* behaviour among moving entities. This is particularly true to identify groups of moving objects for which a strong relationship and interaction exist within a defined spatial region during a given time duration. Some examples of these patterns are flocks [1], moving clusters [4, 9], convoy queries [3], closed swarms [6], group patterns [2], periodic patterns [10], etc...

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 276–288, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Table 1. An example of a Spatio-Temporal Database

O_{DB}	T_{DB}	x	y
o_1	t_1	2.3	1.2
o_2	t_1	2.1	1
o_1	t_2	10.3	28.1

Table 2. Cluster Matrix

T_{DB}		t_1			t_2			t_3	
Clusters C_{DB}		c_{11}	c_{21}	c_{31}	c_{12}	c_{22}	c_{32}	c_{13}	c_{23}
O_{DB}	o_1	1			1			1	
	o_2	1			1			1	
	o_3	1				1		1	
	o_4			1	1				1
	o_5		1				1	1	

Table 3. Closed Itemset Matrix

Block B		b_1	b_2	
FCIs CI		ci_{11}	ci_{12}	ci_{22}
O_{DB}	o_1	1		1
	o_2	1		1
	o_3	1	1	
	o_4	1	1	

To extract these kinds of patterns, different algorithms have been proposed. Naturally, the computation is costly and time consuming because we need to execute different algorithms consecutively. However, if we had an algorithm which could extract different kinds of patterns, the computation costs will be significantly decreased and the process would be much less time consuming. Therefore, we need to develop an efficient unifying algorithm. Additionally, in real world applications (e.g. cars), object locations are continuously reported by using Global Positioning System (GPS). Thus, new data is always available. If we do not have an incremental algorithm, we need to execute again and again algorithms on the whole database including existing data and new data to extract patterns. This is of course, cost-prohibitive and time consuming. An incremental algorithm can indeed improve the process by combining the results extracted from the existing data and the new data to obtain the final results.

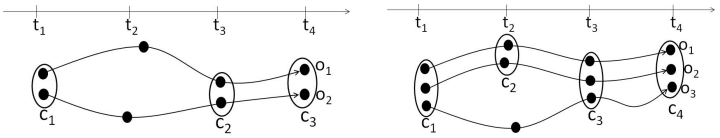
With these above issues in mind, we propose *GeT_Move*: a unifying incremental spatio-temporal pattern-mining approach. The main idea of the algorithm and contributions of this paper are summarized below. **(1)** We redefine the spatio-temporal pattern mining in the itemset context which enables us to effectively extract different kinds of spatio-temporal patterns. **(2)** We present approaches, called *GeT_Move* and *Incremental GeT_Move*, which efficiently extract Frequent Closed Itemsets (FCI) from which spatio-temporal patterns are retrieved. **(3)** We present comprehensive experimental results over both real and synthetic databases. The results demonstrate that our techniques enable us to effectively extract different kinds of patterns. Furthermore, our approaches are more efficient compared to other algorithms in most of cases.

The remaining sections are organized as follows. Section 2 discusses preliminary definitions of the spatio-temporal patterns and the related work. The properties of these patterns are provided in an itemset context in Section 3. We introduce the GeT_Move and Incremental GeT_Move algorithms in Section 4. Experiments testing effectiveness and efficiency are shown in Section 5. Finally, we draw our conclusions in Section 6.

2 Spatio-temporal Patterns

2.1 Preliminary Definitions

Basically, spatio-temporal patterns are designed to group similar trajectories or objects which tend to move together during a time interval. Recently, many patterns have been defined such as *flocks [1], convoys [3], swarms, closed swarms [6], moving clusters [4, 9], group pattern [2]* and even *periodic patterns [10].*



(a) Swarm (b) Convoy

Fig. 1. An example of swarm and convoy where c_1, c_2, c_3, c_4 are clusters

Let us assume that we have a group of moving objects $O_{DB} = \{o_1, o_2, \ldots, o_z\}$, a set of timestamps $T_{DB} = \{t_1, t_2, \ldots, t_n\}$ and at each timestamp $t_i \in T_{DB}$, spatial information[1] x, y for each object. For example, Table 1 illustrates an example of a spatio-temporal database. Usually, in spatio-temporal mining, we are interested in extracting a group of objects staying together during a period of time. Therefore, from now, $O = \{o_{i_1}, o_{i_2}, \ldots, o_{i_p}\} (O \subseteq O_{DB})$ stands for a group of objects, $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$ $(T \subseteq T_{DB})$ is the set of timestamps within which objects stay together. Let ε be a user-defined threshold standing for a minimum number of objects and min_t a minimum number of timestamps. Thus $|O|$ (resp. $|T|$) must be greater than or equal to ε (resp. min_t). In the following, we formally define all the different kinds of patterns.

Informally, a *swarm* is a group of moving objects O containing at least ε individuals which are closed each other for at least min_t timestamps. To avoid this redundancy, Zhenhui Li et al. [6] propose the notion of *closed swarm* for grouping together both objects and time. A swarm (O, T) is *object-closed* if when fixing T, O cannot be enlarged. Similarly, a swarm (O, T) is *time-closed* if when fixing O, T cannot be enlarged. Finally, a swarm (O, T) is a closed swarm if it is both object-closed and time-closed.

Definition 1. *Swarm and Closed Swarm [6]. A pair (O, T) is a swarm if:*

$$\begin{cases} (1) : \forall t_{a_i} \in T, \exists c \text{ s.t. } O \subseteq c, \ c \text{ is a cluster.} \\ (2) : |O| \geq \varepsilon \text{ and } |T| \geq min_t. \end{cases} \tag{1}$$

A swarm (O, T) is a closed swarm if:

$$\begin{cases} (1) : \nexists O' \text{ s.t. } (O', T) \text{ is a swarm and } O \subset O'. \\ (2) : \nexists T' \text{ s.t. } (O, T') \text{ is a swarm and } T \subset T'. \end{cases} \tag{2}$$

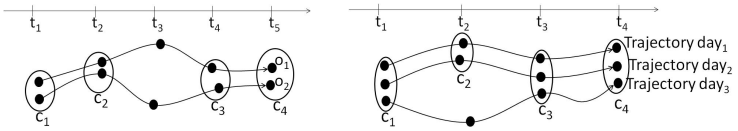
For example, as shown in Figure 1a, if we set $\varepsilon = 2$ and $min_t = 2$, we can find the following swarms $(\{o_1, o_2\}, \{t_1, t_3\})$, $(\{o_1, o_2\}, \{t_1, t_4\})$, $(\{o_1, o_2\}, \{t_3, t_4\})$, $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$. We can note that these swarms are in fact redundant since they can be grouped together in the following closed swarm $(\{o_1, o_2\}, \{t_1, t_3, t_4\})$.

A *convoy* is also a group of objects such that these objects are closed each other during at least min_t *consecutive* time points.

Definition 2. *Convoy [3]. A pair (O, T), is a convoy if:*

$$\begin{cases} (1) : (O, T) \text{ is a swarm.} \\ (2) : \forall i, 1 \leq i < |T|, \ t_{a_i}, t_{a_{i+1}} \text{ are consecutive.} \end{cases} \tag{3}$$

[1] Spatial information can be, for instance, GPS location.



A group pattern example. A periodic pattern example.

Fig. 2. Group pattern and periodic pattern example

For instance, on Figure 1b, with $\varepsilon = 2, min_t = 2$ we have two convoys ($\{o_1, o_2\}$, $\{t_1, t_2, t_3, t_4\}$) and ($\{o_1, o_2, o_3\}, \{t_3, t_4\}$).

Until now, we have considered that we have a group of objects that move close to each other for a long time interval. As shown in [11], moving clusters and different kinds of flocks virtually share essentially the same definition. Basically, the main difference is based on the clustering techniques used (i.e. rigid definition of the radius, DBScan [5]). Moving clusters can be seen as special cases of convoys with the additional condition that they need to share some objects between two consecutive timestamps [11]. Therefore, in the following, for brevity and clarity sake we will mainly focus on convoy and density-based clustering algorithms.

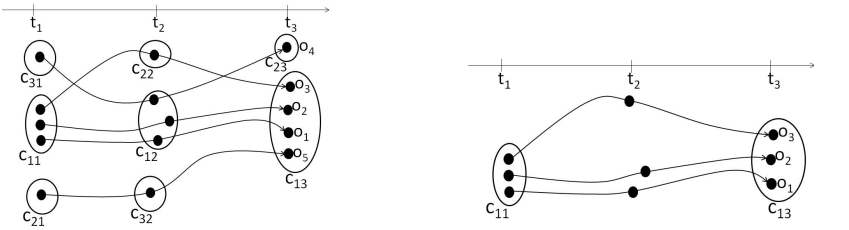
Hwang et al. [2] propose *group pattern* which essentially is a combination of both convoys and closed swarms. By considering a convoy as a timepoint, a group pattern can be seen as a closed swarm of disjointed convoys. Thus, it can be defined as follows.

Definition 3. *Group Pattern [2]. Given a set of objects O, a minimum weight threshold* min_{wei}*, a set of disjointed convoys* $T_{\mathcal{S}} = \{s_1, s_2, \ldots, s_n\}$*, a minimum number of convoys* min_c*.* $(O, T_{\mathcal{S}})$ *is a group pattern if:*

$$\begin{cases} (1) : (O, T_{\mathcal{S}}) \text{ is a closed swarm with } \varepsilon, min_c \text{. (e.g. } |T_{\mathcal{S}}| \geq min_c) \\ (2) : \frac{\sum_{i=1}^{|T_{\mathcal{S}}|} |s_i|}{|T_{DB}|} \geq min_{wei}. \end{cases} \tag{4}$$

For instance, see Figure 2a, with $min_t = 2$ and $\varepsilon = 2$ we have a set of convoys $T_{\mathcal{S}} = \{(\{o_1, o_2\}, \{t_1, t_2\}), (\{o_1, o_2\}, \{t_4, t_5\})\}$. Additionally, with $min_c = 1$ we have $(\{o_1, o_2\}, T_{\mathcal{S}})$ a closed swarm of convoys because $|T_{\mathcal{S}}| = 2 \geq min_c$, $|O| \geq \varepsilon$ and $(O, T_{\mathcal{S}})$ cannot be enlarged. Furthermore, with $min_{wei} = 0.5$, $(O, T_{\mathcal{S}})$ is a group pattern since $\frac{|[t_1,t_2]|+|[t_4,t_5]|}{|T_{DB}|} = \frac{4}{5} \geq min_{wei}$.

In [10], N. Mamoulis et al. propose the notion of *periodic patterns* in which an object follows approximately the same routes over regular time intervals. For example, people wake up at the same time and generally follow the same route to their work everyday. Given that an object's trajectory $\mathcal{N}$ is decomposed into $\lfloor \frac{\mathcal{N}}{\mathcal{T}_{\mathcal{P}}} \rfloor$ sub-trajectories (see Figure 2b). $\mathcal{T}_{\mathcal{P}}$ is data-dependent and has no definite value (e.g. 'a day', 'a year'). Essentially, a periodic pattern is a closed swarm discovered from $\lfloor \frac{\mathcal{N}}{\mathcal{T}_{\mathcal{P}}} \rfloor$ sub-trajectories. As we have provided the definition of a closed swarm, we will mainly focus on closed swarm mining below.



(a) An illustrative example.

(b) A swarm from the illustrative example.

Fig. 3. An illustrative example

2.2 Related Work

As we mentioned before, many approaches have been proposed to extract patterns. For instance, Gudmundsson and Van Kreveld [1] define a flock pattern, in which the same set of objects stay together in a circular region with a predefined radius, Kalnis et al. [4] propose the notion of *moving clusters*. Jeung et al. [3] define a convoy pattern and propose three algorithms $CMC, CuTS, CuTS^*$ that incorporate trajectory simplification techniques in the first step. Then, the authors proposed to interpolate the trajectories by creating virtual time points and by applying density measurements on trajectory segments. Additionally, the convoy is defined as a candidate when it has at least *k* clusters during *k* consecutive timestamps.

Recently, Zhenhui Li et al. [6] propose the concept of swarm and closed swarm and the *ObjectGrowth* algorithm to extract closed swarm patterns. The ObjectGrowth method is a depth-first-search of all subsets of O_{DB} through a pre-order tree traversal. To speed up the search process, they propose two pruning rules. *Apriori Pruning* and *Backward Pruning* are used to stop traversal the subtree when we find further traversal that cannot satisfy min_t and closure property. After pruning the invalid candidates, a *ForwardClosure checking* is used to determine whether a pattern is a closed swarm. In [2], Hwang et al. propose two algorithms to mine group patterns, known as the *Apriori-like Group Pattern mining* algorithm and *Valid Group-Growth* algorithm. The former explores the Apriori property of valid group patterns and the latter is based on idea similar to the FP-growth algorithm. Recently in [7], A. Calmeron proposes a frequent itemset-based approach for flock identification purposes.

Even if these approaches are very efficient they suffer the problem that they only extract a specific kind of pattern. When considering a dataset, it is quite difficult, for the decision maker, to know in advance the kind of patterns embedded in the data. Therefore proposing an approach able to automatically extract all these different kinds of patterns can be very useful and this is the problem we address in this paper and that will be developed in the next sections.

3 Spatio-temporal Patterns in Itemset Context

Basically, patterns are evolution of clusters over time. Additionally, if clusters share some characteristics (e.g. share some objects), they could be a pattern. The main problem essentially is to efficiently combine items (clusters) to find itemsets (a set of

clusters) which share some characteristics or satisfying some properties to be considered as a pattern. To describe cluster evolution, spatio-temporal data is presented as a cluster matrix from which patterns can be extracted.

Definition 4. *Cluster Matrix. Given a set of clusters* $C_{DB} = \{C_1, C_2, \ldots, C_n\}$ *where* $C_i = \{c_{i_1 t_i}, c_{i_2 t_i}, \ldots, c_{i_m t_i}\}$ *is a set of clusters at timestamps* t_i*. A cluster matrix is thus a matrix of size* $|O_{DB}| \times |C_{DB}|$*. Each row represents an object and each column represents a cluster. The value of the cluster matrix cell,* (o_i, c_j) *is 1 (resp. empty) if* o_i *is in (resp. is not in) cluster* c_j*.*

For instance, the data from Figure 3a is presented in a cluster matrix in Table 2. The matrix cell (o_1-c_{11}) is 1 because o_1 belongs to the cluster c_{11} at timestamp t_1. Meanwhile, the matrix cell (o_4-c_{11}) is empty.

By presenting data in a cluster matrix, each object acts as a transaction while each cluster c_j stands for an item. Additionally, an itemset can be formed as $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$ with life time $T_\Upsilon = \{t_{a_1}, t_{a_2}, \ldots, t_{a_p}\}$ where $t_{a_1} < t_{a_2} < \ldots < t_{a_p}$, $\forall a_i : t_{a_i} \in T_{DB}, c_{t_{a_i}} \in C_{a_i}$. The support of the itemset Υ, denoted $\sigma(\Upsilon)$, is the number of common objects in every items belonging to Υ, $O(\Upsilon) = \bigcap_{i=1}^{p} c_{t_{a_i}}$. Additionally, the length of Υ, denoted $|\Upsilon|$, is the number of items or timestamps ($= |T_\Upsilon|$). For instance, in Table 2, for $\varepsilon = 2$ we have: $\Upsilon = \{c_{11}, c_{12}\}$ veryfying $\sigma(\Upsilon) = 2$, $|\Upsilon| = 2$. Every items of Υ, c_{11} and c_{12}, are in the transactions (resp. objects) o_1, o_2. Now, we define useful properties to extract the patterns from frequent itemsets as follows:

Property 1. Closed Swarm. Given a frequent itemset $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$. $(O(\Upsilon), T_\Upsilon)$ is a closed swarm if and only if:

$$\begin{cases} (1) : \sigma(\Upsilon) \geq \varepsilon \textit{ and } |\Upsilon| \geq min_t \\ (2) : \nexists \Upsilon' \textit{ s.t } O(\Upsilon) \subset O(\Upsilon'), T_{\Upsilon'} = T_\Upsilon \textit{ and } (O(\Upsilon'), T_\Upsilon) \textit{ is a swarm.} \\ (3) : \nexists \Upsilon' \textit{ s.t. } O(\Upsilon') = O(\Upsilon), T_\Upsilon \subset T_{\Upsilon'} \textit{ and } (O(\Upsilon), T_{\Upsilon'}) \textit{ is a swarm.} \end{cases} \quad (5)$$

Proof. After construction, we obtain $(O(\Upsilon), T_\Upsilon)$ which is a swarm. Additionally, if $\nexists \Upsilon'$ s.t $O(\Upsilon) \subset O(\Upsilon'), T_{\Upsilon'} = T_\Upsilon$ and $(O(\Upsilon'), T_\Upsilon)$ is a swarm then $(O(\Upsilon), T_\Upsilon)$ cannot be enlarged in terms of objects. Therefore, it satisfies the *object-closed* condition. Furthermore, if $\nexists \Upsilon'$ s.t. $O(\Upsilon') = O(\Upsilon), T_\Upsilon \subset T_{\Upsilon'}$ and $(O(\Upsilon), T_{\Upsilon'})$ is a swarm then $(O(\Upsilon), T_\Upsilon)$ cannot be enlarged in terms of lifetime. Therefore, it satisfies the *time-closed* condition. Consequently, $(O(\Upsilon), T_\Upsilon)$ is a swarm and it satisfies *object-closed* and *time-closed* conditions and therefore $(O(\Upsilon), T_\Upsilon)$ is a closed swarm according to the *Definition 1*.

For instance, in Figure 3b, for the frequent itemset $\Upsilon = \{c_{11}, c_{13}\}$ we have $(O(\Upsilon) = \{o_1, o_2, o_3\}, T_\Upsilon = \{t_1, t_3\})$ which is a closed swarm with support threshold $\varepsilon = 2$ and $min_t = 2$. We can notice that $\sigma(\Upsilon) = 3 > \varepsilon$ and $|\Upsilon| = 2 \geq min_t$.

In this paper, we do not provide the properties and proof for convoys, moving clusters which are basically extended by adding some conditions to *Property 1*. For instance, a convoy is a swarm which satisfies the consecutiveness in terms of time condition. For moving clusters [4], they need to share some objects between two timestamps (integrity proportion). Regarding to periodic patterns, the main difference in periodic pattern mining is the input data while the property is similar to *Property 1*. With a slightly modifying cluster matrix such as "each object o becomes a sub-trajectory", we can extract periodic patterns by applying *Property 1*.

Please remember that group pattern is a set of disjointed convoys. Therefore, the group pattern property is as follows:

Property 2. Group Pattern. Given a frequent itemset $\Upsilon = \{c_{t_{a_1}}, c_{t_{a_2}}, \ldots, c_{t_{a_p}}\}$, min_{wei} and min_c, a set of consecutive time segments $T_{\mathcal{S}} = \{s_1, s_2, \ldots, s_n\}$. $(O(\Upsilon), T_{\mathcal{S}})$ is a group pattern if and only if:

$$\begin{cases} (1) : |T_{\mathcal{S}}| \geq min_c. \\ (2) : \forall s_i, s_i \subseteq T_\Upsilon, |s_i| \geq min_t. \\ (3) : \bigcap_{i=1}^{n} s_i = \emptyset, \bigcap_{i=1}^{n} O(s_i) = O(\Upsilon). \\ (4) : \forall s \notin T_{\mathcal{S}}, s \textit{ is a convoy, } O(\Upsilon) \not\subseteq O(s). \\ (5) : \frac{\sum_{i=1}^{n} |s_i|}{|T|} \geq min_{wei}. \end{cases} \tag{6}$$

Proof. If $|T_{\mathcal{S}}| \geq min_c$ then we know that at least min_c consecutive time intervals s_i in $T_{\mathcal{S}}$. Furthermore, if $\forall s_i, s_i \subseteq T_\Upsilon$ then we have $O(\Upsilon) \subseteq O(s_i)$. Additionally, if $|s_i| \geq min_t$ then $(O(\Upsilon), s_i)$ is a convoy (*Definition 2*). Now, $T_{\mathcal{S}}$ actually is a set of convoys of $O(\Upsilon)$ and if $\bigcap_{i=1}^{n} s_i = \emptyset$ then $T_{\mathcal{S}}$ is a set of disjointed convoys. A little bit further, if $\forall s \notin T_{\mathcal{S}}, s$ is a convoy and $O(\Upsilon) \not\subseteq O(s)$ then $\nexists T_{\mathcal{S}'}$ s.t. $T_{\mathcal{S}} \subset T_{\mathcal{S}'}$ and $\bigcap_{i=1}^{|T_{\mathcal{S}'}|} O(s_i) = O(\Upsilon)$. Therefore, $(O(\Upsilon), T_{\mathcal{S}})$ cannot be enlarged in terms of *number of convoys*. Similarly, if $\bigcap_{i=1}^{n} O(s_i) = O(\Upsilon)$ then $(O(\Upsilon), T_{\mathcal{S}})$ cannot be enlarged in terms of *objects*. Consequently, $(O(\Upsilon), T_{\mathcal{S}})$ is a closed swarm of disjointed convoys because $|O(\Upsilon)| \geq \varepsilon, |T_{\mathcal{S}}| \geq min_c$ and $(O(\Upsilon), T_{\mathcal{S}})$ cannot be enlarged (*Definition 1*). Finally, if $(O(\Upsilon), T_{\mathcal{S}})$ satisfies condition (5) then it is a valid group pattern due to *Definition 3*.

To show the fact that from an itemset mining algorithm we are able to extract the set of all spatio-temporal patterns, we propose the following lemma.

Lemma 1. *Let $FI = \{\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_l\}$ be the frequent itemsets being mined from the cluster matrix ($min_{sup} = \varepsilon$). All swarms and group patterns can be extracted from FI.*

Proof. Let us assume that (O, T) is a swarm. Note, $T = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$. According to the *Definition 1* we know that $|O| \geq \varepsilon$. If (O, T) is a swarm then $\forall t_{a_i} \in T, \exists c_{t_{a_i}}$ s.t. $O \subseteq c_{t_{a_i}}$ therefore $\bigcap_{i=1}^{m} c_{t_{a_i}} = O$. Additionally, we know that $\forall c_{t_{a_i}}$, $c_{t_{a_i}}$ is an item so $\exists \Upsilon = \bigcup_{i=1}^{m} c_{t_{a_i}}$ is an itemset and $O(\Upsilon) = \bigcap_{i=1}^{m} c_{t_{a_i}} = O, T_\Upsilon = \bigcup_{i=1}^{m} t_{a_i} = T$. Therefore, $(O(\Upsilon), T_\Upsilon)$ is a swarm. So, (O, T) is extracted from Υ. Furthermore, $\sigma(\Upsilon) = |O(\Upsilon)| = |O| \geq \varepsilon$ then Υ is a frequent itemset and $\Upsilon \in FI$. Finally, $\forall (O, T)$ s.t. if (O, T) is a swarm then $\exists \Upsilon$ s.t. $\Upsilon \in FI$ and (O, T) can be extracted from Υ, we can conclude $\forall$ swarm (O, T), it can be mined from FI.

Essentially, the set of all convoys, closed swarms, moving clusters are subset of the set of all swarms. By adding constraints such as "consecutive lifetime", "time-closed", "object-closed", "integrity proportion" to swarms, we can retrieve convoys, closed swarms and moving clusters. By applying *Lemma 1*, we retrieve all swarms from frequent itemsets. Consequently, convoys and moving clusters can be completely extracted from frequent itemsets.

4 FCI-Based Spatio-temporal Pattern Mining Algorithms

In this section, we propose two approaches, *GeT_Move* and *Incremental GeT_Move*, to efficiently extract patterns. The global process is illustrated in Figure 4. In the first step, a clustering approach is applied at each timestamp to group objects into different clusters. For each timestamp t_a, we thus have a set of clusters C_a with $\forall c_{kt_a} \in C_a, c_{kt_a} \subseteq O_{DB}$. Spatio-temporal data can thus be converted to a cluster matrix.

After generating the cluster matrix CM, a FCI mining algorithm is applied on CM to extract all the FCIs. By scanning them and checking properties, we can obtain the patterns. In this paper, we apply the LCM algorithm [8] to extract FCIs as it is known to be a very efficient algorithm. In LCM algorithm's process, we discard some useless candidate itemsets. In spatio-temporal patterns, clusters must belong to different timestamps and therefore items (resp. clusters) which form a FCI must be in different timestamps. Consequently, FCIs which include more than 1 item in the same timestamp will be discarded. Thanks to this characteristic, we now have the maximum length of the FCIs which is the number of timestamps $|T_{DB}|$. Additionally, the LCM search space only depends on the number of objects $|O_{DB}|$ and the maximum length of itemsets $|T_{DB}|$. Therefore, by using LCM and the above characteristic, *GeT_Move* is not affected by the number of clusters and therefore the computing time can be greatly reduced.

The pseudo code of *GeT_Move* is described in *Algorithm 1*. The core of *GeT_Move* algorithm is based on the LCM algorithm which has been modified by adding the pruning rule and by extracting patterns from FCIs (line 2). Whenever a closed itemset is detected, the PatternMining sub-function (lines 3-25) is invoked to check properties of the itemset X for extracting spatio-temporal patterns.

Naturally, in real world applications, the objects tend to move together in short interval meanwhile their movements can be different in long interval. For instance, see Figure 5a, objects $\{o_1, o_2, o_3, o_4\}$ move together during first 100 timestamps and after that o_1, o_2 stay together while o_3, o_4 move together in another direction. The problem here is that if we apply *GeT_Move* on the whole dataset, the extraction of the itemsets can be very time consuming. To deal with the issue, we propose the *Incremental GeT_Move* algorithm. The main idea is to split the trajectories (resp. cluster matrix CM) into short intervals, called blocks. By applying FCI mining on each block, the data can then be compressed into local FCIs. Additionally, the length of itemsets and the number of items can be greatly reduced. For instance, see Figure 5, if we consider $[t_1, t_{100}]$ as a

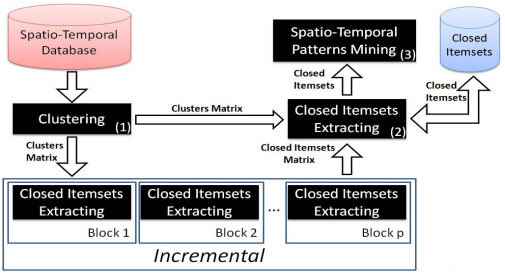


Fig. 4. The main process

Algorithm 1. GeT_Move

Input : int ε, int min_t, set of items C_{DB}, double θ, int min_c, double min_{wei}
begin
  LCM_PatternMining(C_{DB}, ε);
PatternMining(X, min_t)
begin
  if $|X| \geq min_t$ **then**
    output X; /*Closed Swarm*/
    $gPattern := \emptyset$; $convoy := \emptyset$; $mc := \emptyset$;
    for $k := 1$ *to* $|X - 1|$ **do**
      if $x_k.time = x_{(k+1)}.time - 1$ **then**
        $convoy := convoy \cup x_k$;
        if $\frac{|\mathcal{T}(x_k) \cap \mathcal{T}(x_{k+1})|}{|\mathcal{T}(x_k) \cup \mathcal{T}(x_{k+1})|} \geq \theta$ **then**
          $mc := mc \cup x_k$;
        else
          if $|mc \cup x_k| \geq min_t$ **then**
            output $mc \cup x_k$; /*MovingCluster*/
          $mc := \emptyset$;
      else
        if $|convoy \cup x_k| \geq min_t$ *and* $|\mathcal{T}(convoy \cup x_k)| = |\mathcal{T}(X)|$ **then**
          output $convoy \cup x_k$; /*Convoy*/
          $gPattern := gPattern \cup (convoy \cup x_k)$;
        if $|mc \cup x_k| \geq min_t$ **then**
          output $mc \cup x_k$; /*MovingCluster*/
        $convoy := \emptyset$; $mc := \emptyset$;
    if $|gPattern| \geq min_c$ *and* $size(gPattern) \geq min_{wei}$ **then**
      output $gPattern$; /*Group Pattern*/

Where: X is itemset, $\mathcal{T}(X)$ is list of tractions that X belongs to, $x_k.time$ is time index of item x_k, $|\mathcal{T}(convoy)|$ is the number of transactions that the $convoy$ belongs to, $|gPattern|$ and $size(gPattern)$ respectively are the number of convoys and the proportion of total length of the convoys in $gPattern$ to T_{DB}.

block and $[t_{101}, t_{200}]$ as another block, the maximum length of itemsets in both blocks is 100 (instead of 200). Additionally, the original data can be greatly compressed (e.g. Figure 5b) and only 3 items remain: $ci_{11}, ci_{12}, ci_{22}$.

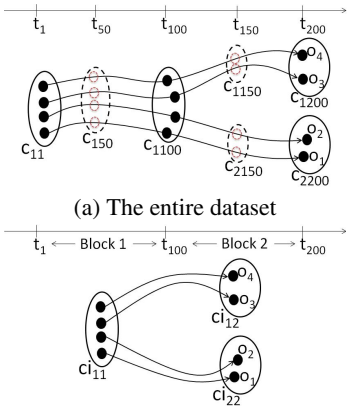
Definition 5. *Block. Given a set of timestamps $T_{DB} = \{t_1, t_2, \ldots, t_n\}$, a cluster matrix CM. CM is vertically split into equivalent (in terms of intervals) smaller cluster matrices and each of them is a block b. Assume T_b is a set of timestamps of block b, $T_b = \{t_1, t_2, \ldots, t_k\}$, thus we have $|T_b| = k \leq |T_{DB}|$.*

Assume that there is a set of blocks $B = \{b_1, b_2, \ldots, b_p\}$ with $|T_{b_1}| = \ldots = |T_{b_p}|$, $\bigcup_{i=1}^{p} b_i = CM$ and $\bigcap_{i=1}^{p} b_i = \emptyset$. Given a set of FCI collections $CI = \{CI_1, CI_2, \ldots, CI_p\}$ where CI_i is mined from block b_i. CI is presented as a *closed itemset matrix* which is formed by horizontally connecting all local FCIs: $CIM = \bigcup_{i=1}^{p} CI_i$.

Definition 6. *Closed Itemset Matrix (CIM). Closed itemset matrix is a cluster matrix such as: 1) Timestamp t now becomes a block b. 2) Item c is a FCI ci.*

For instance, see Table 3, we have two sets of FCIs $CI_1 = \{ci_{11}\}, CI_2 = \{ci_{12}, ci_{22}\}$ which are respectively extracted from blocks b_1, b_2. We have CIM which is created from CI_1, CI_2 in Table 3. Now, by applying FCI mining on closed itemset matrix CIM, we retrieve all FCIs from corresponding data. Note that items (in CIM) which are in the same block cannot be in the same FCIs.

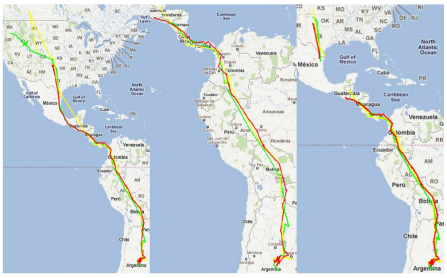
Lemma 2. *Given a cluster matrix CM which is vertically split into a set of blocks $B = \{b_1, b_2, \ldots, b_p\}$ so that $\forall \Upsilon, \Upsilon$ is a FCI and Υ is extracted from CM then Υ can be extracted from the closed itemset matrix CIM.*



(a) The entire dataset

(b) Data after applying FCI mining onblocks

Fig. 5. A case study example. (b)-$ci_{11}, ci_{12}, ci_{22}$ are FCIs extracted from block 1 and block 2.



(a) (b) (c)

Fig. 6. An example of patterns discovered from Swainsoni dataset. (a) One of discovered closed swarms, (b) One of discovered convoys, (c) One of discovered group patterns.

Algorithm 2. Incremental GeT_Move

Input : int ε, int min_t, double θ, set of Occurrence sets (blocks) B, int min_c, double min_{wei}
1 **begin**
2 $K := \emptyset; CI := \emptyset;$
3 **foreach** $b \in B$ **do**
4 CI := CI.update(LCM(b, ε));
5 GeT_Move($\varepsilon, min_t, CI, \theta, min_c, min_{wei}$);

Proof. Let us assume that $\forall b_i, \exists I_i$ is a set of items belonging to b_i and therefore we have $\bigcap_{i=1}^{|B|} I_i = \emptyset$. If $\forall \Upsilon, \Upsilon$ is a FCI extracted from CM then Υ is formed as $\Upsilon = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ where γ_i is a set of items s.t. $\gamma_i \subseteq I_i$. Additionally, Υ is a FCI and $O(\Upsilon) = \bigcap_{i=1}^{p} O(\gamma_i)$ then $\forall O(\gamma_i), O(\Upsilon) \subseteq O(\gamma_i)$. Furthermore, we have $|O(\Upsilon)| \geq \varepsilon$; therefore, $|O(\gamma_i)| \geq \varepsilon$ so γ_i is a frequent itemset. Assume that $\exists \gamma_i, \gamma_i \notin CI_i$ then $\exists \Psi, \Psi \in CI_i$ s.t. $\gamma_i \subseteq \Psi$ and $\sigma(\gamma_i) = \sigma(\Psi), O(\gamma_i) = O(\Psi)$. Note that Ψ, γ_i are from b_i. Remember that $O(\Upsilon) = O(\gamma_1) \cap O(\gamma_2) \cap \ldots \cap O(\gamma_i) \cap \ldots \cap O(\gamma_p)$ then we have: $\exists \Upsilon'$ s.t. $O(\Upsilon') = O(\gamma_1) \cap O(\gamma_2) \cap \ldots \cap O(\Psi) \cap \ldots \cap O(\gamma_p)$. Therefore, $O(\Upsilon') = O(\Upsilon)$ and $\sigma(\Upsilon') = \sigma(\Upsilon)$. Additionally, we know that $\gamma_i \subseteq \Psi$ so $\Upsilon \subseteq \Upsilon'$. Consequently, we obtain $\Upsilon \subseteq \Upsilon'$ and $\sigma(\Upsilon) = \sigma(\Upsilon')$. Therefore, Υ is not a FCI. That violates the assumption and therefore we have: if $\exists \gamma_i, \gamma_i \notin CI_i$ therefore Υ is not a FCI. Finally, we can conclude that $\forall \Upsilon, \Upsilon = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ is a FCI extracted from CM, $\forall \gamma_i \in \Upsilon$, γ_i must belong to CI_i and γ_i is an item in closed itemset matrix CIM. Therefore, Υ can be retrieved by applying FCI mining on CIM.

By applying *Lemma 2*, we can obtain all the FCIs and from the itemsets, patterns can be extracted. Note that the Incremental GeT_Move does not depend on the length restriction min_t. The reason is that min_t is only used in Spatio-Temporal Pattern Mining step.

Whatever min_t ($min_t \geq$ block size or $min_t \leq$ block size), Incremental GeT_Move can extract all the FCIs and therefore the final results are the same. The pseudo code of *Incremental GeT_Move* is described in *Algorithm 2*. The main different between the code of *Incremental GeT_Move* and *GeT_Move* is the *update* function. In this function, we step by step generate the closed itemsets matrix from extracted closed itemsets in blocks (line 4). Next, we apply *GeT_Move* to extract patterns (line 5).

5 Experimental Results

A comprehensive performance study has been conducted on real datasets and synthetic datasets. All the algorithms are implemented in C++, and all the experiments are carried out on a Ubuntu 11.10, g++ version 4.6.1 and 2.8GHz Intel Core i7 system with 4GB Memory. The source code is available on our online demonstration system[2]. As in [6, 3], we report the results on *Swainsoni dataset*[3] includes 43 objects evolving over time and 764 different timestamps. Additionally, we first use linear interpolation to fill in the missing data and then DBScan [5] ($min_{Pts} = 2, eps = 0.001$) is applied to generate clusters at each timestamp. In the comparison, we employ $CMC, CuTS^*$[4][3] (convoy) and $Object\ Growth$ (closed swarm). Note that, in [6], $ObjectGrowth$ outperforms *VG-Growth* [2] (group patterns) in terms of performance and therefore we only consider $ObjectGrowth$.

Effectiveness. We proved that mining spatio-temporal patterns can be similarly mapped into itemsets mining issue. Therefore, in theoretical way, our approaches can provide the correct results. Experimentally, we do a further comparison, we first obtain the spatio-temporal patterns by applying $CMC, CuTS^*$, $ObjectGrowth$ as well as our approaches. To apply our algorithms, each block b contains 25 timestamps and to retrieve all the patterns, the default value of ε is set to 2, min_t is 1, min_c is 1 and min_{wei} is 0. Note that the default values are the hardest conditions for examining the algorithms. Then in the following we mainly focus on different values of min_t in order to obtain different sets of convoys, closed swarms and group patterns. The results show that our proposed approaches obtain the same results compared to the traditional algorithms. An example of extracted patterns is illustrated in Figure 6.

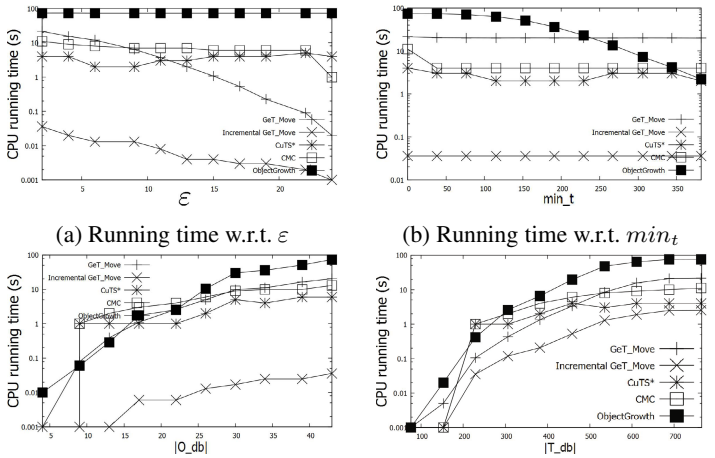
In the efficiency reported experiments, GeT_Move and Incremental GeT_Move extract closed swarms, convoys and group patterns while $CMC, CuTS^*$ only extract convoys and $ObjectGrowth$ only extracts closed swarms. Additionally, all the algorithms are applied on cluster matrices, thus clustering cost was not considered.

Efficiency. Figure 7a shows running time w.r.t. ε. It is clear that our approaches outperform other algorithms. ObjectGrowth is the lowest one and the main reason is that with low min_t (default $min_t = 1$), the *Apriori Pruning* rule (the most efficient pruning rule) is no longer effective. Therefore, the search space is greatly enlarged ($2^{|O_{DB}|}$ in the worst case). Additionally, there is no pruning rule for ε and therefore the change

[2] `www.lirmm.fr/~phan/index.jsp`

[3] `http://www.movebank.org`

[4] The source code of $CMC, CuTS^*$ is available at `http://lsirpeople.epfl.ch/jeung/source_codes.htm`



(a) Running time w.r.t. ε (b) Running time w.r.t. min_t

(c) Running time w.r.t. $|O_{DB}|$ (d) Running time w.r.t. $|T_{DB}|$

Fig. 7. Running time on Swainsoni Dataset

of ε does not directly affect the running time of ObjectGrowth. A little bit further, GeT_Move is lower than Incremental GeT_Move. The main reason is that GeT_Move has to process with large number of items and long itemsets. While, thanks to blocks, the number of items is greatly reduced and itemsets are not long as the ones in GeT_Move. Figure 7b shows running time w.r.t. min_t. In almost all cases, our approaches outperform other algorithms. Note that min_t does not affect to the running time of the proposed algorithms and the reason is that min_t is only used in pattern mining step. Figure 7c-d show the running time when varying $|O_{DB}|$ and $|T_{DB}|$ respectively. In all the cases, Incremental GeT_Move outperforms other algorithms.

In addition, the other experimental results on different real and synthetic datasets are available on our demo system website. Some interesting experiments on the number of patterns, algorithm scalability, optimal block-sizes and also a parameter free version of Incremental GeT_Move are integrated in the online system. Interested readers may refer to "www.lirmm.fr/ ∼phan/GeTMove.html".

6 Conclusion and Discussion

In this paper, we propose unifying incremental approaches to automatically extract different kinds of spatio-temporal patterns by applying FCI mining techniques. Their effectiveness and efficiency have been evaluated by using real and synthetic datasets. Experiments show that our approaches outperform traditional ones.

One next issue we plan to address is how to take into account the arrival of new objects which were not available for the first extraction. Now, we can store the result to improve the process when new object movements arrive. But, in this approach, we take the hypothesis is that the number of objects remains the same. However in some applications these objects could be different.

References

1. Gudmundsson, J., van Kreveld, M.: Computing Longest Duration Flocks in Trajectory Data. In: GIS 2006, New York, NY, USA, pp. 35–42 (2006)
2. Wang, Y., Lim, E.-P., Hwang, S.-Y.: Efficient Mining of Group Patterns from User Movement Data. In: DKE 2006, pp. 240–282 (2006)
3. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of Convoys in Trajectory Databases. In: PVLDB 2008, vol. 1(1), pp. 1068–1080 (2008)
4. Kalnis, P., Mamoulis, N., Bakiras, S.: On Discovering Moving Clusters in Spatio-temporal Data. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 364–381. Springer, Heidelberg (2005)
5. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: KDD 1996, Portland, pp. 226–231 (1996)
6. Li, Z., Ding, B., Han, J., Kays, R.: Swarm: Mining Relaxed Temporal Moving Object Clusters. In: VLDB 2010, Singapore, pp. 723–734 (2010)
7. Romero, A.O.C.: Mining Moving Flock Patterns in Large Spatio-Temporal Datasets Using a Frequent Pattern Mining Approach. Master Thesis, University of Twente (March 2011)
8. Uno, T., Kiyomi, M., Arimura, H.: LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In: ICDM FIMI (2004)
9. Jensen, C.S., Lin, D., Ooi, B.C.: Continuous Clustering of Moving Objects. In: KDE, pp. 1161-1174 (2007) ISSN: 1041-4347
10. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, Indexing, and Querying Historical Spatiotemporal Data. In: SIGKDD 2004, pp.236-245 (2004)
11. Han, J., Li, Z., Tang, L.A.: Mining Moving Object, Trajectory and Traffic Data. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 485–486. Springer, Heidelberg (2010)

Fuzzy Frequent Pattern Mining in Spike Trains

David Picado Muiño, Iván Castro León, and Christian Borgelt

European Centre for Soft Computing
Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Spain
{david.picado,ivan.castro,christian.borgelt}@softcomputing.es

Abstract. We present a framework for characterizing spike (and spike-train) synchrony in parallel neuronal spike trains that is based on identifying spikes with what we call *influence maps*: real-valued functions describing an influence region around the corresponding spike times within which possibly graded synchrony with other spikes is defined. We formalize two models of synchrony in this framework: the bin-based model (the almost exclusively applied model in the literature) and a novel, alternative model based on a continuous, graded notion of synchrony, aimed at overcoming the drawbacks of the bin-based model. We study the task of identifying frequent (and synchronous) neuronal patterns from parallel spike trains in our framework, formalized as an instance of what we call the fuzzy frequent pattern mining problem (a generalization of standard frequent pattern mining) and briefly evaluate our synchrony models on this task.

1 Introduction

The principles of *neural coding* (i.e., how information is represented in biological neural networks) are still not well understood and continue to be the topic of ongoing debate. Several coding schemes have been proposed in the neuroscience literature. Among them, one of the most prominent is the so-called *temporal coordination scheme* (see, e.g., [1]), first advocated by D.O. Hebb [2] and driven by more recent physiological and anatomical evidence, according to which the coordinated emission of *spikes* (i.e., electrical impulses, also called *action potentials*), in particular *synchronous* spiking (see, e.g., [3–5]) by groups of neurons plays a major role in neuronal information processing.

In order to understand the principles of coordinated neuronal activity and neural coding it is necessary to observe the activity of multiple neurons simultaneously. This is nowadays possible due to improvements in multiple-electrode recording (see, e.g., [6]), which allows to monitor the activity of hundreds of neurons. Such recordings are typically represented by sequences of points in time (i.e., point processes) and referred to as *parallel spike trains*.

It is not a trivial matter to determine what constitutes synchronous spiking in parallel spike trains. *Exact* spike-time coincidence cannot be expected and thus a characterization of spike synchrony in such terms is not suitable.

We present in Section 2 a very general, flexible framework for characterizing and quantifying synchrony among spikes (spike synchrony) and spike trains

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 289–300, 2012.
© Springer-Verlag Berlin Heidelberg 2012

(spike-train synchrony), which is able to accommodate graded, continuous characterizations of synchrony. The framework is based on the identification of spikes with what we call *influence maps*: real-valued functions describing an *influence region* around the corresponding spike times within which synchrony with other spikes is defined. Together with our influence maps, a class of so-called *synchrony operators* (to quantify spike synchrony) and a class of *support operators* (to quantify spike-train synchrony) are defined, based on what we consider desirable criteria in the assessment of synchrony among spikes and spike trains.

Within this framework, two concrete models for characterizing and quantifying spike (and spike-train) synchrony are described in Section 3. The first model formalizes the so-called *bin-based* method, which is the almost exclusively applied one in the field and based on time-bin discretization: spikes are binned in time intervals of equal length. In this model, two or more spikes are considered to be synchronous if they lie in the same time bin. The second model characterizes synchrony in a graded fashion and aims at overcoming the main drawbacks of the bin-based model by avoiding time-bin discretization and bivalence in the amount of synchrony among spikes. Graded, non-bivalent (i.e., fuzzy, as in fuzzy set theory—see, e.g., [7]) synchrony itself is not an entirely new concept, though. Implicit notions of presumably non-bivalent, graded synchrony can be found in some papers on topics related to synchronous spiking, such as [8, 9]. However, in these approaches (at least in the two just mentioned) a formal derivation and treatment of such a concept is a non-trivial task.

In Section 4 we consider the problem of identifying frequent neuronal patterns in parallel spike trains. For the models presented in Section 3 the problem is formalized as an instance of what we call the *fuzzy frequent pattern mining problem*—a generalization of standard frequent pattern mining. The task of identifying frequent neuronal patterns is of particular relevance in itself but also in relation to tasks concerned with identifying other sets of neurons that are characterized by different synchrony requirements (such as, e.g., synchronous patterns, called *unitary events* in [10], or *neural assemblies*, as defined in e.g. [11, 12]).

In Section 5 we provide a summary of a first evaluation of our graded, continuous synchrony model (as presented in Section 3) for the identification of frequent neuronal patterns on artificially generated spike trains as an alternative to the commonly used bin-based model. Section 6 briefly summarizes our discussion.

2 Synchronous Spiking

Let N be a (finite) set of neurons. We are given parallel spike trains, one for each neuron in N, formalized as spike-time sequences of the form $\{t_1^a, ..., t_{k_a}^a\} \subset (0, T]$, for $a \in N$, where k_a is the number of times neuron a fires in the interval $(0, T]$. We denote the set of all these sequences by $\mathcal{S}$. Collections of sequences like $\mathcal{S}$ constitute the raw data on which we build our framework.

2.1 Influence Maps

The first step in setting our framework consists in our characterization of spikes. We do so by considering an *influence region* around each spike time in $\mathcal{S}$ together with a possibly varying *influence degree* along that region, which we formalize by introducing what we call *influence maps*.

Definition 1. *A function* $f : \mathbb{R} \to [0, \infty)$ *is called an* influence map *if*

$$\int_{-\infty}^{\infty} f(x)dx = 1.$$

The class of all possible influence maps is denoted by $\mathcal{F}$.

For any $f \in \mathcal{F}$ the set $\{x \in \mathbb{R} \mid f(x) > 0\}$ gives us the *influence region* of f (or, more precisely, of the spike represented by the influence map f). Intuitively, the influence region of a spike captures the times at which other spikes can be considered to be synchronous to some (positive) degree, as is detailed below.

It is worth noting that, although an influence map is formally a probability density function, we do *not* impose such an interpretation (i.e., an influence map is *not* meant to represent uncertainty about the actual location of a spike).

2.2 Spike Synchrony

In order to characterize spike synchrony based on our identification of spikes with influence maps in $\mathcal{F}$, we define a class of operators, called *synchrony operators*, aimed at quantifying the *degree of synchrony* of a collection of spikes.

We define synchrony operators on *multisets* (sometimes also called *bags*) over $\mathcal{F}$ (i.e., collections of elements of $\mathcal{F}$ that can contain multiple copies of the same element), which are formally defined as pairs $\langle \mathcal{F}, h \rangle$, with $h : \mathcal{F} \to \mathbb{N} \cup \{0\}$. Intuitively, the function h "counts" the occurrences of the elements of $\mathcal{F}$ in $\langle \mathcal{F}, h \rangle$. Throughout this paper, whenever convenient, we use set notation for multisets, for example, $\{f_1, f_1, f_2, f_2, f_3\}$ instead of $\langle \mathcal{F}, h \rangle$, with $h(f_1) = 2$, $h(f_2) = 2$, $h(f_3) = 1$ and $h(f) = 0$ for any $f \in \mathcal{F} \setminus \{f_1, f_2, f_3\}$. The collection of all possible multisets over $\mathcal{F}$ is denoted by $\mathscr{M}(\mathcal{F})$.

The use of multisets over $\mathcal{F}$ (instead of simple subsets of $\mathcal{F}$) is motivated by the fact that distinct spikes of different neurons in our recordings can be represented by the same influence maps, for example, if their respective spike times coincide exactly (but possibly also in other situations, see below).

In the definition of the class of synchrony operators and other classes of operators defined later we make use of the *sum operator* on multisets over $\mathcal{F}$, denoted by $\uplus$, and defined for two multisets $\langle \mathcal{F}, h_1 \rangle$ and $\langle \mathcal{F}, h_2 \rangle$ as follows:

$$\langle \mathcal{F}, h_1 \rangle \uplus \langle \mathcal{F}, h_2 \rangle = \langle \mathcal{F}, h_1 + h_2 \rangle = \langle \mathcal{F}, h \rangle,$$

where, for all $f \in \mathcal{F}$, $h(f) = h_1(f) + h_2(f)$.[1] It is worth noting that $\uplus$ is commutative and associative and thus its definition can be extended trivially to

[1] For a wider insight into basic multiset theory and, in particular, binary operations on multisets see e.g. [13].

more than two multisets. It is also straightforward to define the *cardinality* of a multiset $G = \langle \mathcal{F}, h \rangle$ as $|G| = |\langle \mathcal{F}, h \rangle| = \sum_{f \in \mathcal{F}} h(f)$.

Definition 2. *The function* $Sync : \mathscr{M}(\mathcal{F}) \to [0,1]$ *is called a* synchrony operator *on* $\mathcal{F}$ *if* $Sync(G) \geq Sync(G \uplus H)$ *for* G, H *multisets in* $\mathscr{M}(\mathcal{F})$.

Definition 2 aims at formalizing the requirement that the degree of synchrony of a certain collection of spikes cannot increase if we add another spike to it.

Let us consider now a subset $A = \{a_1, ..., a_k\} \subseteq N$ of neurons, for some natural number $k \leq |N|$, and $G = \{f^{a_1}, ..., f^{a_k}\} \in \mathscr{M}(\mathcal{F})$, with f^{a_i} the influence map corresponding to a spike time of neuron a_i in $\mathcal{S}$, for all $i \in \{1, ..., k\}$. We refer to the joint-spike event G as an A-event (i.e., an event in $\mathcal{S}$ for the neuronal pattern or subset A) and denote the multiset of all such A-events generated from $\mathcal{S}$ (i.e. one for each combination of spike times in $\mathcal{S}$, one for each neuron in A) by E^A. Notice that E^A is a multiset, since distinct combinations of spike times in $\mathcal{S}$, one for each neuron in A, may generate the same multiset $G \in \mathscr{M}(\mathcal{F})$ above. We denote by $E^A_{Sync^+}$ the subset of A-events that have a strictly positive degree of synchrony with respect to the operator *Sync*.

2.3 Spike-Train Synchrony

We now characterize and quantify spiking activity in spike trains and synchrony among them. We do so by means of the family of what we call *support* functions. We first define the set $\mathcal{H}^n$ as follows, for $n \in \mathbb{N}$:

$$\mathcal{H}^n = \{G \mid G \text{ is a multiset in } \mathscr{M}(\mathcal{F}) \text{ and } |G| = n\}.$$

That is, the set $\mathcal{H}^n$ comprises all representations of n spikes (of n neurons) by (multisets of) influence maps. It is meant to capture all possible representations of A-events for $|A| = n$ and thus all possible specific synchronous spiking events.

The collection of all possible multisets over $\mathcal{H}^n$ is denoted by $\mathscr{M}(\mathcal{H}^n)$. The use of $\mathscr{M}(\mathcal{H}^n)$ in the definition of our family of support functions is motivated by the possibility that different A-events in E^A, for some $A \subseteq N$, $|A| = n$, may be represented by the same multiset in $\mathscr{M}(\mathcal{F})$.

Definition 3. *A collection of maps* $Supp_n : \mathscr{M}(\mathcal{H}^n) \to \mathbb{R}^+ \cup \{0, \infty\}$, *for* $n \in \mathbb{N}$, *is a family of* support operators *on* $\mathcal{F}$ *if the following conditions hold:*

- ***Upward monotonicity.*** *For* $\mathcal{H}, \mathcal{G} \in \mathscr{M}(\mathcal{H}^n)$, $Supp_n(\mathcal{H}) \leq Supp_n(\mathcal{H} \uplus \mathcal{G})$.[2]
- ***Downward monotonicity.*** *For* $\mathcal{H} = \{H_1, ..., H_k\} \in \mathscr{M}(\mathcal{H}^n)$ *and* $\mathcal{H}' = \{H_1 \uplus \{f_1\}, ..., H_k \uplus \{f_k\}\} \in \mathscr{M}(\mathcal{H}^{n+1})$, *with* $f_i \in \mathcal{F}$ *for all* $i \in \{1, ..., k\}$ *and* $k \in \mathbb{N}$, *we have that* $Supp_n(\mathcal{H}) \geq Supp_{n+1}(\mathcal{H}')$.

Let $\mathcal{G} \in \mathscr{M}(\mathcal{H}^n)$ be a multiset of influence maps. We call $Supp_n(\mathcal{G})$ the *support* or *amount of synchrony* (if $n > 1$) of $\mathcal{G}$ with respect to the operator $Supp_n$.

[2] The ordering $\leq$ is defined on the extended positive reals $\mathbb{R}^+ \cup \{0, \infty\}$ as conventional.

Intuitively, a support operator $Supp_n$ is meant to aggregate the synchrony of all individual A-events (with $A \subseteq N$, $|A| = n$) over the (parallel) spike trains of the neurons in A.

Upward monotonicity encodes the intuitive idea that the spiking activity or, for $n > 1$, the amount of synchrony of a certain collection of multisets in $\mathcal{F}$ (e.g. a collection of A-events in E^A, for some $A \subseteq N$, $|A| = n$) should be no less than the spiking activity or amount of synchrony of smaller collections contained in it.

Similarly, *downward monotonicity* is intended to capture the idea that synchronous spiking in a certain set of spike trains (the amount of synchrony of such a set) should be no less than the amount of synchrony of any of its proper subsets. This generalizes the monotonicity property in Definition 2 to the amount of synchrony among spike trains.

3 Synchrony Models

The most common method in tasks related to the identification of synchronous spiking in parallel spike trains—one may even say the almost exclusively applied method—is *bin-based* (i.e., it employs time discretization): spikes are binned in time intervals of equal length (time bins) that partition the recording-time interval $(0, T]$. In this approach, two or more spikes are considered to be synchronous if they lie in the same time bin[3] and thus what characterizes synchrony is the length and boundaries of time bins (although it is only the length of time bins that gives us the *intended* characterization of spike synchrony in this approach).

The bin-based characterization of synchrony can be easily formalized in our framework. For a time-bin length $d \in (0, T]$, each $a \in N$ and time t^a in $\mathcal{S}$ we can define, for example, the following influence map in $\mathcal{F}$:

$$f_{t^a}(x) = \begin{cases} \frac{1}{d} \text{ if } x \in \left(d \left\lfloor \frac{t^a}{d} \right\rfloor, d \left\lfloor \frac{t^a}{d} \right\rfloor + d\right] \\ 0 \text{ otherwise,} \end{cases}$$

where $\left\lfloor \frac{t^a}{d} \right\rfloor$ denotes the integer part of $\frac{t^a}{d}$. For $G \in \mathscr{M}(\mathcal{F})$ a multiset of influence maps, we define our synchrony operator as follows:

$$Sync(G) = \begin{cases} 1 \text{ if } \exists x \in \mathbb{R} : \min_{f_t \in G} f_t(x) > 0 \\ 0 \text{ otherwise.} \end{cases}$$

Synchronous spiking in the spike trains of neurons in $A \subseteq N$ can be quantified by $Supp_{|A|}(E^A)$, which we simply define here as follows:

$$Supp_{|A|}(E^A) = \sum_{G \in E^A} Sync(G).$$

[3] *Clipping* is usually applied, that is, at most one spike per neuron is considered in each time bin. We follow this common practice here.

The bin-based characterization of synchrony has two main drawbacks:

- **Boundary problem.** A collection of spikes separated by a time distance smaller than the bin length may be split into different bins and thus be regarded as non-synchronous.
- **Bivalence problem.** A collection of spikes can be either (fully) synchronous or non-synchronous. Small variations in the time distance between two spikes (possibly moving one of them over a time bin boundary) may cause a jump from (full) synchrony to non-synchrony and vice versa.

A graded, continuous characterization of synchrony, able to overcome these drawbacks, is certainly desirable. We present here a model within our framework that responds to this motivation.

The first step in setting our model is to decide the form of the influence regions around spike times within which positive synchrony degrees are defined. At least in the absence of any specific knowledge about noise or jitter in our recordings, symmetry of influence regions and influence maps around the points given by the corresponding spike times seems to be a reasonable choice. Likewise, in the absence of any evidence to the contrary, influence regions of the same length may be considered for all spikes of all neurons. We follow this view and consider for our model influence regions of a certain length, say $r \in \mathbb{R}^+$, that are symmetric around the corresponding spike times (i.e., for t a spike time in $\mathcal{S}$, we consider the influence region $[t - \frac{r}{2}, t + \frac{r}{2}]$).

We now characterize our influence maps. For our model we choose (possibly) the simplest subclass of $\mathcal{F}$, given by the maps of the following form, for t a spike time in $\mathcal{S}$ and $[t - \frac{r}{2}, t + \frac{r}{2}]$ the corresponding influence region:

$$f_t(x) = \begin{cases} \frac{1}{r} \text{ if } x \in [t - \frac{r}{2}, t + \frac{r}{2}], \\ 0 \text{ otherwise.} \end{cases}$$

For every $r \in \mathbb{R}^+$ we will denote the corresponding subclass of functions by $\mathcal{F}^r$.

We define our synchrony operator *Sync* as follows, for G a multiset in $\mathscr{M}(\mathcal{F}^r)$:

$$Sync(G) = \int_{-\infty}^{\infty} \min_{f \in G} f(x) dx. \tag{1}$$

We thus characterize synchrony among spikes by means of the *minimum* operator (a natural choice in the context: all influence maps must be positive for a point in time in order for this point in time to contribute to the synchrony) and measure their degree of synchrony by means of integration (that is, we aggregate over all points in time that contribute to the synchrony).

We move now to the definition of the family of support operators $Supp_n$ for characterizing and quantifying spiking activity of neurons and synchronous spiking of groups of neurons in our model. Let us start by assuming that the spike train of any single neuron in N is represented by a collection of influence maps in $\mathcal{F}^r$ with non-overlapping influence regions. (For reasonably small r this can be expected, due to the sparseness of spike times in real neuronal recordings

and to the *refractory period* of neurons, that is, a certain period of time needs to elapse between the emission of two spikes—see, e.g., [14]). Under such an assumption the *naïve* characterization of $Supp_n$ as

$$Supp_n(\mathcal{H}) = \sum_{G \in \mathcal{H}} Sync(G), \tag{2}$$

for $\mathcal{H} \in \mathscr{M}(\mathcal{H}^n)$ and $n \in \mathbb{N}$, and the corresponding quantification of synchrony for $A \subseteq N$ given by $Supp_{|A|}(E^A)$ would constitute a reasonable choice. However, if the aforementioned assumption does not hold (i.e., if it is not the case that the influence regions of maps representing spikes emitted by the same neuron do not overlap) the above family of operators $Supp_n$ is not a family of support operators as set in Definition 3. An alternative characterization of $Supp_n$ in agreement with Definition 3 is thus needed for our model and we opt for

$$Supp_n(\mathcal{H}) = \int_{-\infty}^{\infty} \max_{G \in \mathcal{H}} (\min_{f \in G} f(x))\, dx, \tag{3}$$

for $\mathcal{H} \in \mathscr{M}(\mathcal{H}^n)$ and $n \in \mathbb{N}$. In words, we merge every collection of functions $G \in \mathcal{H}$ (e.g., every A-event in E^A for the neuronal pattern $A \subseteq N$) into the new function $\min_{f \in G} f(x)$, which characterizes synchrony in G, and integrate the maximum of all such functions over the real line. This removes the problems resulting from overlapping influence maps: overlapping influence maps can lead to overlaps of the functions $\min_{f \in G} f(x)$ that represent different A-events and thus certain time regions may be "counted" multiple times if we simply sum the synchrony of A-events. Taking the maximum eliminates this potential multiplicity and thus ensures downward monotonicity of our family of support operators.

Note that we can equivalently express $Supp_{|A|}(E^A)$, for $A \subseteq N$, as

$$Supp_{|A|}(E^A) = \int_{-\infty}^{\infty} \min_{a \in A} (\max_{f \in G_a} f(x))\, dx, \tag{4}$$

with $G_a = \{f_1^a, ..., f_{k_a}^a\} \in \mathscr{M}(\mathcal{F})$ the multiset representing the spike train in $\mathcal{S}$ corresponding to neuron $a \in A$ within our model.

Note that, given our choice of $\mathcal{F}^r$ as the class of influence maps, Equation (4) can be computed easily. This motivated us to prefer Equation (3) over alternative definitions of $Supp_n$ that are (arguably) better from a conceptual point of view. For example, one may consider to choose at most one A-event in E^A for each spike time in $\mathcal{S}$ in the computation of $Supp_n$ (e.g. those events which would maximize the support given to the corresponding collection of spike trains), so that no spike contributes to more than one synchronous event.[4]

4 Fuzzy Frequent Pattern Mining

We now move to the task of identifying frequent neuronal patterns, by which we mean sets of neurons in N whose corresponding spike trains in $\mathcal{S}$ have an amount

[4] Some of these alternative characterizations are being considered in current research.

of synchrony or support greater than a certain user-defined frequency threshold. We start by presenting a generalization of the standard frequent pattern mining problem based on what we call fuzzy transactions, which we refer to as the *fuzzy frequent pattern mining* problem—which however differs from other known fuzzy generalizations in the literature that receive the same name, such as e.g. that introduced in [15], based on the consideration of transactions with potentially missing items, or in [16], where fuzzy membership (to a transaction) is assigned to items.

In order to describe and formalize our problem we follow here [17] for standard frequent pattern mining terminology and notation.

Let $\mathcal{I} = \{a_1, ..., a_n\}$ be an item base, for $n \in \mathbb{N}$. A *fuzzy transaction* T (over $\mathcal{I}$) is a pair $\langle id, \eta \rangle$, where id is a unique *transaction identifier* and $\eta : \mathcal{P}(\mathcal{I}) \to [0, 1]$ is a function that assigns *degrees of membership* to all subsets of $\mathcal{I}$ (i.e., the pair $\langle \mathcal{P}(I), \eta \rangle$ is a *fuzzy set*, with η its definitory membership function—see, e.g., [7]).

A transaction $T = \langle id, \eta \rangle$ is said to *support* a set $J \subseteq \mathcal{I}$ to the degree ζ if $\eta(J) = \zeta$. For $\mathcal{D}$ a fuzzy transaction database, the *cover* of J in $\mathcal{D}$ consists of the set of transactions in $\mathcal{D}$ that support J to a degree strictly greater than 0. The *support* of J in $\mathcal{D}$ is given by a function of the membership degrees assigned to J by the transactions in the cover of J that is anti-monotone on the partially ordered set $\mathcal{P}(\mathcal{I})$ with respect to set inclusion.

A subset $J \subseteq \mathcal{I}$ is called *frequent* in $\mathcal{D}$ if its support in $\mathcal{D}$ is greater than some user-specified *minimum support* $\sigma_{\min}$. Our problem thus consists in finding the collection of all frequent subsets in $\mathcal{I}$, which we denote by $F(\mathcal{D}, \sigma_{\min})$.

Note that standard frequent pattern mining can be formalized in terms of fuzzy transactions by means of membership functions that assign value 1 to all subsets of, say, J—for some $J \subseteq \mathcal{I}$—and value 0 to all other subsets of $\mathcal{I}$ (which actually allows us to dispense with η altogether and simply specify the set J). The corresponding support would, in this case, be given by $\sum_{\langle id, \eta \rangle} \eta(J)$.

4.1 Frequent Neuronal Patterns

In our context, the set of neurons N plays the role of $\mathcal{I}$, the item base. In general, fuzzy transactions are formally obtained as follows: each point in time $t \in (0, T]$ gives rise to a transaction $\langle t, \eta_t \rangle$ (and thus we work with a continuous, i.e., uncountable database), where η_t is given by a function on the values taken at t by the influence maps representing spikes in $\mathcal{S}$.

In particular, in relation to our model based on a continuous characterization of synchrony given in Section 3, we have

$$\eta_t(A) = \min_{a \in A}(\max_{f \in G_a} f(t)) \qquad \text{and} \qquad Supp_{|A|}(E^A) = \int_{-\infty}^{\infty} \eta_t(A) dt,$$

for A and G_a as in Equation (4).

Frequent neuronal patterns in N will be those subsets $A \subseteq N$ whose corresponding spike trains (in $\mathcal{S}$) show an amount of synchrony greater than a certain support threshold $\sigma_{\min}$ (i.e., those subsets $A \subseteq N$ with $Supp_{|A|}(E^A) > \sigma_{\min}$).

4.2 Algorithms for Fuzzy Frequent Pattern Mining

In general, the main difference compared to standard frequent pattern mining algorithms (see, e.g., [17]) consists in how the support for sets $A \subseteq N$ is calculated. For the bin-based model of synchrony, the assessment of the support for sets $A \subseteq N$ is as in standard frequent pattern mining. Our alternative graded model of synchrony, based on the identification of spikes with influence maps in the class $\mathcal{F}^r$, favours algorithms suitable for a *vertical layout* of the database, similar in spirit to the well-known Eclat algorithm. In particular, the collection of sequences $\{f^{a_1}, ..., f^{a_k}\} \subset \mathcal{F}^r$ corresponding to the spike times for neuron a, for each $a \in N$, would constitute our primary database. In fact, due to the shape of influence maps in $\mathcal{F}^r$, the sequence of the corresponding influence regions suffices. Thus $Supp_{|A|}(E^A)$—or, equivalently, $Supp_{|A|}(E^A_{Sync^+})$—can be computed as follows: in a first step we form the union of all influence regions corresponding to neuron $a \in A$, for all $a \in A$ (this corresponds to taking the maximum of the influence maps over $(0, T]$ and finding the intervals in which it is not 0). In a second step, we intersect these unions in order to obtain the time intervals with $\eta_t(A) > 0$ (this corresponds to taking the minimum of the maxima of the influence maps and finding the region in which the result is not 0). Finally $Supp_{|A|}(E^A_{Sync^+})$ is computed by summing the lengths of the time intervals obtained and dividing by r.

5 Evaluation and Results

We provide some results[5] concerning a first evaluation of the two models presented in Section 3 on the identification of synchronous patterns from $\mathcal{S}$-like samples: in our settings, frequent patterns $A \subseteq N$ (with respect to some threshold $\sigma_{\min}$) whose support in $\mathcal{S}$ is significantly greater than that expected by chance under the assumption of independence in the spiking activity of neurons in A (found by means of a statistical significance test on the set of frequent neuronal patterns in N).[6] Synchronous patterns are the object of the *unitary events* analysis methodology, based on time-bin discretization (see [10]).

The results provided here aim at emphasizing some of the drawbacks of the bin-based model of synchrony for tasks such as the aforementioned, thus motivating alternative characterizations of synchrony within our framework, like the one introduced earlier in Section 3. For our evaluation we employed an Eclat-based routine in keeping with the indications given in Section 4.2—for the graded model of synchrony.

[5] A full account of our evaluation methodology and results—mostly out of the scope of this paper—will be reported elsewhere, in the form of a journal publication.

[6] The statistical significance test is often redundant for this task. In practice, it is possible to determine a minimum support threshold and minimum pattern cardinality beyond which statistical significance of any neuronal pattern is granted. Research in this direction is being carried out at present.

We generated two types of $\mathcal{S}$-like samples: samples of independent spike trains and of correlated spike trains.

Independent Spike Trains. Spike trains generated as Poisson point processes with properties similar to those of some experimental data. Two types of processes with typical average rates were considered: stationary and non-stationary (based on a phasic-tonic rate responses—see, e.g., [18]).

A small number of neurons was considered ($|N| = 10$) and the time duration was set to 1000 milliseconds (i.e., $T = 1000$) for all generated spike trains. Two data sets, each with 1000 trials for each neuron, were generated: one data set with independent stationary Poisson processes and the other with non-stationary Poisson processes.

Correlated Spike Trains. In order to generate data sets with varying amounts of synchrony among spike trains we essentially adopted the basic features of the SIP (Single Interaction Process) model—described in [19]—along with some modifications aimed at generating non-exact spike coincidences.

We generated two data sets like the ones above to represent the background activity of neurons in N. The size of spike coincidences was set to 10 (i.e., $|N|$). In order to determine the time and number of such coincidences a random choice of $n \in \{3, 4, 5, 6, 7\}$ points in the interval $[0, T]$ was considered for each trial. Each spike time generated this way was added to the background spiking activity of neurons in N by taking into account some random time deviation in order to produce *non-exact* spike coincidences, modeled by means of a uniform random variable on the interval $[\frac{-1}{2}, \frac{1}{2}]$.

The assessment of significant synchrony was done by manipulation of our original trials (i.e., generation of surrugate data). The method employed is *trial shuffling*, a well-established method for destroying spike synchrony across instances of possibly non-stationary point processes (see, e.g., [20]). From the surrogate trials obtained this way we assess the critical amounts of synchrony corresponding to each neuronal pattern $A \subseteq N$ beyond which, given a significance level of 0.01, we declare synchrony to be significant.

The two models presented in Section 3 were tested for influence-region and time-bin lengths $r, d \in \{1, 2, 3, 4\}$. Values for the minimum support $\sigma_{\min}$ were taken from the set $\{0, 1\}$.

Our tests made some of the problems of the bin-based model evident. In particular, results on correlated spike trains show the negative effects of the boundary problem in the characterization of synchrony based on time-bin discretization. Even though (except for $d = 4$ in neuronal patterns of cardinality three) the critical amounts of synchrony estimated from surrogate data for every neuronal pattern $A \subseteq N$ of cardinality at least three were all smaller than three (recall that the number of injected coincidences in trials is at least three), the number of undetected joint-spike coincidences was in general high, even for reasonably large time bins (for $d = 3$ and even $d = 4$).

With respect to the two models, amounts of synchrony strictly greater than 0 in neuronal patterns beyond a certain cardinality are rare and thus, effectively, the significance level goes often far below 0.01 (since the critical amount of

synchrony may be 0 and the condition for significance in our testing procedure was chosen to be an amount of synchrony *strictly greater than* the critical one). In the bin-based model this problem is also observed in relation to other amounts of synchrony. For example, the critical amount of synchrony (with respect to $d = 1$) corresponding to 4-neuron patterns is 1, being an amount of synchrony greater than or equal to 2 for these patterns very rare. Therefore, since we require an amount of synchrony strictly greater than the corresponding critical amount for a neuronal pattern to be considered synchronous, we will have as a result a very small number of synchronous patterns of this cardinality in these trials. Overall, discretization of the sample space in the bin-based approach greatly undermines our testing procedure.

6 Conclusion and Future Work

We have presented a general, flexible framework for the characterization and quantification of spike and spike-train synchrony—based on some general, desirable criteria such notions should meet—able to accommodate (possibly) graded notions of synchrony, like the one presented in Section 3 as an alternative to the bin-based characterization, aimed at overcoming some of its main drawbacks.

Motivated by the task of identifying frequent (and synchronous) neuronal patterns from parallel spike trains, we formalized the so-called fuzzy frequent pattern mining problem—a generalization of standard frequent pattern mining.

Alternative synchrony models are currently being tested on the basis of methodological and/or conceptual improvements. In particular, as pointed out earlier, quantification of spike-train synchrony based on the choice of at most one joint-spike event per spike in the computation of the amount of synchrony (i.e., $Supp_n$) is being considered.

References

1. Abeles, M.: Local Cortical Circuits: An Electrophysiological Study. Springer, Heidelberg (1982)
2. Hebb, D.O.: The Organization of Behavior. J. Wiley & Sons, New York (1949)
3. Abeles, M.: Role of the cortical neuron: integrator or coincidence detector? Israel Journal of Medical Science 18, 83–92 (1982)
4. Goedeke, S., Diesmann, M.: The mechanism of synchronization in feed-forward neuronal networks. New Journal of Physics 10, 015007 (2008)
5. König, P., Engel, A.K., Singer, W.: Integrator or coincidence detector? The role of the cortical neuron revisited. Trends in Neuroscience 19, 130–137 (1996)
6. Buzsáki, G.: Large-scale recording of neuronal ensembles. Nature Neuroscience 7, 446–461 (2004)
7. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
8. Gerstein, G.: Gravitational clustering. In: Analysis of Parallel Spike Trains, pp. 157–172. Springer, Heidelberg (2010)
9. Grün, S., Diesmann, M., Grammont, F., Riehle, A., Aertsen, A.: Detecting unitary events without discretization of time. Journal of Neuroscience Methods 93, 67–79 (1999)

10. Grün, S., Diesmann, M., Aertsen, A.: Unitary event analysis. In: Analysis of Parallel Spike Trains, pp. 191–218. Springer, Heidelberg (2010)
11. Feldt, S., Waddell, J., Hetrick, V.L., Berke, J.D., Zochowski, M.: A functional clustering algorithm for the analysis of dynamic network data. Physical Review E: Statistical, Nonlinear and Soft Matter Physics 79 (2009)
12. Gerstein, G.L., Perkel, D.H., Subramanian, K.N.: Identification of functionally related neural assemblies. Brain Research 140, 43–62 (1978)
13. Syropoulos, A.: Mathematics of multisets. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 347–358. Springer, Heidelberg (2001)
14. Tuckwell, H.C.: Introduction to Theoretical Neurobiology, vols. I and II. Cambridge University Press, Cambridge (1988)
15. Wang, X., Borgelt, C., Kruse, R.: Mining fuzzy frequent item sets. In: Proceedings of the 11th International Fuzzy Systems Association World Congress (IFSA 2005), p. 533. Tsinghua University Press and Springer (2005)
16. Delgado, M., Marín, N., Sánchez, D., Vila, M.-A.: Fuzzy association rules: general model and applications. IEEE Transactions on Fuzzy Systems 11, 214–225 (2003)
17. Goethals, B.: Frequent set mining. In: Data Mining and Knowledge Discovery Handbook, pp. 377–397. Springer, Heidelberg (2005)
18. Louis, S., Borgelt, C., Grün, S.: Generation and selection of surrogate methods for correlation analysis. In: Analysis of Parallel Spike Trains, pp. 359–382 (2010)
19. Kuhn, A., Aertsen, A., Rotter, S.: Higher-order statistics of input ensembles and the response of simple model neurons. Neural Computation 15, 67–101 (2003)
20. Grün, S.: Data-driven significance estimation of precise spike correlation. Journal of Neurophysiology 101, 1126–1140 (2009)

Mass Scale Modeling and Simulation of the Air-Interface Load in 3G Radio Access Networks

Dejan Radosavljevik[1], Peter van der Putten[1], and Kim Kyllesbech Larsen[2]

[1] LIACS, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
{dradosav,putten}@liacs.nl
[2] Deutsche Telecom AG, Landgrabenweg 151, D-53227 Bonn, Germany
kim.larsen@telekom.de

Abstract. This paper outlines the approach developed together with the Radio Network Strategy & Design Department of a large telecom operator in order to forecast the Air-Interface load in their 3G network, which is used for planning network upgrades and budgeting purposes. It is based on large scale intelligent data analysis and modeling at the level of thousands of individual radio cells resulting in 100,000 models. It has been embedded into a scenario simulation framework that is used by end users not experienced in data mining for studying and simulating the behavior of this complex networked system.

Keywords: Mobile Network, Air-Interface Load, Linear Regression.

1 Introduction

This paper reports on a deployed data mining application that has been developed by one of the largest European telecom operators and has been in continuous use ever since. In order to accommodate the continuing strong increase of mobile internet traffic, the operator's Radio Network Department has to continuously monitor and upgrade the 3G Radio Access Network. This requires an Air-interface Load forecast for every radio cell in the network. However, such a detailed forecast was not readily available. Furthermore, there is a need to simulate different scenarios for different parts of the network. Given the complexity of the problem, the dimension of the network and the repetitiveness of the task, a manual approach is out of the question.

In this paper we present a fully automated approach that generates multi-variate linear regression models on a grand scale, using primarily open source tools. The key business value of this research is that it solves a very complex and high impact business problem that cannot be approached by using simpler planning approaches. The exact return is confidential, but cellular network infrastructure forms a major part of an operator investment budget, and this is a key system for tactical and strategic network investment decisions. In the group where this operating company belongs, up to 50% of wireless CAPEX investments are going into the radio access. For reference, operators invest more than 20 billion USD into cellular network infrastructure worldwide. Our methodology is first and foremost intended to ensure that capacity is added in time and at the right place, thus avoiding inefficient investments and poor customer

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 301–312, 2012.
© Springer-Verlag Berlin Heidelberg 2012

experience due to traffic congestion, which can ultimately lead to churn. The system has been rolled out to full production use. None of the other operator companies in the telecommunications group uses a similar fine grained approach.

Whilst the core intelligent data analysis algorithms used are not novel, we apply these on a large scale by modeling individual radio cells across a variety of dimensions (section 3 motivates why we model at cell level). This has also been embedded into a simulation framework targeted at non-data miners in tools they are familiar with to enable them to run low level simulation scenarios. Hence, the goal is to provide a case example of an embedded, deployed intelligent data analysis system, dealing with real world aspects such as scale and having major business impact.

As discussed, the technical novelty is not determined by the complexity of the base estimators used. We use simple linear regression models as data inspection has shown that the behavior to be predicted is primarily linear, and experiments confirmed that complex algorithms actually performed worse given the high variance associated with these models. This is not uncommon in real world data mining problems [1]. What makes this problem out of the ordinary is the massive number of models. For each of the 20,000 cells in the network we create five models to predict different kinds of outcomes, resulting in a total of 100,000 models. Model parameters are estimated using ten-fold cross validation, which increases the number of models estimated to over 1 million. This process is repeated on a regular basis, given that the customer base and its behavior, as well as the cellular network itself change constantly. Finally, we do not just deploy the forecasted loads. The underlying regression formulas are provided by the data miners to the end user analysts as simple spreadsheets, which enables them to tune various simulation and forecasting scenarios without further involvement from the data miners. We think this approach can easily be replicated and applied to problems from other industries which require similar predictive models and simulation of networked systems on a large scale, such as for instance sensor networks, retail outlet planning and supply chain logistics.

The rest of the paper is structured as follows. Section 2 describes the load parameters. Section 3 discusses the complex nature of network load and how to approximate it, including a motivation for modeling at the granular cell level. Section 4 describes the construction of the load formulas and the forecasting of the future load of the network using simulation based on these formulas. Limitations and future work are discussed in section 5. Finally, we present our conclusions in section 6.

2 Defining the Air-Interface Load Parameters

The communication between a network cell and a mobile device is separated into downlink communication- directed from the cell to the mobile device and uplink communication- directed from the mobile device to the cell. Therefore, the Air-interface load for a cell consists of the Downlink Load (DL) and Uplink Load (UL). When measuring the actual Air-Interface load typically only the maximum of the UL and DL values is taken. Multiple measures of both DL and UL can be devised. A cell is considered to be in overload if the load is above a certain threshold. When a cell is

in overload, it cannot serve additional customers that demand its resources. Obviously, all cells in overload require an adequate upgrade.

Most of the literature on telecom network is related to network optimization or load control rather than load prediction [2, 3, 4, 5]. Therefore, there was no previous knowledge on which of the measures of either Downlink or Uplink Load would be possible to predict with the least error, so several measurements of these were chosen.

We used a number of measures to characterize uplink load. Firstly, the *Count of RAB (Radio Access Bearer) Releases Due To Interference* was chosen; a RAB is a cell resource which is necessary to be assigned to the mobile subscriber/device in order for any voice/data transaction to be possible. Normally, it is released after it is no longer necessary, unless there were circumstances (e.g. interference from other users or cells) which caused it to be dropped [2]. Secondly, we used the *Average Noise Rise (ANR)*, measured per hour in dBm (Decibels per milliwatt), which is the difference between the Uplink power received in a given time when a number of users consume cell resources, and the Uplink power of the same cell when it is not serving any users at all [3]. Thirdly, we chose the *Average Noise Rise on Channels Dedicated to Release 99 Capable Devices* (refers to lower data transfer speed up to 384 Kbps). Two additional uplink measures were considered: *Count of RAB (Radio Access Bearer) Setup Failures* and *Count of RRC (Radio Resource Control) Setup Failures*. These measurements were discarded at later stages of the process due to the very low number of models that could be generated because of too many zero-values.

The parameters used as measures for downlink load were the following. Firstly, we used *Percentage of Consumed Downlink Power (CDP),* similar to Downlink Noise Rise [4]. Downlink power is a finite cell resource and it amounts to 20W. Each mobile device/user gets a portion of this, which is proportional to the bandwidth they require. In an overload situation there is no more power to be distributed. The other downlink load measure was the *Count of "No Code Available" Situations* (NCA) [5]. Each cell has 256 codes that can be assigned to a mobile device for a voice call or a data session. The higher the downlink bandwidth required, the higher the number of codes will be assigned. After all the codes have been assigned, the next devices that requests a code from the cell, gets a "no code available" message and cannot use the cell resources.

As input parameters we used different measures from the Nokia Data Warehouse [6], a tool that is used in telecom operators to monitor Radio Network Performance. Even though we included input parameters related to voice services, most of the input parameters are related to consumption of Data Services, because they require more of the cell resources. These include the following: Average Voice Call Users, Average Release 99 Uplink users, Average Release 99 Downlink users, Average High Speed Uplink Packet Access (HSUPA) users, Average High Speed Downlink Packet Access (HSDPA) users, Maximum HSUPA users, Maximum HSDPA users, Total RRC attempts, Total Active RABs, Total Voice Call RAB Attempts, Total Data Session RAB Attempts, Average Downlink Throughput, Average Uplink Throughput, Average Soft Handover Overhead Area (measures the intersection of coverage of the particular cell with other cells), Average Proportion of Voice Traffic originated in that cell (as opposed to traffic originated in other cells and handed over to that cell),

Average Proportion of Data Traffic originated in that cell. Forecasts for future values of the input parameters were available at the operator.

Both the input and the output parameters were taken on per cell per hour level.

3 Approximating and Predicting the Air-Interface Load

Most of the literature on load forecasting is related to electrical networks. A good overview is presented in [7]. Various methods have been deployed for this purpose: regression models, time series, neural networks, expert systems, fuzzy logic etc. The authors state a need for load forecasts for sub-areas (load pockets) in cases when the input parameters are substantially different from the average, which is a case similar to different cells in a mobile telecom network.

Related to mobile telecommunications, data traffic load (which is different than air interface load) focusing on a highly aggregated link has been forecasted in [8], comparing time series (moving averages and dynamic harmonic regression) with linear and exponential regression. Also, Support Vector Regression was used by [9] for link load prediction in fixed line telecommunications.

In order to forecast the future load for each cell in the network, it is necessary to understand the relationship between the input parameters (causing the load situation) and the current load. The input parameters in case of the Air Interface load are all parameters which can be made accountable for the load situation in the cell (Section 2). Therefore, the load parameter (output) can be expressed as $L=f(x_1, x_2,\ldots, x_n)$. Ideally, the load of each cell x in a given time could be expressed as the sum of all users consuming resources of that cell at the particular time multiplied by the amount of resources they use plus the interference between that cell and all the other cells in the network (in practice limited to the neighboring cells):

$$\mathrm{L(x)} = \sum_{i=0}^{m} \sum_{j=1}^{n} User_i * Resource_j + \sum_{y=1}^{z} interference(x,y) \quad (1)$$

where m is the count of users that are using the resources of cell x, n is the count of resources of the cell x, z is count of all cells in the network and *interference(x,y)* is the interference measured between cell x and y.

Unfortunately, there was no tool that would provide such a detailed overview. In order to approximate the load function, we recorded the five different load parameters (outputs) and 16 input parameters described in section 2, on an hourly basis during 6 weeks. This provided approximately 1,000 instances for each cell to build a predictive model, or 20,000,000 instances in total.

One of the choices to be made was whether a distinct formula for every cell shall be built or – alternatively – a common formula valid for all cells should be used. The approach where a model is created for each cell was chosen, due to the network experts' conviction that each cell is different, and a unified approach simply would not work, because some of the parameters influencing the load of each cell were immeasurable and unpredictable.

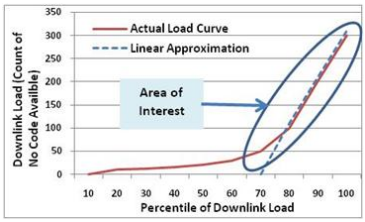


Fig. 1. Actual Downlink Load vs. Linear approximation

Next, the domain experts were intrinsically interested in being able to model cells that actually do not behave like other cells, especially when these are highly loaded. Furthermore, there would be a challenge in normalizing with respect to the varying capacity of the cells, i.e. what where the cell sized to handle. Finally, we hypothesized that not just model parameters could differ by cell, but also the optimal selection of features, similar to the load pockets explained by [7].

The choice of linear regression [10] was made due to several reasons. First of all, even though the distribution of the values of each of the load measures we are trying to predict vary between close to linear and close to exponential, we are only interested in the higher values of the load curve, and this can be approximated quite well with linear regression, as shown on Figure 1. For this purpose, before constructing the regression formulas, we remove all zero instances. Furthermore, linear regression is a very fast algorithm compared to other methods, which is very useful when it is necessary to develop a large number of models in a short time. Even though it is imaginable that better results might be achieved by using non-linear regression, regression trees, or other algorithms, this might not be necessary (Figure 1). Also, simple low variance methods such as linear regression frequently perform much better in practice than more complicated algorithms, which can very often over fit the data (e.g. high variance algorithms such as neural networks). In other words, in real world problems variance is typically a more important problem than bias when it comes to data preparation and algorithm selection [1]. Trials on a smaller sample were already made with regression trees, but apart from the visibly increased time consumption, the accuracy did not improve. On the contrary, in some instances it decreased.

Last but not least, linear regression is easy to implement, easy to explain and its results and models are easy to export for other use. Exporting the models to Excel was of crucial value, as analysts would use them in order to predict the future load of each cell, by scaling the input parameters, based on internal forecasting models. In other words, this allows non data miners to simulate future network load based on changes in the various type of network traffic, using simple tools they are familiar with.

4 Approach Description and Results

In this section we will describe how the models are being generated and put to work. This includes the tools that were used, a detailed description of the approach, the results of this mass modeling process and the process of forecasting the future load.

4.1 Tools

The tools used in this research are either open source, or can be found in the IT portfolio of any telecom operator. These are the following:

Nokia Data Warehouse [6] was used for data collection for both the input and the output parameters. This software tool was already a part of the Network/IT infrastructure of the operator. It contains technical parameters related to the mobile network performance. The most important feature of this tool for our research was that it contained hourly aggregates of all the input and output parameters we used in our research (Section 2). Obviously, any other tool that collects data about network performance could have been used. This is the only domain specific tool from our process.

Load Prediction and Simulation Data Mart. This is an Oracle Database 10g- 64 bit v10.2.0.5.0 [11] used for all our task specific data preparation and manipulation**.**

Due to the fact that the necessary input and output parameters were stored at different tables in the Nokia Data Warehouse, we needed a separate database where we could manipulate the data easier (e.g. merge tables, create indexes, and build the final flat table). This reduced the duration of the data collection and data preparation process from two weeks to 1 day by productizing data collection. Because we are rebuilding and rescoring models on a continuous and automated basis, this was a key improvement. Any other database platform (commercial or open source) could have been used. We opted for Oracle based on license availability.

WEKA 3.6.4 x64, an open source data mining platform [12], was used for building the linear regression formulas and validating them. Of course, any other tool capable of deriving linear regression can also be used for this purpose. That said, this shows that even a research focused open source tool like WEKA can be used in critical commercial settings, at high complexity (20.000 cells, 5 models each, around 1000 instances each).

Strawberry Perl for Windows v5.12.3 [13] is an open source scripting language that we used in order to create the script that is the core of this approach. Our script creates WEKA input files by querying the Oracle database, generates the regression models by executing calls to WEKA, and stores the regression formulas and the cross-validation outputs (Correlation Coefficient, Mean Absolute Error, Root Mean Squared Error, Relative Absolute Error, Root Relative Squared Error, and Total Number of Instances used to build the model) in csv files.

MS Excel 2010 [14] – part of MS Office 2010, was used to predict the future load of cells, using the regression formulas created by WEKA and extrapolations of the input values using a internet traffic model scaling factors based on handset/internet usage developments (internal to the operator).

4.2 Process Description

A graph of how our approach uses these tools to derive and store the regression models is presented on Figure 2. In the core of our approach is a Perl [15] script that automated the derivation of the regression formulas for each cell. This script executed calls to WEKA and queries to the Oracle Database. It works in the following manner:

```
Get list of cells from the database;
for each cell
  Run a query on the database to isolate only the data
  related to that cell (all the input and the 5 output
  parameters);
  Make separate files for each of the 5 load parameters;
  For each of the 5 load parameters
Filter out all instances where the load is 0;
    Select only the relevant variables for the regression
    formula of that cell, using a wrapper approach;
    Build the linear regression model and store it;
    Validate the model- Use 10-fold cross-validation;
    Store the formula, the number of instances used to
    build it, the correlation between the predicted and
    actual value for load, the Mean Absolute Error (MAE)
    and the Root Mean Square Error (RMSE) as reported
    from the cross-validation;
```

While generating the models/regression formulae, we used a wrapper [16] approach. Wrapper approaches automatically select the best variables for predicting the outcome, taking into account the algorithm to be used, which in our case is linear regression. They do not necessarily perform better or worse than filter approaches [17]. Our motivation to use the wrapper approach was to avoid human interaction with the model building process as much as possible, which obviously makes the process much faster.

It is worthwhile mentioning that the optimal feature and linear regression model selection were performed using 10-fold cross validation [3]. This was done in order to balance between cells with large sample of non-zero instances and cells with a smaller sample. The reported correlation coefficient, MAE and RMSE are averages from the 10 repetitions. Using 10-fold cross validation already provides a good estimate of the accuracies of these formulas. Of course, we intend to test them on a completely new dataset, not only to confirm the accuracies achieved, but also to find out when is a good time to update the model. We expect that updates should be necessary every few months, because of the reconfiguration of the network, additions of new cells and upgrades to the existing ones.

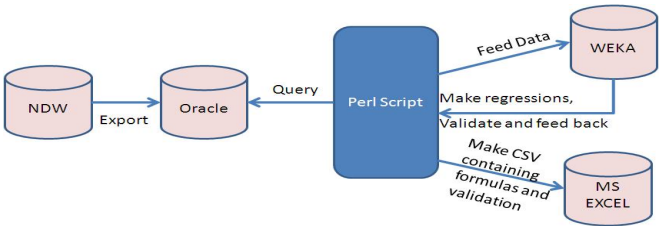


Fig. 2. Communication Graph of the Tools used

4.3 Results of the Modeling Process

Using this process we were able to run 100,000 (5 outputs for 20,000 cells) regressions in less than 1 week, by just one click. This did not result in 100,000 models, because in some cases it was impossible to derive a formula due to the large number of instances that were filtered out for zero load. But, in order to measure the load of a cell, it is sufficient that a model is generated for at least one output variable. Cases of cells where it was not possible to generate a model for any of the variables were rare. Furthermore, cells that do not show any load by the means of the five output variables, are not of interest for our problem situation. For practical purposes, we will only present the modeling results for two of the five output variables we used to describe the air interface load in section 2. We chose one Uplink Load measure- Count of RAB Releases due to Interference-RRI, presented in Table 1, and one Downlink Load measure- Count of No Code Available- NCA, presented in Table 2.

Both Table 1 and 2 have the same structure. In the first column Bands of Averages for the respective output variables RRI and NCA are given. The second column contains the count of cells that falls into this band. The third column presents the average count of non-zero instance (NZI) in each band.

Table 1. Regression Modeling Results for Count of RAB Releases due to Interference (RRI)

Count of RAB Releases due to Interference (RRI)	Count of Cells	Avg Count of NZI	Avg nonzero RRI	Avg CC	Models Built Vs Number of Cells
RRI<=1	8373				
1<RRI<2	7344	89.4	1.3	0.141	0.582
2<=RRI<3	1359	229.4	2.4	0.545	0.769
3<=RRI<5	972	296.2	3.8	0.658	0.829
5<=RRI<10	780	365.0	7.0	0.751	0.881
10<=RRI<20	503	407.6	14.0	0.810	0.905
RRI>=20	538	431.3	56.8	0.873	0.920

Table 2. Regression Modeling Results for Count of No Code Available (NCA)

Count of No Code Available (NCA)	Count of Cells	Avg Count of NZI	Avg nonzero NCA	Avg CC	Models Built Vs Number of Cells
NCA<=1	682				
1<NCA<2	792	130.5	1.6	0.454	0.775
2<=NCA<5	2229	331.5	3.3	0.635	0.971
5<=NCA<10	2321	500.5	7.3	0.773	0.994
10<=NCA<20	3420	597.2	14.7	0.836	0.994
20<=NCA<30	2706	681.3	24.9	0.862	0.994
30<=NCA<50	3858	732.0	39.0	0.861	0.998
50<=NCA<100	3063	758.1	67.8	0.872	0.996
NCA>=100	798	760.2	208.7	0.790	0.992

In other words, it presents the number of instances used to build the regression, because we only took non-zero output values into account. The fourth column, perhaps redundant, presents the average of the variable in that band. The fifth column presents the average Correlation Coefficient (CC) between the predicted and actual variables in the particular band. These Correlation Coefficients are the result of the 10-fold cross validation. The last column presents the ratio between the number of formulas that were generated and the total count of cells in each band. Namely, for certain cells it was not possible to build the regression because of a very low number of non-zero instances.

The results can also be evaluated by using two criteria: The Correlation Coefficient and The Ratio of The Models Built (the last two columns in Tables 1 and 2). Obviously, the Correlation Coefficient in both these cases (RRI and NCA) grows alongside the number of instances, which is to be expected. But, because of the choice we made at the beginning of the research, to focus on the higher levels of load and eliminate the zero values, the Correlation Coefficient between actual and predicted values also grows as the output variable value is higher. The case is similar with the Ratio of the Number of Formulas built. The difference is more evident in the case of RRI (Table 1), because this is an event that occurs less frequently than NCA. The number of formulas here is also lower, especially in cases with low RRI levels. But, as mentioned before, we are not interested in these cases.

4.4 Forecasting Future Load

Once the load formulas have been derived it is possible to forecast the future load situation if the changes in the describing parameters are known. These changes of the input parameters are described by means of scaling factors. The scaling factors are calculated by using a traffic forecast model developed by the operator (out of scope of this paper).

This is done in the following way:

```
For each output variable
  For each cell
    Select the top 100 instances of the output variable
    and its corresponding values for the input varia-
    ble;
    Make averages of these input variables;
    Scale the input variables up or down, according to
    scaling factors developed by a traffic model;
    Feed the scaled values of the input parameters into
    the regression formula;
    If resulting value > critical threshold then cell
    should be upgraded;
```

This part of the process is performed in a tool as simple as MS Excel. This was a key driver for the business success of the solution. In our experience the importance of the Deployment step in the data mining process is generally underestimated. By

providing not just the scores but also the underlying models in a format and tool that was immediately usable and tunable to end users who are not data miners, the solution was readily accepted and also used in new ways not necessarily intended by the data miners, for instance detailed simulation scenarios.

We mentioned in Section 2 that the Air interface load is calculated as a maximum of Uplink Load and Downlink Load. This means that a cell will be upgraded if it is in overload on either the Uplink or the Downlink. In terms of our approach, a cell is upgraded if any of the five output variables, used as measures of Uplink or Downlink load, are above a critical value.

5 Limitations and Future Work

The regression formulas developed by this approach can be used on a long term basis only if the mobile network stays the same (is frozen) over a longer period. But, this is not the case. The cellular network is a system of very complex dynamics. The many changes that occur, such as hardware and software updates, network reconfigurations and optimizations, as well as network upgrades and roll-out of new cells cannot be taken into account in advance. It is necessary to collect a new dataset and rebuild the regression formulas, in order to incorporate all these changes into the model. This is why the process described in this paper is scheduled for execution every 3-4 months.

Further evaluation of the quality of the derived load formulas of course also involves the comparison of the predicted load with the actually measured load in the future. It should however be noted that there a lot of factors impeding a direct comparison. As noted above, all changes to the settings of a cell within the forecasting timeframe affect the load formula, which means that after such changes the derived formula is - at least to some degree - no longer correct. For this reason it will be challenging to really quantify the accuracy of the predictive model. Developing a fair method of evaluation, which would incorporate the network changes, would be beneficial. In terms of the core algorithms, we do want to keep the benefit of using a simple, fast and robust low variance approach such as linear regression.

However, we do plan to explore a methodology that would allow us to combine a global network model with local models for each cell, for instance multitask or transfer learning [18]. In principle, we have almost infinite data available for most cells, so local models cannot be improved by a global model. Nevertheless, there could be exception for a non select small number of cells. Last but not least, a clustering approach could be devised to group cells with similar formulas or levels of load, thereby generating new knowledge for the telecom domain experts.

6 Conclusions

In this paper we presented a very simple yet effective approach of applying data mining in commercial surroundings. Unfortunately, data mining is still seen as a black box in many industries, telecom not excluded. Even though some data mining activities are taken, typically in the Marketing/Customer Retention field, there is a myriad

of other possibilities in business where data mining can be applied. In our opinion, it is better to start with simple methods, such as regression, because it is easier to understand them. Once these simple approaches gain acceptance, and familiarize the industries with data mining, opportunities to apply more advanced techniques will arise.

In our result section we show that it is easier to accomplish a target, if one is focused on it. Namely, with our approach we wanted to target cells where some load (non-zero load) occurs, in order to predict the part that really matters more correctly: the high end part of the load curve (the cells in overload). In other words, as the network load grows, so does the quality of the model's predictions. We willingly sacrificed the models' performances within the lower loaded cells, because they are of no interest.

Next, one of the key values of the approach is that a large number of regression models (close to 100,000) are developed in a very short period of time with minimum human interaction. In order to do this, we deployed a simple algorithm such as linear regression, motivated by its speed and other benefits explained earlier, a wrapper feature selection, in order to avoid human interaction, and 10-fold cross validation which makes the models statistically sound. Manually, this task would be impossible. Obviously, the possibility to generate these formulas was crucial to the operator.

Another large benefit of our approach is that after the models have been generated, they are exported into Excel sheets, which allows a team of radio network analysts, which are not data miners, to use these formulas for forecasting the future network load. This allows them to simulate multiple traffic scenarios by scaling the current input parameters. These scenarios include evaluations of network investments necessary to accommodate localized user growth due to targeted marketing campaigns or more extreme, adding a new wholesale client- or an MVNO (Mobile Virtual Network Operator).

Typically, planning network upgrades is a reactive process. Our approach makes it proactive, which was acknowledged by the operator, who has fully integrated our approach into its network upgrade planning and budgeting activities. Of course, due to the fast pace network changes, the formulas need to be upgraded every 3-4 months, but this is also scheduled as a part of a standard process. Due to confidentiality, we cannot disclose the exact return of this project, but given that the network is the key resource of an operator, the investments into its upgrades are quite sizeable. To our knowledge, this is the first time a telecom operator has applied data mining in order to create a proactive network upgrade management process. This allows the operator to manage network performance better and avoid extreme congestion situations, which can result in degraded customer experience and loss of reputation for the operator. As mentioned at the beginning, the research was performed at a large telecom operator with branches in many European countries. At the moment, our research is deployed in only one of the countries where this operator is present, but efforts are made to replicate it in the other branches as well.

Perhaps one of the most interesting aspects of this approach is the extremely low cost. Given that we used the existing IT infrastructure (Server, Nokia Data Warehouse, Oracle, Excel) combined with open source tools (WEKA, Perl), the only cost that incurred are the 1 week Processing Time Cost (of the Server) and the labor cost of the employees in this project. Also, the Oracle Database that we used can be

replaced with a less expensive or free database alternative in order to further reduce the cost, in case the potential user of our approach does not have an Oracle License.

Last but not least, we would like to point out the possibility of applying our research onto domains other than telecom. This approach would be applicable to any other industry where large scale regression models are necessary. This can be accomplished simply be replacing the data source, in this case Nokia Data Warehouse, with a data source suitable for the industry that would like to apply our research.

References

1. van der Putten, P., van Someren, M.: A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. Machine Learning 57(1-2), 177–195 (2004)
2. Yates, R.: A framework for uplink power control in cellular radio systems. IEEE JSAC 13(7), 3141–3147 (1995)
3. Geijer Lundin, E., Gunnarsson, F., Gustafsson, F.: Uplink load estimation in WCDMA. In: Proc. IEEE Wireless Communications and Networking Conference (2003)
4. Muckenheim, J., Bernhard, U.: A Framework for Load Control in 3rd Generation CDMA Networks. In: Proc. of the IEEE Global Telecommunications Conference, vol. 6, pp. 3738–3742 (2001)
5. Natalizio, E., Marano, S., Molinaro, A.: Packet scheduling algorithms for providing QoS on UMTS downlink shared channels. In: IEEE VTC, vol. 4, pp. 2597–2601 (2005)
6. Nokia Siemens Networks: Nokia Siemens Networks WCDMA RAN, Rel. RU10- System Library, v.1: RNC Counters – RNW Part. Nokia Siemens Networks. Proprietary and Confidential (2008)
7. Feinberg, E.A., Genethliou, D.: Load Forecasting. In: Chow, J.W., Wu, F.F., Momoh, J. (eds.) Applied Mathematics for Restructured Electric Power Systems, pp. 269–285. Springer, Heidelberg (2005)
8. Svoboda, P., Buerger, M., Rupp, M.: Forecasting of Traffic Load in a Live 3G Packet Switched Core Network. In: Proc. of 6th International Symposium on CNSDSP, pp. 433–437 (2008)
9. Bermolen, P., Rossi, D.: Support vector regression for link load prediction. Computer Networks 53(2), 191–201 (2009)
10. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Technique, 2nd edn. Morgan Kaufmann, San Francisco (2005)
11. Oracle.: Oracle Database Documentation Library, `http://www.oracle.com/pls/db102/homepage`
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
13. Strawberry Perl, `http://strawberryperl.com/`
14. Microsoft Corporation: Microsoft Excel, `http://office.microsoft.com/en-us/excel/`
15. Christiansen, T., Torkington, N.: Perl Cookbook, 2nd edn. O'Reilly, Sebastopol (2003)
16. Kohavi, R., John, G.: Wrappers for feature subset selection. In: Artificial Intelligence 1997, pp. 273–324 (1997)
17. Tsamardinos, I., Aliferis, C.: Towards principled feature selection: Relevancy, filters and wrappers. In: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistic (2003)
18. Caruana, R.: Multitask Learning. Machine Learning 28, 41–75 (1997)

Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data

Jesse Read[2], Albert Bifet[1], Bernhard Pfahringer[1], and Geoff Holmes[1]

[1] University of Waikato
Hamilton, New Zealand
{abifet,bernhard,geoff}@cs.waikato.ac.nz
[2] Universidad Carlos III
Madrid, Spain
jesse@tsc.uc3m.es

Abstract. Many real world problems involve the challenging context of data streams, where classifiers must be incremental: able to learn from a theoretically-infinite stream of examples using limited time and memory, while being able to predict at any point. Two approaches dominate the literature: batch-incremental methods that gather examples in batches to train models; and instance-incremental methods that learn from each example as it arrives. Typically, papers in the literature choose one of these approaches, but provide insufficient evidence or references to justify their choice. We provide a first in-depth analysis comparing both approaches, including how they adapt to concept drift, and an extensive empirical study to compare several different versions of each approach. Our results reveal the respective advantages and disadvantages of the methods, which we discuss in detail.

Keywords: data streams, incremental, dynamic, evolving, on-line.

1 Introduction

The trend towards dynamic data sources is clear, both in the real world and the academic literature. Modern data sources are not only dynamic but generated at high speed in real time. Such contexts can be found in sensor applications, measurements in network monitoring and traffic management, log records or click-streams in web exploration, manufacturing processes, call-detail records, email, blogs, news feeds, and social networks. Real-time analysis of these data streams is becoming a key area of data mining research as the number of applications demanding such processing increases.

A *data stream* environment has different requirements from the traditional batch learning setting. The most significant are the following, as outlined in [1]:

- be ready to predict at any point;
- data may be evolving over time; and
- expect an infinite stream, but process it under finite resources (time and memory).

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 313–323, 2012.
© Springer-Verlag Berlin Heidelberg 2012

It is important to note that in this study we do not tackle two important aspects of this setting. First, we do not consider changes to the input distribution in terms of the addition, deletion or updating of attributes. In a sensor network, for example, a new sensor could be added to or deleted from an existing network, or a new sensor could replace an old one and produce a different range of values. Second, we assume that all instances *can* be labelled. Labels could, in practice, come at a cost. Stream-based methods to tackle these problems are being developed but are beyond the scope of this study.

The approaches to data-stream classification in the literature can generally be considered as being of one of two types: *batch-incremental*, or *instance incremental*.

In the batch-incremental approach, a traditional batch-learning method is trained on batches of the data: every w new examples form a batch, and when that batch is complete, it is given to a learner to train on. The main disadvantages of these methods are that they:

- require a parameter w specifying the batch-size;
- are forced to delete trained models to make room for new ones; and
- cannot learn from the most recent examples until a new batch is full.

Having to delete trained models may affect these methods' ability to learn a complete concept, and not being able to learn from new examples immediately may affect their ability to respond to a new concept.

Instance-incremental methods are truly incremental in the sense that they learn from each training example as it arrives. This category includes lazy learners (like k-Nearest Neighbour, e.g., [2,3]) and incremental learners such as Naive Bayes [4] and Hoeffding Trees [5] that can essentially learn indefinitely. Due to their incremental nature, instance-incremental methods are often chosen over batch-incremental methods, but they also have disadvantages; most notably:

- are fewer in number than batch methods (thus a smaller selection of appropriate methods); and often
- only learn a concept correctly from a huge number of examples.

For example, Hoeffding Trees grow very slowly with respect to the number of instances they observe. Lazy methods such as k-Nearest Neighbour learn more quickly in this respect from far fewer examples, but these methods are limited to a relatively small internal buffer of instances, that they must search through and add to for every new example in the stream. Thus, `kNN` methods must discard information over time like batch-incremental methods do (although only one instance at a time in this case).

Since data streams can be susceptible to concept drift, response to concept drift is an important issue; although detecting drift is a more important issue for instance-incremental methods like Hoeffding Trees. k-Nearest Neighbour and batch-incremental methods adapt to some extent automatically, as they are forced to phase out part of their model over time due to resource limitations, but Hoeffding Trees will simply learn a new concept 'on top' of an old concept and, therefore, often perform better when used with a scheme with an explicit concept-change detector [6].

In the following section we review batch-incremental and instance-incremental methods. Although both kinds of methods have seen improvement over the years, so far no exhaustive comparison study has been published. Instance-incremental papers consistently mention the benefits of learning incrementally, whereas batch-incremental papers consistently mention that batch-learning is adequate for data streams.

In this paper we address this important lack of knowledge. We investigate the relative importance of the advantages and disadvantages of both approaches with a large empirical evaluation involving a variety of concept-drifting data streams and several of the best batch-incremental and instance-incremental approaches from the literature. We present these results as evidence indicating which approaches are better in which circumstances; thus providing crucial information for researchers deciding on an approach to use. It should be noted that such studies have been crucial to the development of classification algorithms in the non-incremental batch setting, both in terms of improving the quality of datasets used for evaluation [7] and for the assessment of overall classifier performance [8].

2 Prior Work

Naive Bayes [4] is a widely known instance-incremental learner; it simply updates internal counters with each new instance and uses these counters to assign a class in a probabilistic fashion to a new item in the stream. Stochastic Gradient Descent [9] is another incremental algorithm which forms the base for many neural network methods.

Naive Bayes provides a baseline for instance-incremental classification, but in terms of performance, has already been superseded by *Hoeffding Trees* [5]: an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams by exploiting the fact that a small sample is often enough to choose an optimal splitting attribute.

Bagging and Boosting ensemble methods can be adapted to the stream setting and do improve the accuracy of base classifier methods. Oza and Russell [10,11] proposed *Online Bagging* which gives each example a weight according to a Poisson distribution. This method has been shown to work well with Hoeffding Trees [12]. A more recent version of Bagging is presented in [13] which obtains better accuracy albeit at the cost of additional use of computational resources.

Batch-incremental methods cannot learn instance-by-instance, but must create models from batches, and at some point remove them as memory fills up. The size of the batch must be chosen to provide a balance between best model accuracy (large batches) and best response to new instances (smaller batches). In this context, it makes sense to use an ensemble, where several models are created from relatively small batches and their predictive power is combined under a voting scheme, such as in [14] and the Accuracy Weighted Ensemble (AWE) [15], where typically, when the maximum number of models is reached (the ensemble size) the oldest model is reset with a model built from the newest batch. In AWE, ensemble members are additionally weighted by their classification performance.

Table 1. The methods we consider. Ensemble iterations are specified by parameter n. Note that Accuracy Weighted Ensemble (`AWE-`) can be used with any classifier; Leveraging Bagging (`LB-`) can be used with any incremental classifier.

Key	Classifier	Parameters
`NB`	Naive Bayes	
`SGD`	Stochastic Gradient Descent	
`HT`	Hoeffding Tree	
`LB-HT`	Leveraging Bagging / `HT`	$n = 10$
`kNN`	k Nearest Neighbour	$w = 1000, k = 10$
`LB-kNN`	Leveraging Bagging / `kNN`	$n = 10$
`AWE-SMO`	AWE of Support Vector Machines	$w = 500, n = 10$
`AWE-J48`	AWE of C4.5 Decision Trees	$w = 500, n = 10$
`AWE-LR`	AWE of Logistic Regression	$w = 500, n = 10$

The k-Nearest Neighbour algorithm, which assigns the most common class of the k most similar examples, has been used in data streams in [2]. This algorithm is naturally suited to this setting because of its instance-incremental nature. Improved searching [3] and instance-compressing techniques [16] have been shown to improve its capacity considerably.

Reacting to concept drift is a fundamental part of learning from data streams. `kNN` and batch-incremental methods inherently phase out data (and thus, old concepts) but instance-incremental models such as Hoeffding Trees need an explicit change detection, or else they will learn a new concept 'on top of' and old concept. `ADWIN` [6] keeps a variable-length window of recently seen items (such as the current classification performance) and signals change when the concept within this window changes, and thus can be coupled with any method that requires explicit change detection, as has been done successfully with Hoeffding Trees in [12].

3 Experimental Setup and Methodology

We compare the performance of the batch-incremental methods (using the recent Accuracy Weighted Ensemble method of [15]) employed with powerful batch classifiers (Support Vector Machines, C4.5 Decision Trees, Logistic Regression) with instance-incremental methods (Naive Bayes, Hoeffding Tree ensembles, Stochastic Gradient Descent, and k Nearest Neighbour variations). These methods, and their parameters, are displayed in Table 1. All methods have been implemented in Java extending the MOA framework for data streams [17]. Any parameters not shown in the table are the default ones set in this framework.

We use the experimental framework for concept drift presented in [1]: considering data streams as data generated from pure distributions, we can model a concept drift event as a weighted combination of two pure distributions that characterizes the target concepts before and after drift. This framework defines the probability that a new instance of the stream belongs to the new concept after the drift based on the sigmoid function.

3.1 Data

In our experiments we use a range of both real and synthetic data sources.

Synthetic data has several advantages–it is easier to reproduce and there is little cost in terms of storage and transmission. We use the data generators most commonly found in the literature.

SEA Concepts Generator. An artificial dataset, introduced in [18], which contains abrupt concept drift. It is generated using three attributes. All attributes have values between 0 and 10. The dataset is divided into four concepts by using different thresholds θ; such that: $f_1 + f_2 \leq \theta$ where f_1 and f_2 are the first two attributes, for $\theta = 9, 8, 7$ and 9.5.

Rotating Hyperplane. The orientation and position of a hyperplane in d-dimensional space is changed to produce concept drift; see [19].

Random RBF Generator. Using a fixed number of centroids of random position, standard deviation, class label and weight. Drift is introduced by moving the centroids with constant speed.

LED Generator. The goal is to predict the digit displayed on a seven-segment LED display, where each attribute has a 10inverted; LED comprises 24 binary attributes, 17 of which are irrelevant; see [20].

We consider three of the largest datasets from the UCI repository [21]:

Forest Covertype. Contains the forest cover type for 30 x 30 meter cells obtained from US Forest Service data. It contains 581,012 instances and 54 attributes. It has been used in, for example, [22,10].

Poker-Hand. 1,000,000 instances represent all possible poker hands. Each card in a hand is described by two attributes: suit and rank. Thus there are 10 attributes describing each hand. The class indicates the value of a hand. We sorted by rank and suit and removed duplicates.

Electricity. Contains 45,312 instances describing electricity demand. A class label identifies the change of the price relative to a moving average of the last 24 hours. It was described by [23] and analysed also in [24].

Since these real datasets are relatively small compared to the synthetic datasets we consider, and because we do not know when drift occurs (or, indeed, if there is any drift) we simulate concept drift, joining the three datasets, merging attributes, and supposing that each dataset corresponds to a different concept, as described in [13].

Text data is also an important source of data streams in the real-world. We consider the following two sources:

20 Newsgroups. [25] is a dataset commonly used in cluster analysis. It has 19300 entries, each represented with 1000 binary attributes (word presence/absence), corresponding to at least one of 20 newsgroups. We convert this dataset into

20 binary classification problems, one for each newsgroup, and append them all into one large binary-class dataset. Thus we model a stream (similarly to CovPokElec) of 386,000 records with 19 shifts in concept.

IMDB. A dataset used in multi-label learning [26]. It contains 120919 textual movie plot summaries of 1000 binary class attributes and 0/1-associations to genres. We use the *drama* genre, which is the most frequently occurring.

3.2 Methodology

The experiments were performed on 2.66 GHz Core 2 Duo E6750 machines with 4 GB of memory. We used the Interleaved Test-Then-Train evaluation methodology: every example was used for testing the model before using it to train. From the synthetic concepts we generated 1 million examples with the following parameters:

- RBF(x,v): RandomRBF of 5 classes with x centroids moving at speed v.
- HYP(x,v): Hyperplane of 2 classes with x attributes changing at speed v.
- SEA(v): SEA dataset, with length of change v.
- LED(v): LED dataset, with length of change v.

The Nemenyi test [27] is used for computing significance: it is an appropriate test for comparing multiple algorithms over multiple datasets, being based on the average ranks of the algorithms across all datasets. We use a p-value of 0.05. Under the Nemenyi test, $\{x\} \succ \{z\}$ indicates that algorithm x is statistically significantly more likely to be more favourable than z.

In [28] the use of RAM-Hours is introduced as an evaluation measure of the resources used by streaming algorithms. Every GB of RAM deployed for 1 hour equals one RAM-Hour.

3.3 Parameter Selection

As discussed in Section 1, both lazy-learning and batch-incremental methods require a *window* parameter, which determines the number of examples used by their model (the only difference being that lazy methods learn from this window incrementally rather than as a batch). We conduct an analysis on the effects of different window-size parameters for `kNN` and `AWE-` methods. Table 2 displays results for a variety of window/batch sizes, which provides justification for our choice of parameters in the following section: $w = 1000$ and $w = 500$ for `kNN` and `AWE-`, respectively. Although $w = 5000$ results in slightly better accuracy for `kNN`, the computation cost is clearly potentially prohibitive. The setting of $w = 500$ appears to work well for all batch methods both with respect to classification and time and memory performance. However, the optimal w is different for each stream. For example as we see in Table 3, `AWE-J48` has no clear optimal value for all datasets.

Table 2. Finding the best window size for kNN, and AWE-

(a) Average accuracy

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	66.32	80.24	82.33	**82.63**
AWE-J48	70.72	**77.36**	76.90	73.76
AWE-LR	68.77	**69.62**	67.83	65.56
AWE-SMO	67.13	**70.77**	70.07	67.67

(b) Total Time (seconds)

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	2,180	9,993	18,349	71,540
AWE-J48	3,809	6,883	10,865	28,429
AWE-LR	9,659	66,757	10,247	10,112
AWE-SMO	13,860	5,800	6,414	39,298

(c) RAM-Hours

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	0.13	1.11	2.98	41.27
AWE-J48	1.96	8.49	21.81	221.66
AWE-LR	12.65	48.07	22.47	67.52
AWE-SMO	3.19	4.12	9.36	255.96

Table 3. Finding the best window size for AWE-J48

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
20 Newsgroups	94.30	94.74	**95.06**	94.60
IMDB	**55.09**	53.59	53.54	54.33
CovType	55.79	**87.82**	85.58	76.05
Electricity	**78.47**	75.27	74.37	65.10
Poker	76.06	77.89	**79.32**	75.98
CovPokElec	68.03	**81.60**	81.45	74.32
LED(50000)	70.60	71.99	**72.03**	71.37
SEA(50)	84.95	88.03	88.56	**88.68**
SEA(50000)	84.63	87.71	88.16	**88.43**
HYP(10,0.0001)	66.69	71.58	73.41	**78.63**
HYP(10,0.001)	70.95	75.79	77.69	**79.94**
RBF(0,0)	69.42	83.01	84.96	**87.38**
RBF(50,0.0001)	69.12	**79.30**	77.05	60.75
RBF(10,0.0001)	68.49	81.79	**82.78**	80.79
RBF(50,0.001)	**53.78**	50.95	38.55	24.50
RBF(10,0.001)	65.18	76.76	77.92	**79.36**
Average	70.72	**77.36**	76.90	73.76

4 Results: Batch-Incremental versus Instance-Incremental

Table 4 displays the final accuracy and resource use (time and RAM-hours) of methods and their parameters (see Table 1). Accuracy is measured as the final percentage of examples correctly classified over the test/train inter-leaved evaluation.

Naive Bayes (NB) uses very few resources, but its accuracy is poor compared to all other methods; with a few exceptions: HYP(10,0.0001), the SEA streams, and to some extent on Electricity and IMDB. Such low average Naive Bayes accuracy indicates that the concepts to be learned are reasonably hard problems. A similar scenario is observed for Stochastic Gradient Descent (SGD) which is only seriously competitive on the 20 Newsgroups dataset.

As claimed in the literature, Hoeffding Trees (HT) are generally a better option than NB for instance-incremental methods. Table 4 provides an important comparison between Hoeffding trees and non-incremental decision trees in a batch setting (AWE-J48). It shows that, although the computational cost of AWE-J48

Table 4. Comparison of all methods

(a) Accuracy

	NB	kNN	HT	AWE-J48	LB-HT	SGD	AWE-LR	AWE-SMO	LB-kNN
20 Newsgroups	68.13	94.86	94.30	94.74	94.38	94.86	88.43	95.56	DNF
IMDB	60.42	60.82	63.51	53.59	61.76	63.79	53.96	54.52	62.44
CovType	60.52	92.22	80.31	87.82	88.61	60.70	84.50	84.24	92.39
Electricity	73.36	78.38	79.20	75.27	88.77	57.58	70.55	68.56	80.78
Poker	59.55	69.35	76.07	77.89	94.97	68.92	60.90	60.38	70.34
CovPokElec	24.24	78.41	79.34	81.60	92.41	68.06	70.07	69.77	79.09
LED(50000)	54.02	63.20	68.65	71.99	73.15	11.84	73.03	72.80	69.77
SEA(50)	85.37	86.80	86.42	88.03	88.24	85.41	89.44	89.57	88.00
SEA(50000)	85.38	86.55	86.42	87.71	88.80	85.21	89.01	89.15	87.74
HYP(10,0.0001)	91.25	83.29	89.04	71.58	88.06	79.54	93.73	93.41	87.10
HYP(10,0.001)	70.91	83.33	78.77	75.79	84.85	71.10	91.75	92.02	86.91
RBF(0,0)	51.21	88.99	83.25	83.01	89.70	16.63	46.91	50.52	90.59
RBF(50,0.0001)	30.99	89.36	45.49	79.30	76.70	16.63	54.89	57.85	90.49
RBF(10,0.0001)	52.10	89.30	79.24	81.79	85.54	16.63	50.96	52.80	90.73
RBF(50,0.001)	29.14	84.03	32.29	50.95	55.72	16.63	46.48	50.42	82.10
RBF(10,0.001)	51.96	88.34	76.39	76.76	81.82	16.63	49.37	50.74	88.93
Average	59.29	82.33	74.92	77.36	83.34	51.89	69.62	70.77	83.16

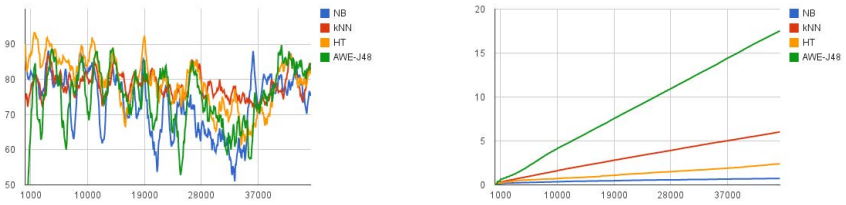
Nemenyi significance: kNN≻NB; kNN≻SGD; LB-HT≻NB; LB-HT≻SGD; LB-kNN≻NB; LB-kNN≻SGD;

(b) Time (seconds)

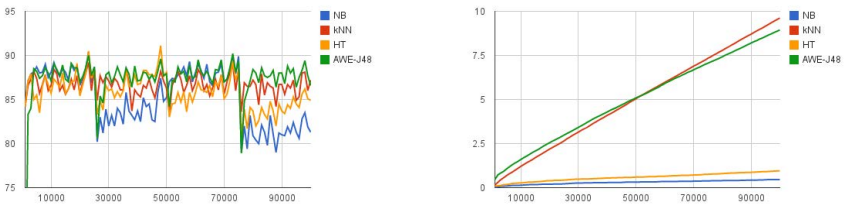
	NB	kNN	HT	AWE-J48	LB-HT	SGD	AWE-LR	AWE-SMO	LB-kNN
20 Newsgroups	93.48	11,544.68	177.34	3,448.89	4,996.64	5.45	3,187.23	293.93	DNF
IMDB	27.85	2,761.45	49.54	1,855.76	1,563.07	1.49	1,240.25	224.85	63,784.28
CovType	18.52	266.18	20.06	91.72	247.59	5.43	823.42	257.51	6,708.11
Electricity	0.64	7.05	1.15	4.64	8.72	0.32	4.32	9.73	163.61
Poker	8.88	177.82	9.26	81.80	127.59	2.29	381.05	322.51	3,454.15
CovPokElec	47.72	1,447.92	46.77	284.39	1,032.43	8.60	2,006.50	899.75	30,329.08
LED(50000)	9.14	447.36	15.90	135.10	189.13	2.04	57,466.83	1,662.00	10,816.07
SEA(50)	2.90	107.82	4.63	63.46	94.44	1.30	56.04	91.96	3,138.38
SEA(50000)	3.01	112.84	4.80	60.83	94.62	1.41	56.18	95.59	3,125.99
HYP(10,0.0001)	4.33	222.72	8.46	103.50	221.79	1.45	191.67	164.04	6,492.46
HYP(10,0.001)	4.34	222.26	9.70	103.62	224.57	1.46	193.61	159.64	6,379.68
RBF(0,0)	8.20	206.97	13.90	136.90	203.74	2.38	227.62	357.36	6,279.06
RBF(50,0.0001)	8.24	200.41	14.92	128.60	236.72	2.41	234.64	332.03	7,494.13
RBF(10,0.0001)	7.31	202.51	13.00	132.31	200.82	1.70	231.30	311.82	5,523.36
RBF(50,0.001)	8.34	218.12	14.16	120.13	234.15	2.41	225.13	303.85	6,765.54
RBF(10,0.001)	7.40	202.84	12.94	131.49	201.36	1.68	231.23	313.33	5,858.01
Total	260.28	18,348.95	416.53	6,883.14	9,877.38	41.82	66,757.02	5,799.90	166,311.91

(c) RAM-Hours (MB)

	NB	kNN	HT	AWE-J48	LB-HT	SGD	AWE-LR	AWE-SMO	LB-kNN
20 Newsgroups	0.01	2.18	3.61	4.94	340.77	0.00	10.24	0.63	DNF
IMDB	0.00	0.41	0.38	2.63	20.39	0.00	3.76	0.44	33.95
CovType	0.00	0.04	0.01	0.08	0.05	0.00	0.75	0.25	4.21
Electricity	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
Poker	0.00	0.01	0.00	0.03	0.20	0.00	0.13	0.12	0.88
CovPokElec	0.00	0.25	0.11	0.33	1.95	0.00	2.31	1.12	20.94
LED(50000)	0.00	0.03	0.01	0.10	0.49	0.00	30.38	0.95	4.47
SEA(50)	0.00	0.00	0.00	0.01	1.95	0.00	0.01	0.02	0.65
SEA(50000)	0.00	0.00	0.00	0.01	0.89	0.00	0.01	0.02	0.65
HYP(10,0.0001)	0.00	0.01	0.00	0.04	13.30	0.00	0.06	0.05	1.77
HYP(10,0.001)	0.00	0.01	0.01	0.04	1.54	0.00	0.06	0.05	1.75
RBF(0,0)	0.00	0.01	0.00	0.06	3.01	0.00	0.07	0.11	1.69
RBF(50,0.0001)	0.00	0.01	0.00	0.05	0.20	0.00	0.07	0.10	2.01
RBF(10,0.0001)	0.00	0.01	0.00	0.06	3.31	0.00	0.07	0.09	1.50
RBF(50,0.001)	0.00	0.01	0.00	0.05	0.02	0.00	0.07	0.09	1.81
RBF(10,0.001)	0.00	0.01	0.00	0.06	3.08	0.00	0.07	0.09	1.59
Total	0.02	2.98	4.15	8.49	391.16	0.00	48.07	4.12	77.90
Total without 20 Newsgroups	0.01	0.80	0.54	3.55	50.39	0.00	37.83	3.49	77.90



(a) A selection of methods on Electricity dataset, accuracy (left) time (right)



(b) A selection of methods on SEA dataset, accuracy (left) time (right)

Fig. 1. Classification accuracy (left) and running time (right) over time for methods on a selection of datasets

is often up to 10 times greater than Hoeffding trees (see also Figure 1), it often improves on them–for example on the RBF(50,0.0001) and CovType streams. On the other hand, on IMDB and Electricity, this trend is reversed; `HT` is superior. These two real-world datasets do not contain any obvious abrupt concept drift, thus supporting the claim that batch approaches automatically deal to some extent with concept drift. Figure 1b clearly shows the importance of adapting or changing models when there is concept drift.

Under a modern adaptive bagging scheme (`LB-HT`) (which resets models when drift is detected) Hoeffding Trees are powerful, albeit – in many cases – at an increased computational cost, particularly with regard to RAM-Hours.

The lazy `kNN` method performs very well across all data sources, and even as a standalone method it is one of the highest-performing methods overall. This is particularly surprising since `kNN`'s model is based on an internal buffer of 1000 instances; `kNN` models a concept well with a relatively small number of examples. We also note that in our experiments `AWE` is based only upon $n \times w = 5000$ instances; a small number relative to the size of the streams. The strengths of `kNN` are even apparent on datasets without drift, but it competes best on the most evolving data since it models new examples as soon as they arrive in the stream. Only `LB-HT` can compete seriously with the `kNN` methods, but the difference in accuracy is insignificant. It is true that the time costs of `kNN` can be quite high, and that `LB-kNN` is one of the most expensive methods to run, but this can be mitigated somewhat by

different search techniques, as explained in [3]. kNN is particularly robust: it obtains very good results across a wide range of data sources.

As expected, AWE gives different performance depending on its base classifier. This illustrates a key advantage of batch-methods: any existing classifier can be used. Under all the SEA and HYP streams and the 20 Newsgroups text dataset AWE-SMO obtains the best accuracy of all methods. On the other hand; its accuracy is very poor compared to several other methods on the RBF streams and the Electricity and Poker datasets; contributing significantly to its overall average performance. With the exception of the HYP streams, Logistic Regression (AWE-LR) does not make much of an impact in classification accuracy, and clearly runs very slowly on many datasets.

A summary of the most important observations are:

- storing a model from recent examples can be as effective as learning incrementally and keeping statistics from hundreds of thousands of examples,
- the best batch size is dependent on the data stream in consideration; and
- certain batch methods excel on certain problems, but lazy learners provide similar or better classification performance using less resources.

5 Conclusions

We investigated a variety of methods from two distinct branches of the data-stream literature: *instance-incremental* and *batch-incremental* approaches to classification. Our extensive and varied empirical evaluation of real and synthetic data sources of up to 1 million training instances with diffrent types and magnitudes of concept drift provide us with enough evidence to draw some important novel conclusions: instance-incremental methods perform similarly to their equivalent batch-learning implementation while using fewer resources. An explicit drift-detection and adaption mechanism is essential for any learner which does not automatically discard old information. We found lazy methods perform exceptionally well when using just a buffer of the 1000 most recent instances, even compared to powerful incremental methods.

References

1. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: KDD, pp. 139–148 (2009)
2. Beringer, J., Hüllermeier, E.: Efficient instance-based learning on data streams. Intelligent Data Analysis 11(6), 627–650 (2007)
3. Zhang, P., Gao, B.J., Zhu, X., Guo, L.: Enabling fast lazy learning for data streams. In: ICDM, pp. 932–941 (2011)
4. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, pp. 338–345. Morgan Kaufmann (1995)
5. Domingos, P., Hulten, G.: Mining high-speed data streams. In: KDD, pp. 71–80 (2000)

6. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SDM (2007)
7. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine Learning 11, 63–91 (1993)
8. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: ICML, pp. 161–168 (2006)
9. Bottou, L.: Online algorithms and stochastic approximations. Online Learning and Neural Networks (1998)
10. Oza, N.C., Russell, S.J.: Experimental comparisons of online and batch versions of bagging and boosting. In: KDD, pp. 359–364 (2001)
11. Oza, N., Russell, S.: Online bagging and boosting. In: Artificial Intelligence and Statistics 2001, pp. 105–112. Morgan Kaufmann (2001)
12. Bifet, A., Gavaldà, R.: Adaptive Learning from Evolving Data Streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009)
13. Bifet, A., Holmes, G., Pfahringer, B.: Leveraging Bagging for Evolving Data Streams. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 135–150. Springer, Heidelberg (2010)
14. Qu, W., Zhang, Y., Zhu, J., Qiu, Q.: Mining Multi-label Concept-Drifting Data Streams Using Dynamic Classifier Ensemble. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 308–321. Springer, Heidelberg (2009)
15. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD 2003, pp. 226–235. ACM, New York (2003)
16. Spyromitros-Xioufis, E., Spiliopoulou, M., Tsoumakas, G., Vlahavas, I.: Dealing with concept drift and class imbalance in multi-label stream classification. In: IJCAI, pp. 1583–1588 (2011)
17. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Journal of Machine Learning Research, JMLR (2010)
18. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: KDD, pp. 377–382 (2001)
19. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD, pp. 97–106 (2001)
20. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
21. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
22. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: KDD, pp. 523–528 (2003)
23. Harries, M.: Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales (1999)
24. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
25. Lang, K.: The 20 newsgroups dataset (2008), `http://people.csail.mit.edu/jrennie/20Newsgroups/`
26. Read, J., Bifet, A., Holmes, G., Pfahringer, B.: Scalable and efficient multi-label classification for evolving data streams. Machine Learning, 1–30 (2012)
27. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30 (2006)
28. Bifet, A., Holmes, G., Pfahringer, B., Frank, E.: Fast Perceptron Decision Tree Learning from Evolving Data Streams. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 299–310. Springer, Heidelberg (2010)

Applying Piecewise Approximation in Perceptron Training of Conditional Random Fields

Teemu Ruokolainen

Department of Information and Computer Science,
Aalto University, FI-00076 AALTO, FINLAND
teemu.ruokolainen@aalto.fi

Abstract. We show that the recently proposed piecewise approximation approach can benefit conditional random fields estimation using the structured perceptron algorithm. We present experiments in noun-phrase chunking task on the CoNLL-2000 corpus. The results show that, compared to standard training, applying the piecewise approach during model estimation may yield not only savings in training time but also improvement in model performance on test set due to added model regularization.

1 Introduction

The conditional random field (CRF) model presented by Lafferty et al. [1] provides a flexible framework for structured classification tasks involving multiple interdependent output variables. The essential idea of CRFs is to exploit an undirected graph defined over the output variables and condition the output globally on the input. CRF training corresponds to estimation of model parameters based on a set of training examples and is commonly performed using the maximum likelihood (ML) criterion [1] or margin-based approaches [2,3]. In this work, we consider CRF training using the structured perceptron algorithm presented by Collins [2]. Recently, the structured perceptron was shown to be the basis for a state of the art approach to structured prediction in [4].

We show that CRF model training using the structured perceptron algorithm can benefit from the piecewise approximation approach presented by Sutton and McCallum [5]. In piecewise training, we split the original graphs corresponding to training instances into smaller and possibly overlapping subgraphs (pieces) and perform graph inference on these small subgraphs rather than on the assumably much larger original graphs. Subsequent to training, we apply graph inference on full graphs corresponding to the test instances using the estimated model parameters in order to exploit the dependency structure between the output variables.

Our motivation for using the piecewise approximation is two-fold. First, it is computationally less costly to perform inference on the subgraphs than on the larger original graphs. Therefore, the splitting operation is expected to speed up the training in case the training time is dominated by time consumed by

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 324–333, 2012.
© Springer-Verlag Berlin Heidelberg 2012

the graph inference – this was the main motivation behind the work of Sutton and McCallum. Second, the piecewise approach may provide additional regularization, that is, robustness against noise in the training data. This view is also supported by the empirical results presented by Sutton and McCallum, in which using the piecewise approximation yielded small gains in performance in many natural language processing tasks compared to standard training.

The central idea of this work is to simplify the training of the structured perceptron (or of some general structured classifier) by modifying the graph inference step. Recent studies in this line of work include, for example, the pseudo-max approach presented by Sontag et al. [6], the decomposed learning by Samdani [7], and the amortizing approach by Srikumar [8]. Compared to the work presented here, none of the previous studies consider training the model on the subgraphs in a straightforward manner.

We summarize the contribution of the presented work as follows. First, the presentation of Sutton and McCallum focused on applying the piecewise approach to a CRF model trained using the ML criterion. We extend their work by showing that the approach can be applied successfully also in perceptron training. Second, our work provides support for the idea of using the piecewise approximation as a regularization method in order to improve model performance when the number of training instances is low.

The rest of the paper is organized as follows. In Sect. 2, we describe CRF model training and the standard and piecewise approximated variants of the structured perceptron training algorithm. In Sect. 3, we describe the applied experiment setup and present the obtained results along with a discussion. Finally, we present conclusions on the work in Sect. 4.

2 Methods

2.1 Conditional Random Fields

The aim of our work is to estimate a conditional probability model $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w})$ parameterized by a real-valued vector $\boldsymbol{w}$ for an arbitrary input vector $\boldsymbol{x}$ and a discrete output vector $\boldsymbol{y}$ so that the model can be used to predict output for previously unseen inputs subsequent to training. The model estimation is equivalent to finding the model parameters $\boldsymbol{w}^*$ minimizing the average loss [9] on a training data set D consisting of n independently and identically distributed (i.i.d.) input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$. This corresponds to an optimization problem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_i \ell(\boldsymbol{y}^*, \boldsymbol{y}^i) \tag{1}$$

where $\ell(\boldsymbol{y}^*, \boldsymbol{y}^i)$ denotes the real-valued sample loss function determining the cost of making a prediction $\boldsymbol{y}^*$ given the correct output $\boldsymbol{y}^i$. The prediction $\boldsymbol{y}^*$ is defined here as the maximum a posteriori (MAP) estimate corresponding to an optimization problem

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} p\left(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w}\right) \tag{2}$$

also referred to as MAP inference. The probability function $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w})$ corresponding to the CRF model of Lafferty et al. [1] is expressed as

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w}) \propto \prod_c \psi_c(\boldsymbol{y}_c, \boldsymbol{x}_c; \boldsymbol{w}) \tag{3}$$

where c indexes the node cliques[1] in an undirected graph, or equivalently a random field, defined over the output $\boldsymbol{y}$. The factors ψ_c are defined to be log-linear in parameters $\boldsymbol{w}$ formally written as

$$\psi_c(\boldsymbol{y}_c, \boldsymbol{x}_c; \boldsymbol{w}) = \exp\Big(\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c)\Big) \tag{4}$$

where $\boldsymbol{u}^\top \boldsymbol{v}$ denotes the vector dot product and the real-valued feature extracting function $\boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c)$ captures the co-occurrence behavior of the output $\boldsymbol{y}$ and a set of features describing the input $\boldsymbol{x}$ at clique c.

2.2 Structured Perceptron Training

Here, we describe the structured perceptron algorithm of Collins [2] in terms of loss functions. Given the model (3) and the log-linear factors (4), the structured perceptron algorithm corresponds to a sample loss

$$\ell_{perc} = \sum_c \boldsymbol{w}^\top \Big(\boldsymbol{f}(\boldsymbol{y}_c^*, \boldsymbol{x}_c) - \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c)\Big) \tag{5}$$

with a corresponding gradient with respect to $\boldsymbol{w}$ expressed as

$$\nabla \ell_{perc} = \sum_c \Big(\boldsymbol{f}(\boldsymbol{y}_c^*, \boldsymbol{x}_c) - \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c)\Big). \tag{6}$$

where the MAP estimates $\boldsymbol{y}_c^*$ for each node clique c are obtained by solving the exact MAP estimate $\boldsymbol{y}^*$ over the complete graph.

Given the sample loss (5), the risk minimization problem (1) can be solved using a stochastic gradient algorithm with a fixed learning rate ($\eta = 1$) presented by Collins. The algorithm proceeds in three steps by evaluating the factor functions (4) using current parameter setting, performing the MAP inference (2) and updating the parameters according to the gradient (6) one training instance at a time. The algorithm converges to a perfect fit, that is the gradient becomes zero for every instance in training set, if and only if the data is linearly separable. The detailed convergence analysis is provided by Collins in [2].

2.3 Piecewise Approximated Perceptron Training

In this section, we discuss a variant of the perceptron algorithm obtained using the *piecewise approximation* presented by Sutton and McCallum [5]. In the

[1] Node clique is a subset of graph nodes with every pair of nodes connected by an edge.

piecewise approximation, we split the original graphs corresponding to training instances into small and possibly overlapping subgraps (pieces) and subsequently consider the subgraphs as i.i.d. instances replacing the original samples in the training set. Subsequent to training, the acquired model parameters are applied in a standard manner to obtain the MAP estimates (2) for the test instances.

For the rest of this paper, we focus on a special case of the piecewise approximation referred to as the *factor-as-piece* approach by Sutton and McCallum, in which each node clique c in a given graph constitutes its own separate subgraph yielding an approximated sample loss

$$\hat{\ell} = \sum_c \ell(\hat{\boldsymbol{y}}_c^*, \boldsymbol{y}_c) \tag{7}$$

where the approximated MAP estimates $\hat{\boldsymbol{y}}_c^*$ for each node clique c are inferred based solely on the input as

$$\hat{\boldsymbol{y}}_c^* = \arg\max_{\boldsymbol{y}_c} \psi_c(\boldsymbol{y}_c, \boldsymbol{x}_c; \boldsymbol{w})\,. \tag{8}$$

Note that this choice completely eliminates propagation of information across the original graph during inference – the approximation is therefore expected to work well when the input alone is sufficiently informative to enable reasonable predictions.

Applying the approximation (7) to the perceptron sample loss (5) then yields

$$\hat{\ell}_{perc} = \sum_c \boldsymbol{w}^\top \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_c^*, \boldsymbol{x}_c) - \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c) \Big) \tag{9}$$

with a corresponding approximate gradient

$$\nabla \hat{\ell}_{perc} = \sum_c \Big(\boldsymbol{f}(\hat{\boldsymbol{y}}_c^*, \boldsymbol{x}_c) - \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c) \Big) \tag{10}$$

where, given the log-linear factor form (4), the approximate MAP estimates $\hat{\boldsymbol{y}}_c^*$ (8) are obtained simply as

$$\hat{\boldsymbol{y}}_c^* = \arg\max_{\boldsymbol{y}_c} \Big(\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{y}_c, \boldsymbol{x}_c) \Big)\,. \tag{11}$$

As a consequence of eliminating the use of output structure during inference according to (8) and (11), it is more difficult for the CRF model (3) to fit the training data. Due to this rigidity, the approximation may have the potential benefit of adding robustness against noise in the training data compared to standard training. This is supported by the experiments presented in Sect. 3.

The computational load of performing the inference after eliminating the output structure using (11) is negligible compared to performing inference algorithm on the original graphs. Consequently, applying the approximation is expected to speed up the training in case the training time is dominated by the time consumed by the graph inference. This condition may be violated if the graph

structure allows efficient (exact) inference and the number of model parameters is exceptionally high (say, millions). In this case the time consumed by the evaluation of the factor functions (4) and performing the gradient updates using (10) can make the reduction in total training time insignifcant.

The final observation on the factor-as-piece approximation discussed here relates to both the model rigidity and total training time. Essentially, the added rigidity means that the approximated perceptron algorithm is not likely to reach as low averaged loss over training instances as the standard variant. Consequently, in case the standard training converges to a perfect fit, the approximated algorithm may fail to do so. However, one can argue that even in this case, the training time of the approximated algorithm will still be low if the training time is dominated by the inference algorithm and a reasonable maximum number of passes over the training set is applied. One could also use alternative stopping criteria such as terminating the training if no new minimum averaged loss has been obtained in the past k passes over training set. Again, the experiments presented in Sect. 3 provide support for this argument.

2.4 Parameter Averaging

In order to observe how the piecewise approximation behaves in the presence of another regularization scheme, the experiments described in Sect. 3 employ the parameter averaging approach [2]. The averaging approach was originally proposed by Freund and Schapire [10] as an approximation to the voting scheme presented in the same work. In this approach, the parameter setting applied to the test set is the average of the parameters obtained after processing each training instance during the stochastic gradient descent algorithm. An efficient implementation technique for the approach described by Daume can be found in [11, p. 19].

3 Experiments

3.1 Setup

We present experiments conducted in noun-phrase chunking on the CoNLL-2000 corpus [12], which is a subset of the widely applied Penn Treebank corpus [13]. Noun-phrase chunking is a commonly applied benchmark task for classification algorithms and was also considered in the experimental sections of Collins [2] and Sutton and McCallum [5].

Each word token in the CoNLL-2000 corpus is annotated with a corresponding phrase chunk. The tag set we consider consists of three noun-phrase labels (*beginning of noun-phrase*, *inside a noun-phrase* and *outside a noun-phrase* which subsumes other phrase tags in the corpus). In addition, we assume tags and word markers for beginning and end of each sentence. By default, the corpus is divided into training (8,935 sentences, 211,727 tokens) and test (2,011 sentences, 47,377 tokens) sets.

The graph structure used is the linear chain [1]. While the model presentation in [1] includes node cliques consisting of exactly two nodes, our implementation includes also node cliques of size one. This provides robustness against data sparseness. In the linear chain, the MAP estimate over the complete graph can be obtained using the standard Viterbi algorithm (described e.g. by Bishop in [14, Chapter 8]).

We next describe the features included in feature vector in factor (4). For outputs corresponding to positions t and $(t-1, t)$ for single and double node cliques, respectively, we extract the feature set describing the input from words in positions $t-2, .., t+2$. The features take binary values and include the word identities (such as *entertaining* and *entertainment*), presence of common English suffixes (such as *-ing* and *-ment*), and presence of useful orthographic properties (such as letter capitalization and special characters) at each of the five word positions. We additionally use a default feature which is always active independent of the input. Combinations of input features and output configurations not appearing in the training set are exluded. The number of features, equal to the number of model parameters, extracted in this manner from the complete training set is 311,747.

The model performance is evaluated using the *F-measure* and *per-label accuracy*. The F-measure equals the geometric mean of precision (the percentage of correctly assigned phrases with respect to all assigned phrases) and recall (the percentage of correctly assigned phrases with respect to the gold standard phrases), whereas the per-label accuracy is simply the percentage of correctly assigned tags. Therefore, both measures take values between 0 and 100, with higher values indicating better performance.

Given the specifications above, the training procedure is as follows. We first form a training set by randomly selecting a subset of the available 8,936 sentences – used subset sizes are 559, 1,117, 2,234, 4,468, and 8,936 sentences. We then train linear chain CRF models on the subset applying the standard and the factor-as-piece variants of the perceptron algorithm described in Sect. 2.2 and 2.3, respectively. The variants initialize the model parameters with zero vectors and process the training instances in identical order. Both variants terminate training if no convergence is reached within 50 passes over training set (the perceptron algorithm has been reported to reach optimal performance already around 10 passes [2]). At the end of each training set pass, models corresponding to averaged parameter settings described in Sect. 2.4 are applied to the test set. This procedure is then repeated 50 times to take into account the stochastic noise introduced by the on-line perceptron algorithm – the training subset is resampled at the beginning of each repetition. The experiments are run on a computer grid using our own single-threaded Python-based implementation[2] utilizing the PySparse sparse matrix library[3].

3.2 Results and Analysis

The training CPU times and obtained average losses over the training set and model performances on the test set are presented in Tables 1, 2, 3 and 4. The

[2] The implementation can be requested from the author.

[3] `http://pysparse.sourceforge.net/`

training times include the time consumed in evaluating the factor functions, applying inference, performing the gradient descent and maintaining the averaged parameters. Time used for data preprocessing, data input and output, or evaluation of the test set have been excluded. The reported average losses and test set performances are the lowest and highest obtained during the training, respectively. All the presented values are averages and standard deviations (in parentheses) computed over the 50 experiment repetitions.

We summarize the results as follows. First, on all training set sizes, the model estimated using the factor-as-piece approximation used less time on average for training compared to the model estimated using the standard perceptron algorithm. The minimum and maximum obtained time reductions were 26% and 61% corresponding to training set size of 1,117 and 2,234 sentences, respectively. Second, for all training set sizes, the standard perceptron algorithm achieved lower average loss computed over the training set. The standard variant reached zero loss in majority of experiment repetitions using a low number of training instances (559 and 1,117 sentences), whereas the approximated training converged only once on the smallest data set. Third, for all training set sizes, the model estimated using the factor-as-piece variant gave higher average performance on the test set. The minimum and maximum relative improvements obtained were 1.5% and 0.1% corresponding to training set sizes of 559 and 4,468 sentences, respectively, using the F-measure. Using the per-label accuracy, the minimum and maximum relative improvements were 0.8% and 0.2% corresponding to training set sizes of 559 and 4,468 sentences, respectively.

In order to assess the statistical significance of the improvement in performance obtained applying the factor-as-piece approximation compared to standard training, we applied the right-tailed Wilcoxon signed-rank test for repeated measurements using the large sample approximation. We conclude that the factor-as-piece approximated perceptron algorithm yielded higher performance compared to the standard variant with high confidence levels (with p-values of 9.0×10^{-4} and 7.5×10^{-10} or lower using F-measure and per-label accuracy, respectively, on all training set sizes).

3.3 Discussion

Compared to standard training, the factor-as-piece approach consistently yielded improvement in model performance and higher average losses over training sets. These results imply that the added rigidity to the model caused by the factor-as-piece approximation can in fact perform useful model regularization. As a general trend, the relative improvements were larger using small training sets compared to large ones. This was expected since increasing the number of training instances should alleviate the problem of model overfitting. We note that these improvements were obtained in the presence of another powerful regularization approach, namely, the parameter averaging approach.

The factor-as-piece approach also yielded consistent reductions in consumed training CPU time compared to standard training. The obtained relative reductions were largest for high numbers of training instances (4,468 and 8,936

Table 1. Training time consumption in CPU seconds as a function of training set size n in sentences, using the piecewise approximated and standard training

n	piecewise	standard
559	22.1 (0.6)	33.0 (6.9)
1117	26.2 (3.0)	53.2 (9.0)
2234	50.6 (1.1)	131.3 (5.6)
4468	103.2 (2.2)	261.0 (1.7)
8936	212.9 (4.9)	504.0 (9.8)

Table 2. Minimum average losses obtained during training as a function of training set size n in sentences, using the factor-as-piece approximated and standard training

n	piecewise	standard
559	0.05 (0.04)	0.0009 (0.004)
1117	0.11 (0.04)	0.007 (0.02)
2234	0.25 (0.05)	0.02 (0.02)
4468	0.64 (0.05)	0.04 (0.02)
8936	1.68 (0.02)	0.1 (0.01)

Table 3. Highest F-measures obtained on test set as a function of training set size in sentences n, using the factor-as-piece approximated and standard training

n	piecewise	standard
559	79.1 (0.4)	77.9 (0.5)
1117	82.0 (0.3)	81.2 (0.4)
2234	84.4 (0.3)	84.2 (0.3)
4468	86.3 (0.2)	86.2 (0.2)
8936	88.0 (0.1)	87.9 (0.1)

Table 4. Highest per-label accuracies obtained on test set as a function of training set size in sentences n, using the factor-as-piece approximated and standard training

n	piecewise	standard
559	92.0 (0.2)	91.2 (0.2)
1117	93.2 (0.1)	92.7 (0.1)
2234	94.2 (0.1)	94.0 (0.1)
4468	95.0 (0.1)	94.9 (0.1)
8936	95.7 (0.04)	95.5 (0.05)

sentences). In this case neither training variant reached convergence within the allowed training set passes. However, reductions were also obtained with small data sets where the standard perceptron variant converged before the maximum number of training set passes were reached. Therefore, the possible failure to converge to a perfect fit due to applying the piecewise approximation does not necessarily imply higher training time if reasonable stopping criteria are applied.

As a preprocessing step, all the combinations of input features and output configurations which did not appear in the training data set were excluded from the feature set. This frequency-based cutoff considerably reduces the number of model parameters (2,787,960 versus 311,747). Our preliminary experiments indicated that reducing the number of parameters in this manner ensured that the training times were dominated by the graph inference using our implementation.

The experiments were conducted using the standard linear chain graph structure, in which the graph inference can be performed efficiently. Meanwhile, the inference is much more costly in tasks involving complex, cyclic graphs. In these cases it is generally not feasible to perform exact inference and approximative MAP inference algorithms such as loopy belief propagation [15], tree-reweighted message passing algorithm [16] and graph cuts [17], are applied instead. Applying the piecewise approximation in the presence of cyclic graphs is expected to yield higher savings in training time as discussed by Sutton and McCallum [5].

4 Conclusions

In this work we have showed that CRF training using the structured perceptron algorithm [2] can benefit from the piecewise approximation approach [5]. The approximation may yield not only savings in total training time but also improved model performance on test instances. The lower training time can be obtained in case the training time is dominated by the inference algorithm. The improved performance indicates that the piecewise approach can reduce model overfitting to training data in a meaningful manner. On a more general note, the work shows that the piecewise approach can be successfully applied to margin-based training of CRFs. A potential direction of future work is to study the piecewise approach further in combination with margin-based CRF training using structured support vector machines and graph structures, in which exact inference is infeasible.

Acknowledgements. I would like to thank the anonymous reviewers for their valuable comments. I would also like to thank Mikko Kurimo, Oskar Kohonen, Hande Topa, Ulpu Remes and Paul Wagner for the comments and help throughout the work. This work was financially supported by the Academy of Finland under the grant no 251170 (Finnish Centre of Excellence Program (20122017)) and Langnet (Finnish doctoral programme in language studies).

References

1. Lafferty, J., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289 (2001)
2. Collins, M.: Discriminative training methods for hidden markov models: Theory and Experiments with Perceptron Algorithms. In: Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing, vol. 10, pp. 1–8. Association for Computational Linguistics (2002)

3. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
4. Zhang, Y., Clark, S.: Syntactic processing using the generalized perceptron and beam search. Computational Linguistics 37(1), 105–151 (2011)
5. Sutton, C., McCallum, A.: Piecewise training for structured prediction. Machine Learning 77(2), 165–194 (2009)
6. Sontag, D., Meshi, O., Jaakkola, T., Globerson, A.: More data means less inference: A pseudo-max approach to structured learning. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, pp. 2181–2189 (2010)
7. Samdani, R., Roth, D.: Efficient decomposed learning for structured prediction. In: Proceedings of the 29th International Conference on Machine Learning (2012)
8. Srikumar, V., Kundu, G., Roth, D.: On amortizing inference cost for structured prediction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1114–1124 (2012)
9. Vapnik, V.: An overview of statistical learning theory. IEEE Transactions on Neural Networks 10, 988–999 (1999)
10. Freund, Y., Schapire, R.: Large margin classification using the perceptron algorithm. Machine Learning 37(3), 277–296 (1999)
11. Daumé III, H.: Practical structured learning techniques for natural language processing. PhD thesis, University of Southern California (2006)
12. Kim, T., Sang, E., Buchholz, S.: Introduction to the conll-2000 shared task: Chunking. In: Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning, vol. 7, pp. 127–132. Association for Computational Linguistics (2000)
13. Marcus, M., Marcinkiewicz, M., Santorini, B.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19(2), 313–330 (1993)
14. Bishop, C.: Pattern recognition and machine learning, vol. 4. Springer, New York (2006)
15. Frey, B., MacKay, D.: A revolution: Belief propagation in graphs with cycles. Advances in Neural Information Processing Systems, 479–485 (1998)
16. Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Transactions on Information Theory 51, 3697–3717 (2005)
17. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 1222–1239 (2001)

Patch-Based Data Analysis Using Linear-Projection Diffusion

Moshe Salhov[1], Guy Wolf[1], Amir Averbuch[1], and Pekka Neittaanmäki[2]

[1] School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
[2] Faculty of Information Technology
University of Jyväskylä
Jyväskylä, Finland

Abstract. To process massive high-dimensional datasets, we utilize the underlying assumption that data on a manifold is approximately linear in sufficiently small patches (or neighborhoods of points) that are sampled with sufficient density from the manifold. Under this assumption, each patch can be represented by a tangent space of the manifold in its area and the tangential point of this tangent space. We use these tangent spaces, and the relations between them, to extend the *scalar* relations that are used by many kernel methods to *matrix* relations, which can encompass multidimensional similarities between local neighborhoods of points on the manifold. The properties of the presented construction are explored and its spectral decomposition is utilized to embed the patches of the manifold into a tensor space in which the relations between them are revealed. We present two applications that utilize the patch-to-tensor embedding framework: data classification and data clustering for image segmentation.

Keywords: Dimensionality reduction, manifold learning, kernel PCA, Diffusion Maps, patch processing, stochastic processing, vector processing.

1 Introduction

High-dimensional datasets have become increasingly common in many areas due to high availability of data and continuous technological advances. Classical methods for statistical analysis fail on such datasets because of a problem known as "curse of dimensionality". More recent methods, originated from the field of machine learning, assume that the observable parameters in such datasets are related to a small number of underlying factors via a set of non-linear mappings. Mathematically, this assumption is characterized by a manifold structure on which data points are assumed to lie. This underlying manifold is immersed (or submersed) in an ambient space that is defined by observable parameters. Usually, the intrinsic dimension of the underlying manifold is significantly smaller than the dimension of the ambient space.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 334–345, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Several methods have been suggested to provide a global coordinate system that represents the structure of the underlying manifold of a high-dimensional dataset. Kernel methods such as Diffusion Maps [1] have shown good results. These methods aim at extending the essence of the classical Multi-Dimensional Scaling (MDS) method [2,5] by replacing its Gram matrix with a kernel matrix while preserving the qualities represented by it instead of the inner-products that are preserved by the MDS method. The defined kernel can be thought of as an adjacency matrix of a graph whose vertices are the points in the dataset. The analysis of the eigenvalues and the corresponding eigenvectors of this matrix can reveal many qualities and connections in the graph.

A fundamental, well-based, assumption of kernel methods in general, and diffusion maps in particular, is that, locally, the manifold is approximately linear in sufficiently small patches (or neighborhoods of points). Under this assumption, each patch can, in fact, be represented (up to a small approximation error) by a tangent space of the manifold in its area and the tangential point of this tangent space. Local PCA was suggested in [8,9] to compute an approximation of suitable tangent spaces and their tangential points for patches that define neighborhoods of points that are sampled with sufficient density from the manifold.

Using the suggested representations, the relations between patches can be modeled by the usual affinity between tangential points and an operator that translates vectors from one tangent space to another. The structure of the ambient space was used in [6] to define linear-projection operators between tangent spaces and utilize them to construct a super-kernel that represents the affinity/similarity between patches. The structure of the underlying manifold was utilized for a similar purpose in [9] to define continuous parallel transport operators between tangent spaces and to define such a super-kernel by using discrete approximations of these operators. In fact, algorithmically, the approximations in [9] are achieved by orthogonalization of the linear-projection operators in [6]. Although these constructions differ by only a small modification of the construction algorithm, the resulting super-kernels have very different properties and different derived theories.

In this paper, we aim at analyzing patches of the manifold instead of analyzing single points on the manifold. Each patch is defined as a local neighborhood of a point in a dataset sampled from an underlying manifold. The relation between two patches is described by a matrix rather than by a scalar value. This matrix represents both the diffusion affinity between the points at the centers of these patches and the similarity between their local coordinate systems, which is represented by linear-projections between their tangent spaces. The constructed matrices between all patches are then combined in a block matrix, which we call a super-kernel. We explore both the finite and continuous properties of this Linear-Projection Diffusion (LPD) super-kernel, which extends the diffusion kernel that is used in Diffusion Maps.

The paper has the following structure. Section 1.1 presents the benefits of patch-based analysis. An overview of the suggested construction is presented in Section 2. Its mathematical properties are explored in Section 3. Finally,

Section 4 presents two data analysis applications and results of the presented approach.

1.1 Benefits of Patch-Based Analysis

Manifold learning approaches assume that the sampled high-dimensional data points reside on a low dimensional underlying manifold and that they are sufficient to detect and represent its structure and geometry. Since manifold-based geometries are based on local neighborhoods or patches, then the data must be dense enough to detect them. If the data is spread too sparsely over the manifold in the high-dimensional ambient space, then the application of an affinity kernel to the data will not reveal any local patches or detect the underlying manifold structure. In this case, the only available processing tools are variations of nearest-neighbor algorithms.

Hence, data points on a low-dimensional manifold in a high-dimensional ambient space can either reside in locally-defined patches, and then the method in this paper is applicable to it, or scattered sparsely all over the manifold and thus there is no detectable coherent underlying manifold that can provide an underlying structure for it. Therefore, for manifold learning applications, the local patches, and not the individual points, are the basic building blocks of the underlying structures of the dataset, and their analysis can provide a more natural representation of meaningful insights to the patterns that govern the analyzed phenomenon.

The proposed methodology in this paper is classified as a spectral method. Spectral methods are global in the sense that they usually require the relations between all the samples in the dataset. This global consideration hinders their use in practical large-scale problems due to high memory (e.g., fitting the kernel matrix in memory) and computational costs. However, in many cases there are many duplicities, or near duplicities, in massive datasets and a the number of different patches of closely-related data-points is significantly less then the number of samples in the dataset. Processing patches, instead of individual data points, reduces the many redundancies that usually occur in massive datasets, thus, it enables also to localize spectral processing and reduce these overheads and impracticalities.

2 Overview

2.1 Problem Setup

Let $M \subseteq \mathbb{R}^m$ be a set of n points sampled from a manifold $\mathcal{M}$ that lies in the ambient space $\mathbb{R}^m$. Let $d \ll m$ be the intrinsic dimension of $\mathcal{M}$, thus, it has a d-dimensional tangent space $T_x(\mathcal{M})$, which is a subspace of $\mathbb{R}^m$, at every point $x \in M$. If the manifold is densely sampled, the tangent space $T_x(\mathcal{M})$ can be approximated by a small enough patch (i.e., neighborhood) $N(x) \subseteq M$ around $x \in M$.

Let $o_x^1, \ldots, o_x^d \in \mathbb{R}^m$, where $o_x^i = (o_x^{i1}, \ldots, o_x^{im})^T,\ i = 1, \ldots, d$, form an orthonormal basis of $T_x(\mathcal{M})$ and let $O_x \in \mathbb{R}^{m \times d}$ be a matrix whose columns are these vectors:

$$O_x \triangleq \begin{pmatrix} | & & | \\ o_x^1 & \cdots & o_x^d \\ | & & | \end{pmatrix} \qquad x \in M\ . \tag{2.1}$$

We will assume from now on that vectors in $T_x(\mathcal{M})$ are expressed by their d coordinates according to the presented basis $o_x^1, \ldots, o_x^d$. For each vector $u \in T_x(\mathcal{M})$, the vector $\tilde{u} = O_x u \in \mathbb{R}^m$ is the same vector as u represented by m coordinates, according to the basis of the ambient space. For each vector $v \in \mathbb{R}^m$ in the ambient space, the vector $v' = O_x^T v \in T_x(\mathcal{M})$ is the linear projection of v on the tangent space $T_x(\mathcal{M})$.

2.2 Diffusion Maps

The original diffusion maps method [1] can be used to analyze the geometry of the manifold $\mathcal{M}$. This method is based on defining an isotropic kernel K as $k(x, y) \triangleq e^{-\frac{\|x-y\|}{\varepsilon}}$, for every $x, y \in \mathcal{M}$, where ε is a meta-parameter of the algorithm. This kernel represents the affinities between points on the manifold. Next, a degree is defined for each point $x \in \mathcal{M}$ as $q(x) \triangleq \int\limits_{y \in \mathcal{M}} k(x, y)$. Kernel normalization with this degree produces a stochastic transition operator P that is defined as $Pf(x) = \int f(y)p(x, y)dy$ for every function $f : \mathcal{M} \to \mathbb{R}$, where $p(x, y) = \frac{k(x,y)}{q(x)}$, which defines a Markov process (i.e., a diffusion process) over the points on the manifold $\mathcal{M}$. A symmetric conjugate A of the transition operator P defines the diffusion affinities between points as $a(x, y) = \sqrt{q(x)}p(x, y)\frac{1}{\sqrt{q(y)}} = \frac{k(x,y)}{\sqrt{q(x)q(y)}}$.

Spectral analysis of the diffusion affinity kernel A yields the eigenvalues $1 = \sigma_0 \geq \sigma_1 \geq \ldots$ and their corresponding eigenvectors $\psi_0, \psi_1, \ldots$, which are used to construct the desired map that embeds each data point $x \in \mathcal{M}$ onto the point $\Psi(x) = (\sigma_i \psi_i(x))_{i=0}^{\delta}$ for a sufficiently small δ, which is the dimension of the embedded space and depends on the decay of the spectrum of A. When analyzing a finite dataset $M \subset \mathcal{M}$, the continuous operators K, P and A become finite matrices.

2.3 Super-Kernel

For $x, y \in M$, let $O_{xy} = O_x^T O_y \in \mathbb{R}^{d \times d}$, where O_x and O_y were defined in Eq. 2.1. The matrices O_x and O_y represent bases of the tangent spaces $T_x(\mathcal{M})$ and $T_y(\mathcal{M})$, respectively. Thus, the matrix O_{xy} represents a linear-projection between these tangent spaces, and, in some sense, the similarity between them. We will refer to it as a tangent similarity matrix.

We use the diffusion affinity kernel A and the tangent similarity matrices O_{xy} in the following definition to introduce the concept of a *super-kernel*:

Definition 1 (Linear-Projection Diffusion Super-kernel). *A super-kernel is a matrix $G \in \mathbb{R}^{nd \times nd}$ where in terms of blocks, it is a block matrix of size*

$n \times n$ and each block in it is a $d \times d$ matrix. Each row and each column of blocks in G corresponds to a point in M, and a single block G_{xy} (where $x, y \in M$) represents an affinity or similarity between the patches $N(x)$ and $N(y)$. Each block $G_{xy} \in \mathbb{R}^{d \times d}$ of a Linear-Projection Diffusion Super-kernel *is defined as* $G_{xy} \triangleq a(x,y)O_{xy} = a(x,y)O_x^T O_y,\ x, y \in M$.

It is convenient to consider each single cell in G as an element in a block, i.e., $[G_{xy}]_{ij}$ where $x, y \in M$ and $i, j \in \{1, \ldots, d\}$. We can also use the vectors o_x^i and o_y^j to apply this indexing scheme and use the notation $g(o_x^i, o_y^j) \triangleq [G_{xy}]_{ij}$, $x, y \in M$, $i, j \in \{1, \ldots, d\}$. In this notation, it is easy to see that G is symmetric since $[G_{xy}]_{ij} = [G_{yx}^T]_{ij} = [G_{yx}]_{ji}$ (for $x, y \in M$ and $i, j \in \{1, \ldots, d\}$), where the first equality is due to the symmetry of A, the definition of G_{xy} and since $O_{xy} = O_{yx}^T$. It is important to note that $g(o_x^i, o_y^j)$ is *only a notation* for convenience reasons and a single element of a block in G does not necessarily have any special meaning. The block itself, as a whole, holds meaningful similarity information.

We will use spectral decomposition for analyzing a super-kernel G, and utilize it to embed the patches $N(x)$ of the manifold (for $x \in M$) into a tensor space. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_\ell|$ be the ℓ most significant eigenvalues of G and let $\phi_1, \phi_2, \ldots, \phi_\ell$ be their corresponding eigenvectors. According to the spectral theorem, if ℓ is greater than the numerical rank of G, then $G \approx \sum_{i=1}^{\ell} \lambda_i \phi_i \phi_i^T$, where the eigenvectors are treated as column vectors. For convenience reasons, we will treat this approximation as an equality, since, from a theoretical point of view, ℓ can always be chosen to be large enough for actual equality to hold. In practice, the exact value of ℓ depends on the numerical rank of G, the decay of its spectrum, and the exact application of the construction. Usually, however, the affinity kernel and the tangent similarity matrices can be chosen in such a way that a small ℓ will obtain sufficient accuracy for the desired task.

Each eigenvector ϕ_i, $i = 1, \ldots, \ell$, is a vector of length nd. We denote each of its elements as $\phi_i(o_x^j)$ where $x \in M$ and $j = 1, \ldots, d$. An eigenvector ϕ_i can also be regarded as a vector of n sections, each of which is a vector of length d that corresponds to a point $x \in M$ on the manifold. To express this notion we use the notation $\varphi_i^j(x) = \phi_i(o_x^j)$ (for $x \in M, i = 1, \ldots, \ell, j = 1, \ldots, d$). Thus, the section in ϕ_i, which corresponds to $x \in M$, is the vector $(\varphi_i^1(x), \ldots, \varphi_i^d(x))^T$.

We use the eigenvalues and eigenvectors of G to construct a spectral map whose definition is similar to the standard (i.e., classic) diffusion map: $\Phi(o_x^j) = (\lambda_1 \phi_1(o_x^j), \ldots, \lambda_\ell \phi_\ell(o_x^j))^T$. By using this construction, we get nd vectors of length ℓ. Each $x \in M$ corresponds to d of these vectors, i.e., $\Phi(o_x^j)$, $j = 1, \ldots, d$.

We use these vectors to construct the tensor $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$ for each $x \in M$, which is represented by the following $\ell \times d$ matrix:

$$\mathcal{T}_x \triangleq \begin{pmatrix} | & & | \\ \Phi(o_x^1) & \cdots & \Phi(o_x^d) \\ | & & | \end{pmatrix} \qquad x \in M\ . \tag{2.2}$$

In other words, the coordinates of $\mathcal{T}_x$ (i.e., the elements in this matrix) are $[\mathcal{T}_x]_{ij} = \lambda_i \varphi_i^j(x)$, $x \in M, i = 1, \ldots, \ell, j = 1, \ldots, d$. Each tensor $\mathcal{T}_x$ represents an embedding of the patch $N(x)$, $x \in M$, into the tensor space $\mathbb{R}^\ell \otimes \mathbb{R}^d$.

3 Mathematical Properties

3.1 Spectral Properties

The linear projection operators, which define the tangent similarity matrices by a LPD super-kernel, express some important properties of the manifold structure, e.g., curvatures between patches and differences in orientation. While there might be other ways to construct a super-kernel that expresses these properties, LPD super-kernels do have an important property, which is given by the following theorem:

Theorem 1. *A LPD super-kernel G is positive semi-definite and its operator norm satisfies $\|G\| \leq 1$.*

The patch-to-tensor embedding that is achieved by the LPD super-kernel is defined by the spectral analysis of this super-kernel. Therefore, the spectral properties of this super-kernel, which are shown in Theorem 1, are crucial for the patch-based data analysis that utilizes this embedding.

Theorem 1 is in fact a corollary of Theorem 3.1 from [6]. This theorem deals with general linear-projection super-kernels that are defined by arbitrary scalar affinities (instead of the diffusion affinities) and the linear-projection tangent similarity matrices. The theorem shows that the spectrum of any linear-projection super-kernel is non-negative and is bound from above by the spectral norm of the used scalar affinities. In our case, the spectral norm of the used diffusion affinity kernel is one, and thus we get the result in Theorem 1.

3.2 Embedded Distances

The classical diffusion map provides an embedded space in which the Euclidean distance between data points is equal to a diffusion distance in the original ambient space. This diffusion distance measures the distance between two diffusion "bumps" $a(x,\cdot)$ and $a(y,\cdot)$, each of which is a row in the symmetric diffusion kernel that defines the diffusion map. From a technical point of view, this relation means that the Euclidean distance between two arbitrary points in the range of a diffusion map is equal to the Euclidean distances between the corresponding rows of its symmetric diffusion kernel. The following theorem shows a similar property of the LPD-based patch-to-tensor embedding:

Theorem 2. *Let $x, y \in M$ be two points on the manifold and let $\mathcal{T}_x$ and $\mathcal{T}_y$ be their embedded tensors (Eq. 2.2), then $\|\mathcal{T}_x - \mathcal{T}_y\|_F^2 = \sum_{z \in M} \sum_{j=1}^{d} \|(a(x,z)O_x^T - a(y,z)O_y^T)o_z^j\|^2$, where the tensors are treated as matrices (i.e., their coordinate matrices) when computing the Frobenius distance between them.*

Proof. First, we use the definition of the Frobenius norm and the construction of the embedded tensor space to get $\|\mathcal{T}_x - \mathcal{T}_y\|_F^2 = \sum_{i=1}^{l} \sum_{j=1}^{d} |\lambda_i \varphi_i^j(x) - \lambda_i \varphi_i^j(y)|^2 =$

$\sum_{j=1}^{d} \|\Phi(o_x^j) - \Phi(o_y^j)\|^2$. Next, we combine this result with Lemma 4.1 from [6] to get $\sum_{j=1}^{d} \|\Phi(o_x^j) - \Phi(o_y^j)\|^2 = \sum_{j=1}^{d} \|g(o_x^j, \cdot) - g(o_y^j, \cdot)\|^2 = \sum_{z\in M} \sum_{j=1}^{d} \sum_{\xi=1}^{d} |a(x,z)[O_{xz}]_{j\xi} - a(y,z)[O_{yz}]_{j\xi}|^2 = \sum_{z\in M} \|a(x,z)O_{xz} - a(y,z)O_{yz}\|_F^2$, and using the definition of the tangent similarity matrices we get the result in the theorem. □

The vectors o_z^j in Theorem 2 are unit vectors that form an orthonormal basis of the tangent space $T_x(\mathcal{M})$ at the point $z \in M$. For each point $z \in M$, the matrix $[a(x,z)O_x^T - a(y,z)O_y^T]$ is applied to each of these unit vectors and the squared lengths of the resulting vectors are summed. These terms can be seen as extensions of the terms $(a(x,z) - a(y,z))$ of the original diffusion distance, which only consider the differences between scalar affinities. Further explanations about the meaning of the extended diffusion distance can be found in [6].

3.3 Linear-Projection Diffusion Process and Infinitesimal Generator

The diffusion affinity A is a symmetric conjugate of a diffusion operator P. When these operators are considered for a finite dataset M they become finite matrices, but they can also be analyzed as continuous operators. The diffusion operator P defines a stochastic diffusion process over the manifold (or the sampled dataset). A similar interpretation can be stated for the LPD super-kernel G. When all the points on the manifold are considered (instead of a finite dataset) the super-kernel matrix becomes an operator. This super-kernel operator G can be regarded as a symmetric conjugate of a vector-transition operator $\bar{G}$ (i.e., it defines transitions of tangent vectors) whose blocks are defined as $\bar{G}_{xy} = p(x,y)O_x^T O_y$ for every $x, y \in \mathcal{M}$. Let $\boldsymbol{\nu} : \mathcal{M} \to \mathbb{R}^d$ be a tangent vector field expressed by the local coordinates of the tangent spaces of the manifold $\mathcal{M}$, then $\bar{G}\boldsymbol{\nu}(x) = \int_{y\in\mathcal{M}} \bar{G}_{xy}\boldsymbol{\nu}(y)dy$ for every $x \in \mathcal{M}$. We call the transition operator $\bar{G}$ the LPD operator, since it defines a stochastic linear-projection diffusion process of vectors (or vector fields). A detailed description of this process can be found in [10]. The stochastic steps (or "jumps") that are performed by the LPD process are illustrated in Fig. 3.1.

In the scalar case, the infinitesimal generator of the diffusion operator can be expressed by Laplace operators (specifically, the graph Laplacian and the Laplace-Beltrami operator on manifolds). Theorem 3 provides a similar expression for the infinitesimal generator of the LPD operator using the vector-Laplacian, which extends the Laplacian from scalar functions to vector fields.

Theorem 3. *Let G be the LPD operator with the infinitesimal generator $\mathcal{L}(\bar{G})$. Let $\boldsymbol{\nu}$ be a tangent vector field expressed by the local coordinates of the tangent spaces of the manifold $\mathcal{M}$. Then, $\mathcal{L}(\bar{G})\boldsymbol{\nu}(x) = \bar{\Delta}(\mathrm{proj}_x \boldsymbol{\nu})(x)$ for every $x \in \mathcal{M}$, where the operator proj_x projects a vector field on the tangent space $T_x(\mathcal{M})$, and $\bar{\Delta}$ is the vector-Laplacian on this tangent space.*

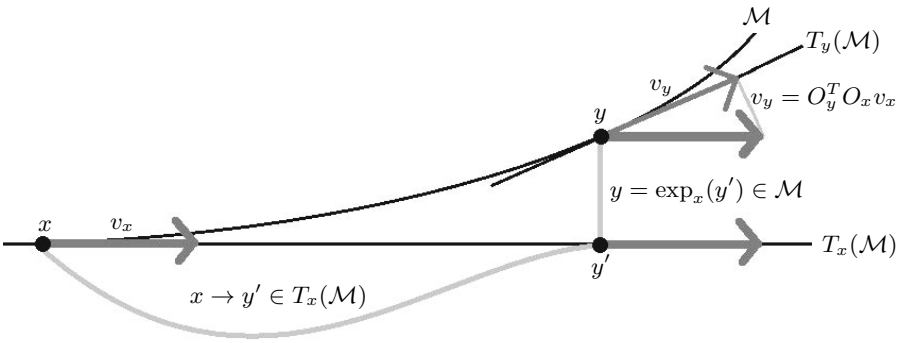


Fig. 3.1. The "jump" of the LPD discrete process. The jump starts with a tangent vector $v_x \in T_x(\mathcal{M})$ at $x \in \mathcal{M}$. First, a point $y' \in T_x(\mathcal{M})$ is chosen according to the diffusion transition probabilities. Then, the exponential map is used to translate this point to a point $y \in \mathcal{M}$ on the manifold. Finally, the vector $v_x \in T_x(\mathcal{M})$ is projected to $v_y \in T_y(\mathcal{M})$ at $y \in \mathcal{M}$.

Theorem 3 is proved by utilizing the relation shown in Theorem 4.2 from [10] between the original diffusion operator and the LPD operator. The detailed proof can be found in [10], where this theorem (i.e., Theorem 3) is presented as Corollary 4.3.

4 Data Analysis Using Patch-to-Tensor Embedding

The LPD and the resulting patch-to-tensor embedding provide a general framework that can be utilized in a wide collection of data analysis tasks such as clustering, classification, anomaly detection and related manifold learning tasks. In this section, we demonstrate the application of the PTE method to two data analysis challenges: 1. Classification of breast tissue impedance measurements. 2. Data clustering that is based on image segmentation. We use Algorithm 1 to compute and construct the embedding for the analysis.

4.1 Electrical Impedance Breast Tissue Classification

Biological tissues have complex electrical impedance related to the tissue dimension, the internal structure and the arrangement of the constituent cells. Therefore, the electrical impedance can provide useful information based on heterogeneous tissue structures, physiological states and functions. Recently, an interesting dataset of breast tissue impedance measurements was published [3]. The dataset consisted of 106 spectra recorded in samples of breast tissue from 64

Algorithm 1. Patch-to-Tensor Embedding Construction (PTEC)

Input: Data points: $x_1, ..., x_n \in R^m$ and parameters: Patch size ρ and ℓ
Output:

1: For each $x \in M$ estimate an orthonormal basis $O_x \in R^{m \times d}$ of its tangent space based on ρ points uniformly distributed over a small neighborhood of x;
2: Construct the diffusion affinity kernel A;
3: Construct the LPD super-kernel G using the kernel A and the matrices O_x, $x \in M$;
4: Construct the spectral map $\Phi(o_x^j)$ for $j = 1, \ldots, d$ by utilizing the SVD of the constructed LPD super-kernel G;
5: Construct a tensor $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$ for each $x \in M$ using the spectral map Φ.

patients undergoing breast surgery. Each spectrum consisted of twelve impedance measurements taken at different frequencies ranging from 488 Hz to 1 MHz. Detailed description of the data collection procedure as well as classification of the cases and frequencies used are given in [4,7].

We follow the foot steps of [7] in classifying post-processing attributes (see [7] for detailed explanation of these attributes) into the same tissue categories using PTE. Initially, the given dataset was normalized to have zero mean and a unit standard deviation for each attribute. Then, the PTE construction, detailed in Algorithm 1, was used to construct the LPD super kernel followed by embedding of the measurements into a tensor space. The ε diffusion meta-parameter was chosen as the mean Euclidean distance between all the pairs of data points in the given dataset. The parameters in the PTE construction were $\ell = 5$ and $\rho = 66$. They were chosen in an exhaustive search to optimize the classification accuracy.

The classification performance is based on a leave-one-out methodology in which each of the measurements was labeled according to its nearest neighbor in the embedded tensor space. The Frobenius norm was used as the distance metric. The classification performance is described in Table 4.1.

The achieved classification performances, which were obtained by PTE with a single classification stage, are competitive to the ones in [7]. The optimization of the classifier was done only with respect to two parameters: ρ the number of points per patch and ℓ the number of eigenvectors from the application of the SVD procedure.

Table 4.1. Performance summery of the PTE-based classification algorithm

Tissue category	Correct detection	False detection	Miss-detection
Fatty	97.2%	0	2.7%.
Carcinoma	86.36%	13.6%	9.5%
FMG	93.9%	6.1%	6.1%

4.2 Image Segmentation

Image segmentation clusters pixels into image regions corresponding to individual surfaces, objects, or natural parts of objects. It plays a key role in many computer vision tasks such as object recognition, image compression, image editing and image retrieval. It has been extensively studied in computer vision and statistics with a vast number of different algorithms and approaches. The PTE framework enables to view the image via a LPD super-kernel that reflects the affinities between pixels and the projection of the related tangent spaces. This construction translates the given pixel-related features into tensors in the embedded space. The image segmentation into similar sets is achieved by clustering the tensors in the embedded space.

For our image segmentation examples, we utilized pixel color information and its spatial (x,y) location multiplied by scaling factor $w = 0.1$. Hence, given an RGB image with $I_x \times I_y$ pixels, we generated a $5 \times (I_x \cdot I_y)$ dataset X.

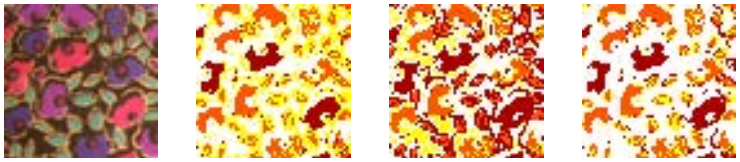
Algorithm 1 embeds X into a tensor space. The first step in Algorithm 1 constructs local patches. Each generated patch captures the relevant neighborhood and considers both color similarity and spatial similarity. Hence, a patch is more likely to include attributes related to spatially close pixels. The ε diffusion meta-parameter in this case equals the mean Euclidean distance between all the pairs in X. The PTE parameters ℓ and ρ were chosen to generate the most homogenous segments. The k-means algorithm with "sum of square differences" was used to cluster the tensors into similar sets.

Figures 4.1 and 4.2 present the segmentation results from the application of the PTE algorithm, where for each figure, (a) is the original image. All of the images are of size 60×60. Each figure describes the segmentation result at several

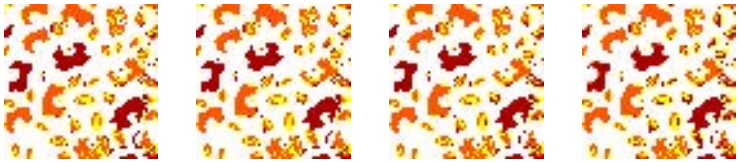
(a) Original image (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

(e) $t = 4$ (f) $t = 5$ (g) $t = 6$ (h) $t = 7$

Fig. 4.1. The PTE segmentation results for the image 'Cubes' when $\ell = 10$ and $d = 10$. The results are shown at several diffusion times t.



(a) Original image (b) $t = 1$ (c) $t = 2$ (d) $t = 3$



(e) $t = 4$ (f) $t = 5$ (g) $t = 6$ (h) $t = 7$

Fig. 4.2. The PTE segmentation results for the image 'Fabric' when $\ell = 10$ and $d = 10$. The results are shown at several diffusion times t.

diffusion times t. The impact of the diffusion time on the segmentation quality is significant in both cases. For example, as can be seen in Fig. 4.2, the first two images (Fig. 4.2(b) and Fig. 4.2(c)), which correspond to $t = 1$ and $t = 2$ respectively, show poor segmentation qualities. As t increases, the segmentation becomes more homogeneous and the main structures in the original image can be separated as we see, for example, in 4.2(e) where $t = 4$. Another interesting aspect related to the diffusion time parameter t is the smoothing effect it has, when it increases, on the pairwise distances between data points in the embedded space. By increasing t, the pairwise distances between similar tensors decrease while the distances between dissimilar tensors increase. In the segmentation case, the result will be pixel-label change. For example, Fig. 4.1 presents the 'Cubes' image segmentation as a function of $t = 1, 2, \ldots, 7$. The rightmost cube in the segmented images becomes more homogeneous as t increases.

5 Conclusion

In this paper, we presented an extension of the scalar-affinity kernels that are used in kernel methods. We used a linear-projection diffusion process to construct this extension, which we call a *super-kernel*. We briefly described important theoretical properties of the LPD super-kernel and its underlying linear-projection diffusion process. Future works will present the utilization of this innovative patch-based approach together with a complete patch-processing data-mining framework that combines coarse-graining, dictionary-based subsampling, dimensionality reduction and smooth interpolation techniques.

Among other benefits, the patch-processing approach introduced here will enable the reduction of wide redundancies in many large-scale datasets. It provides a

meaningful representation of the essential intelligence from the analyzed data without any superfluous information that does not benefit the sought-after patterns and can thus be regarded as noise from the analysis point of view.

Acknowledgements. This research was supported by the Israel Science Foundation (Grant No. 1041/10), the Eshkol Fellowship from the Israeli Ministry of Science & Technology and Agora Center, University of Jyväskylä, Finland.

References

1. Coifman, R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis 21(1), 5–30 (2006)
2. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall, London (1994)
3. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
4. Jossinet, J.: Variability of impedivity in normal and pathological breast tissue. Medical and Biological Engineering and Computing 34, 346–350 (1996)
5. Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29, 1–27 (1964)
6. Salhov, M., Wolf, G., Averbuch, A.: Patch-to-tensor embedding. Applied and Computational Harmonic Analysis 33(2), 182–203 (2012)
7. da Silva, J.E., de Sá, J.M., Jossinet, J.: Classification of breast tissue by electrical impedance spectroscopy. Medical and Biological Engineering and Computing 38, 26–30 (2000)
8. Singer, A., Wu, H.: Orientability and diffusion maps. Applied and Computational Harmonic Analysis 31(1), 44–58 (2011)
9. Singer, A., Wu, H.: Vector diffusion maps and the connection laplacian. Communications on Pure and Applied Mathematics 65(8), 1067–1144 (2012)
10. Wolf, G., Averbuch, A.: Linear-projection diffusion on smooth Euclidean submanifolds. Applied and Computational Harmonic Analysis (2012), doi:10.1016/j.acha.2012.03.003

Dictionary Construction for Patch-to-Tensor Embedding

Moshe Salhov[1], Guy Wolf[1], Amit Bermanis[2], Amir Averbuch[1], and Pekka Neittaanmäki[3]

[1] School of Computer Science,
Tel Aviv University
Tel Aviv 69978, Israel
[2] School of Mathematical Sciences,
Tel Aviv University
Tel Aviv 69978, Israel
[3] Faculty of Information Technology,
University of Jyväskylä
Jyväskylä, Finland

Abstract. The incorporation of matrix relation, which can encompass multidimensional similarities between local neighborhoods of points in the manifold, can improve kernel based data analysis. However, the utilization of multidimensional similarities results in a larger kernel and hence the computational cost of the corresponding spectral decomposition increases dramatically. In this paper, we propose dictionary construction to approximate the kernel in this case and its respected embedding. The proposed dictionary construction is demonstrated on a relevant example of a super kernel that is based on the utilization of the diffusion maps kernel together with linear-projection operators between tangent spaces of the manifold.

1 Introduction

Recent methods for advanced high dimensional data analysis utilize a manifold structure on which data-points are assumed to lie, immersed (or submersed) in an ambient space that is defined by observable parameters. Kernel methods such as k-PCA and Diffusion Maps (DM) [2] have shown to provide good results in analyzing such high dimensional data sets. The defined a kernel can be thought of as an adjacency matrix of a graph whose vertices are the data points in the dataset. The analysis of the eigenvalues and the corresponding eigenvectors of this matrix reveals many properties and connections in the graph. These methods are based on the spectral decomposition of a kernel that was designed to incorporate a scalar similarity measure between data points. The resulted embedding of the data-points into an Euclidean space preserves the qualities represented by the designed kernel. This approach extends the essence of the classical Multi-Dimensional Scaling (MDS) method [3,5] by considering non-linear relations, instead of just the linear one in its original Gram matrix.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 346–356, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Recently, the DM method was extended in several different ways to consider orientation of local tangent spaces [7,8,6,9]. The relation between two patches is described by a matrix rather than by a scalar value. The resulting kernel captures more similarities about the local structure of the underlying manifold while increasing considerably the kernel size.

Kernel size is a limiting factor in the applicability of spectral decomposition based data analysis methods and considerable effort has been invested in approximating the spectral decomposition operator for example [4,1] and many more related variants. The dictionary approach presented in [4] construct a dictionary and the corresponded scalar kernel plus the necessary extension coefficients for approximating the full scalar kernel. The number of dictionary members depend on the given dataset, kernel configuration and a designed parameter for controlling the quality of the full kernel approximation.

In this paper we leverage this dictionary construction approach [4] for approximating the spectral decomposition of a non-scalar kernel. We describe the resulting condition for updating the non-scalar dictionary and a respected bound on the approximation error. Although the proposed method is applicable for many such kernels we focus on a linear projection super kernel construction described in [6]. The super kernel construction aim at analyzing patches of the manifold instead of analyzing single points on the manifold. Each patch is defined as a local neighborhood of a point in a dataset sampled from an underlying manifold. The relation between two patches is described by a matrix which represents both the affinity between the points at the centers of these patches and the similarity between their local coordinate systems. The constructed matrices between all patches are then combined in a block matrix, which is call a super-kernel.

The paper has the following structure: Brief preliminaries are presented in Section 2. Section 3 formulates the discussed problem. A dictionary construction is introduced in Section 4. Finally, Section 5 shows experimental results from the utilization of the dictionary-based analysis for image segmentation.

2 Preliminaries

2.1 Manifold Setup

Let $M \subseteq \mathbb{R}^m$ be a set of n points sampled from a manifold $\mathcal{M}$ that lies in the ambient space $\mathbb{R}^m$. Let $d \ll m$ be the intrinsic dimension of $\mathcal{M}$, thus, it has a d-dimensional tangent space $T_x(\mathcal{M})$, which is a subspace of $\mathbb{R}^m$, at every point $x \in M$. If the manifold is densely sampled, the tangent space $T_x(\mathcal{M})$ can be approximated by a small enough patch (i.e., neighborhood) $N(x) \subseteq M$ around $x \in M$.

Let $o_x^1, \ldots, o_x^d \in \mathbb{R}^m$, where $o_x^i = (o_x^{i1}, \ldots, o_x^{im})^T,\ i = 1, \ldots, d$, form an orthonormal basis of $T_x(\mathcal{M})$ and let $O_x \in \mathbb{R}^{m \times d}$ be a matrix whose columns are these vectors:

$$O_x \triangleq \begin{pmatrix} | & & | & & | \\ o_x^1 & \cdots & o_x^i & \cdots & o_x^d \\ | & & | & & | \end{pmatrix} \qquad x \in M\ . \tag{2.1}$$

We will assume from now on that vectors in $T_x(\mathcal{M})$ are expressed by their d coordinates according to the presented basis $o_x^1, \ldots, o_x^d$. For each vector $u \in T_x(\mathcal{M})$, the vector $\tilde{u} = O_x u \in \mathbb{R}^m$ is the same vector as u represented by m coordinates, according to the basis of the ambient space. For each vector $v \in \mathbb{R}^m$ in the ambient space, the vector $v' = O_x^T v \in T_x(\mathcal{M})$ is the linear projection of v on the tangent space $T_x(\mathcal{M})$.

2.2 Diffusion Maps

The original diffusion maps method [2] can be used to analyze the dataset M by exploring the geometry of the manifold $\mathcal{M}$ from which it is sampled. This method is based on defining an isotropic kernel $K \in \mathbb{R}^{n \times n}$, whose elements are defined as $k(x,y) \triangleq e^{-\frac{\|x-y\|}{\varepsilon}}$, $x, y \in M$, where ε is a meta-parameter of the algorithm. This kernel represents the affinities between points on the manifold. The kernel can be viewed as a construction of a weighted graph over the dataset M. The points in M are used as vertices and the weights of the edges are defined by the kernel K. The degree of each point (i.e., vertex) $x \in M$ in this graph is $q(x) \triangleq \sum_{y \in M} k(x,y)$. Kernel normalization with this degree produces a $n \times n$ row stochastic transition matrix P whose elements are $p(x,y) = k(x,y)/q(x)$ for $x, y \in M$, which defines a Markov process (i.e., a diffusion process) over the points in M. A symmetric conjugate $\bar{P}$ of the transition operator P defines the diffusion affinities between points as

$$\bar{p}(x,y) = \frac{k(x,y)}{\sqrt{q(x)q(y)}} = \sqrt{q(x)}p(x,y)\frac{1}{\sqrt{q(y)}} \qquad x, y \in M. \tag{2.2}$$

The diffusion maps method computes an embedding of data points on the manifold into an Euclidean space whose dimensionality is usually significantly lower than the original data dimensionality. This embedding is a result of spectral analysis of the diffusion affinity kernel $\bar{P}$. The eigenvalues $1 = \sigma_0 \geq \sigma_1 \geq \ldots$ of $\bar{P}$ and their corresponding eigenvectors $\bar{\phi}_0, \bar{\phi}_1, \ldots$ are used to construct the desired map, which embeds each data-point $x \in M$ onto the point $\bar{\Phi}(x) = (\sigma_i \bar{\phi}_i(x))_{i=0}^{\delta}$ for a sufficiently small δ, which is the dimension of the embedded space and depends on the decay of the spectrum of $\bar{P}$.

2.3 Linear-Projection Super-kernel

For $x, y \in M$, let $O_{xy} = O_x^T O_y \in \mathbb{R}^{d \times d}$, where O_x and O_y were defined in Eq. 2.1. The matrices O_x and O_y represent bases of the tangent spaces $T_x(\mathcal{M})$ and $T_y(\mathcal{M})$, respectively. Thus, the matrix O_{xy} represents a linear-projection between these tangent spaces, and, in some sense, the similarity between them. We following [6] will refer to it as a tangent similarity matrix.

Let $\Omega \in \mathbb{R}^{n \times n}$ be a symmetric and positive semi-definite affinity kernel defined on $M \subseteq \mathbb{R}^m$, i.e., each row or each column in Ω corresponds to a data point in M, and each element in it, $[\Omega]_{xy} = \omega(x,y)$, $x, y \in M$, represents an affinity

between x and y. Also, assume that $\omega(x,y) \geq 0$ for every $x, y \in M$. Notice that the diffusion affinity kernel is an example of such an affinity kernel. Definition 1 uses the tangent similarity matrices and the affinity kernel Ω to define the Linear-Projection *super-kernel*. When the diffusion affinities in $\bar{P}$ are used, instead of the general affinities on Ω, this super-kernels is called a Linear-Projection Diffusion (LPD)super-kernel.

Definition 1 (Linear-Projection Super-kernel). *A Linear-Projection (LP) super-kernel is a matrix $G \in \mathbb{R}^{nd \times nd}$ where in terms of blocks, it is a block matrix of size $n \times n$ and each block in it is a $d \times d$ matrix. Each row and each column of blocks in G corresponds to a point in M, and a single block G_{xy} (where $x, y \in M$) represents an affinity or similarity between the patches $N(x)$ and $N(y)$. Each block $G_{xy} \in \mathbb{R}^{d \times d}$ of G is defined as $G_{xy} \triangleq \omega(x,y)O_{xy} = a(x,y)O_x^T O_y,\ x, y \in M$.*

The super-kernel in Definition 1 encompasses both the affinities between points on the manifold $\mathcal{M}$ and the similarities between their tangent spaces. The latter are expressed by linear-projection operators to between tangent spaces. Specifically, for two tangent spaces $T_x(\mathcal{M}), T_y(\mathcal{M})$ of the manifold at $x, y \in M$, the operator $O_x^T O_y$ (i.e., their tangent similarity matrix) expresses a linear projection from $T_y(\mathcal{M})$ to $T_x(\mathcal{M})$ via the ambient space $\mathbb{R}^m$. The obvious extreme cases are an identity matrix, which indicates on complete similarity and a zero matrix, which indicates on orthogonality (i.e. complete dissimilarity). This linear projection operators express some important properties of the manifold structure, e.g., curvatures between patches and differences in orientation. Further details on the properties of LP and LPD super-kernels can be found in [6,9].

It is convenient to use the vectors o_x^i and o_y^j to apply a double-indexing scheme and use the notation $g(o_x^i, o_y^j) \triangleq [G_{xy}]_{ij}$ that considers each single cell in G as an element $[G_{xy}]_{ij}$ in a block G_{xy}, where $x, y \in M$ and $i, j \in \{1, \ldots, d\}$. It is important to note that $g(o_x^i, o_y^j)$ is *only a notation* for convenience reasons and a single element of a block in G does not necessarily have any special meaning. The block itself, as a whole, holds meaningful similarity information.

Spectral decomposition is used to analyze a super-kernel G, and utilize it to embed the patches $N(x)$ of the manifold (for $x \in M$) into a tensor space. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_\ell|$ be the ℓ most significant eigenvalues of G and let $\phi_1, \phi_2, \ldots, \phi_\ell$ be their corresponding eigenvectors. Each eigenvector ϕ_i, $i = 1, \ldots, \ell$ is a vector of length nd. We denote each of its elements as $\phi_i(o_x^j)$ where $x \in M$ and $j = 1, \ldots, d$. An eigenvector ϕ_i can also be regarded as a vector of n sections, each of which is a vector of length d that corresponds to a point $x \in M$ on the manifold. To express this notion we use the notation $\varphi_i^j(x) = \phi_i(o_x^j)$. Thus, the section that corresponds to $x \in M$ in ϕ_i is the vector $(\varphi_i^1(x), \ldots, \varphi_i^d(x))^T$.

The eigenvalues and eigenvectors of G are used to construct a spectral map $\Phi(o_x^j) = (\lambda_1\phi_1(o_x^j), \ldots, \lambda_\ell\phi_\ell(o_x^j)$, which is similar to the standard (i.e., classic)

diffusion map. This spectral map is then used to construct the embedded tensor $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$ for each $x \in M$, which is represented by the following $\ell \times d$ matrix:

$$\mathcal{T}_x \triangleq \begin{pmatrix} | & & | \\ \Phi(o_x^1) & \cdots & \Phi(o_x^d) \\ | & & | \end{pmatrix} \qquad x \in M \ . \tag{2.3}$$

In other words, the coordinates of $\mathcal{T}_x$ (i.e., the elements in this matrix) are $[\mathcal{T}_x]_{ij} = \lambda_i^\mu \varphi_i^j(x)$, $i = 1, \ldots, \ell, j = 1, \ldots, d$. Each tensor $\mathcal{T}_x$ represents an embedding of the patch $N(x)$, $x \in M$, into the tensor space $\mathbb{R}^\ell \otimes \mathbb{R}^d$.

3 Problem Formulation

We consider the problem of estimating a tangential vector field where we assume access to a set $Z^t = \{(x_1, F(x_1)), ..., (x_t, F(x_t))\}$ of input pairs $x_i \in M$ and corresponding tangent vectors $\boldsymbol{\nu}(x_i) \in \mathbb{R}^d$ (expressed in local coordinates). Given a training set Z^{t-1} of $t-1$ input pairs and a new input data-point x_t, we define an optimal predictor to estimate $\boldsymbol{\nu}(x_t)$ as

$$\hat{\boldsymbol{\nu}}(x_t) = \sum_{i=1}^{t-1} G_{ti} \boldsymbol{w}_i, \tag{3.1}$$

where G_{ti} is the block $G_{x_t x_i}$ of the LP super-kernel, and the vectors $\boldsymbol{w}_i$ are suitable tangent vectors. Lemma 3.3 from [6] suggests the above predictor can be defined in terms of the embedded tensors instead of the super-kernel blocks as

$$\hat{\boldsymbol{\nu}}^T(x_t) = \sum_{i=1}^{t-1} \boldsymbol{w}_i^T \mathcal{T}_{x_i}^T \mathcal{T}_{x_t}, \tag{3.2}$$

A solution to the above prediction/estimation problem can be designed to consider all the given input pairs Z^{t-1}, however the resulting optimization problem complexity will grow as the dataset size will grow. Let $\mathcal{T}_{x_j}$ be the embedded tensor of $x_j \in M$. If we can estimate this tensor given a subset of embedded tensors as

$$\mathcal{T}_{x_j} = \sum_{i=1}^{t-1} A_i \mathcal{T}_{x_i}^T, \tag{3.3}$$

where A_i $(i = 1, \ldots, t-1)$ are matrices of suitable sizes and $A_j = 0$, then forcing $\boldsymbol{w}_j = 0$ in Eq. 3.2 will not degrade the performance of the estimator, since $\mathcal{T}_{x_j}$ can be represented by other embedded tensors. In [4], it is suggested to relax the requirement in Eq. 3.3 by allowing a controlled approximation error. The resulting linear dependency test is the Approximate Linear Dependency (ALD) test. The ALD dictates the construction of the dictionary. The dictionary is initialized with a single embedded tensor. Given a new embedded tensor candidate for the dictionary we consider two cases. In the first case the new embedded tensor is approximately dependent on the existing dictionary. In this case the

estimator will only consider the new embedded tensor through its approximating tensors. In the second case the, the new embedded tensor is not well approximated by the dictionary members. In this case the new embedded tensor is added to the dictionary and a corresponding matrix will be added to the estimator. In the following sections we detail several constructions of the dictionary based on the extension of the ALD principle.

4 Dictionary Construction

4.1 Sparsification Procedure

The prediction/estimation design assumes that we are sequentially given input pairs. We further assume that a dictionary $D_{t-1} = \{y_j\}_{j=1}^{\eta_{t-1}}$ was formed based on $t-1$ input pairs in Z_{t-1}. By construction, the embedded tensors $\{\mathcal{T}_{y_j}\}_{j=1}^{\eta_{t-1}}$ of the points in the dictionary are approximately linearly-independent. Then, a new input $x_t \notin Z_{t-1}$ is examined as a candidate to enter the dictionary. The new candidate is added to the dictionary if the minimizer $A = [A_1, \ldots, A_{\eta_{t-1}}]$ (where $A_i \in \mathbb{R}^{d\times d}, i = 1, \ldots, \eta_{t-1}$ and $A \in \mathbb{R}^{d\times d\eta_{t-1}}$) satisfies the tensor ALD condition

$$\delta_t \triangleq \min_A \| \sum_{i=1}^{t-1} A_i^T \mathcal{T}_{x_i}^T - \mathcal{T}_{x_t}^T \|_F^2 \leq \mu, \tag{4.1}$$

where μ is an accuracy parameter same as the one used for the KRLS [4]. Lemma 1 shows that the error term δ_t can be expressed in terms of the super-kernel G. According to this lemma, and its proof, the corresponding approximation in terms of the super-kernel matrix is $G \approx E_t G_t E_t^T$ where $E_t \in \mathbb{R}^{td\times d\eta_t}$ is the matrix that aggregate the optimal matrices A_t for all the relevant input pairs and G_t is the super kernel the corresponds to the dictionary members.

Lemma 1. *Let δ_t be defined in Eq. 4.1 and H_t be a block matrix with ith subblock G_{it}, then $\delta_t = \mathrm{tr}[G_{tt} - H_t^T G_{t-1}^{-1} H_t]$.*

Proof. The minimizer in Eq. 4.1 can be found in a closed form by opening the brackets, thus we get $\delta_t = \min_A\{\, tr[(\sum_{i=1}^{t-1} A_i^T \mathcal{T}_{x_i}^T - \mathcal{T}_{x_t}^T)(\sum_{i=1}^{t-1} A_i^T \mathcal{T}_{x_i}^T - \mathcal{T}_{x_t})^T]\}$. Further simplification yields $\delta_t = \min_A\{\mathrm{tr}[(\sum_{i=1}^{t-1} A_i^T \mathcal{T}_{x_i}^T)(\sum_{i=1}^{t-1} \mathcal{T}_{x_i} A_i)^T + \mathcal{T}_{x_t}^T \mathcal{T}_{x_t} - \mathcal{T}_{x_t}^T (\sum_{i=1}^{t-1} \mathcal{T}_{x_i} A_i) - (\sum_{i=1}^{t-1} A_i^T \mathcal{T}_{x_i}^T)\mathcal{T}_{x_t}]\}$. The products of the form $\mathcal{T}_{x_j}^T \mathcal{T}_{x_i}$ can be replaced with the super-kernel over pairs of embedded tensors. We substitute $\mathcal{T}_{x_j}^T \mathcal{T}_{x_i} = G_{ij}$,and get $\delta_t = \min_A \mathrm{tr}(A^T G_{t-1} A - A^T H_t - H_t^T A + G_{tt})$, where G_{t-1} is the super-kernel after processing $t-1$ input pairs, H_t is a matrix size $d \times t-1$ with $G_{i,t}$ as its i-th block and G_{tt} is the tth subblock on the diagonal of G with size $d \times d$. Solving the above optimization yields $A_t = G_{t-1}^{-1} H_t(x_t)$ and the respected approximation error in the lemma. □

Let $\Psi = [\mathcal{T}_{x_1}, ..., \mathcal{T}_{x_l}]$ be the $l \times nd$ matrix that aggregates all the embedded tensors. Furthermore, let $\hat{\Psi} = [\mathcal{T}_{y_1}, ..., \mathcal{T}_{y_{\eta_{t-1}}}]$ be the dictionary-related embedded tensors and let $\Psi^{\mathrm{res}} = [\psi_1^a, \ldots, \psi_n^a]$ be a matrix that contains estimation errors

of the tensors. Note that each of these errors is bounded by μ. The relation between the estimated tensors and the actual ones is given by $\Psi = E_t\hat{\Psi} + \Psi^{\text{res}}$. Multiplying the last equation with its transpose yield, $G = E_t G_t E_t^T + (\Psi^{\text{res}})^T\Psi^{\text{res}}$ where all the cross terms vanish by the optimality of each A_t. Let $R = (\Psi^{\text{res}})^T\Psi^{\text{res}}$, then from the definition of δ_t the trace of the j-th block on the diagonal of R equals δ_j. Hence, we can bound the l_2 norm of R as $\|R\|_2 \leq \mu n$.

4.2 The LP Super Kernel Dictionary Construction

The dictionary can be utilized to estimate a tangential vector field using a recursive algorithm similar to the well known Recursive Least Squares (RLS) used in [4]. In the supervised learning scheme the predictor Eq. 3.1 is designed to minimize the l_2 distance between the predicted vector-field at time-step t and the actual given vector field (as part of the training set) as

$$J(\boldsymbol{w}) = \sum_{i=1}^{t} \|\hat{\nu}(x_i) - \nu(x_i)\|_2^2 = \sum_{i=1}^{t} \left\| \sum_{j=1}^{t} G_{ij}\boldsymbol{w}_j - \nu(x_i) \right\|_2^2 = \|G\boldsymbol{w} - \boldsymbol{\nu}\|_2^2, \quad (4.2)$$

where $\boldsymbol{w}$ is the predictor weights and $\boldsymbol{\nu}$ is a concatenation of all the given training values of the vector field[1]. The Least Squares solution to the minimization of $J(\boldsymbol{w})$ is given by $\boldsymbol{w}_o = G^{\dagger}\boldsymbol{\nu}$, where † is the pseudo inverse operator. In the case when the number of vector examples is large the complexity of inverting the LP super kernel tends to be formidable in terms of computational cost and memory requirements. Furthermore, the length of the predictor weight vector $\boldsymbol{w}_o$ depends on the number of training samples. Hence, redundant samples, which generate linearly-dependent rows in the super-kernel, will cause over-fitting of the predictor. One possible remedy for these problems is to utilize the sparsification procedure from Section 4.1 in order to design the predictor. The optimizer $\boldsymbol{w}_o$ can be formulated by introducing the dictionary-estimated super-kernel as $J(\boldsymbol{w}) = \|G\boldsymbol{w} - \boldsymbol{\nu}\|_2^2 \approx \|E_t G_t E_t^T w - f\|_2^2$. Let $\alpha = E_t^T w$, then the predictor is reformulated as $\hat{\boldsymbol{\nu}}(x_i) = \sum_{j=1}^{t} A_j G_{ji}\alpha_j$, and the corresponding predictor's l_2 error is given by $J(\alpha) = \|E_t G_t E_t^T w - f\| = \|E_t G_t \alpha - \boldsymbol{\nu}\|$, which can be minimized by $\alpha_o = (E_t G_t)^{\dagger} f = G_t^{-1}(E_t^T E_t)^{-1} f$. Now the predictor coefficients α_o can be computed using the dictionary-based LP super-kernel G_t and the corresponding extension matrix E_t.

The construction of both G_t and E_t is as follows. At each time-step t, the optimal extension matrix $A_t = G_{t-1}^{-1} H_t(x_t)$ is used to compute δ_t and derive two possible scenarios:

1. $\delta_t \leq \nu$**,** hence $\mathcal{T}_{x_t}$ is ALD on D_{t-1} and E_t must be updated. The dictionary set does not need to be updated i.e., $D_t = D_{t-1}$ and $G_t = G_{t-1}$.

2. $\delta_t > \nu$**,** hence $\mathcal{T}_{x_t}$ is **not** ALD on D_{t-1}. The vector x_t and its corresponding tangent basis O_{x_t} are added to the dictionary. The minimizer A_t is trivial and G_t plus E_t must be updated accordingly.

[1] We use this slight abuse of notations (namely, using the same notation for the vector field and the aggregation of its known values) for the sake of simplicity.

In the first case the dictionary is not changed, however the extension matrix must be updated to $E_t = [E_{t-1}, A_t]$. In the second case, the dictionary is updated and thus the related super-kernel is also updated. The updated super-kernel is given by

$$G_t = \begin{bmatrix} G_{t-1} & H_{t_1} \\ H_{t_1}^T & G_{tt} \end{bmatrix}. \tag{4.3}$$

Block matrix inversion can be utilized to find $(G_t)^{-1}$:

$$G_t^{-1} = \begin{bmatrix} G_{t-1}^{-1} + A_t \Delta^{-1} A_t^T & -A_t \Delta^{-1} \\ -\Delta^{-1} A_t^T & \Delta^{-1} \end{bmatrix}, \tag{4.4}$$

where $\Delta = G_{tt} - H_t^T G_{t-1}^{-1} H_t$ is computed through the ALD test. Furthermore, the extension coefficients must describe the extension of the new element in the dictionary as $E_t = [E_{t-1}, A_t]$. Given the QR decomposition of the extension matrix $E_t = Q_E R_E$ and the dictionary related super kernel G_t the first $\eta_t d$ eigenvectors of the full kernel G can be approximated as:

$$U_t = \hat{Q_E} U g, \tag{4.5}$$

where $\hat{Q_E} \in \mathbb{R}^{dt \times d\eta_t}$ is the submatrix os Q_E and the matrix Ug is the eigenvectors that result from the SVD of the matrix product $UgSgVg^T = \hat{R_E} G_t \hat{R_E}^T$ and

Algorithm 1. Patch-to-Tensor Dictionary Construction and Embedding Approximation (PTEA)

Input: Data points: $x_1, ..., x_n \in R^m$ and parameters: Patch size ρ, max approximation tolerance μ and ℓ

1: Initialize: $G_t = G_{11}$, $G_t^{-1} = G_{11}^{-1}$, $E_t = I_d$ and $\eta_t = 1$
2: **for** $t = 2$ *to* n **do**
    Compute $H_t(x_t) = [G_{1t}, ..., G_{\eta_t t}]$
    ALD Test:
    Compute $A_t = G_{t-1}^{-1} H_t(x_t)$
    Compute $\Delta = G_{tt} - H_t(x_t)^T A_t$
    Compute $\delta = Tr(\Delta)$
    if $\delta < \nu$
        $E_t = [E_{t-1}, A_t]$
        $D_t = D_{t-1}$
    else (update dictionary)
        $E_t = \begin{bmatrix} E_t & 0 \\ 0 & I_d \end{bmatrix}$
        $D_t = D_{t-1} \cup \{x_i\}$
        Update G_t according to Eq. 4.3
        Update G_t^{-1} according to Eq. 4.4
        $\eta_t = \eta_{t-1} + 1$
3: Compute the U_t and the approximated spectral map $\Phi(o_x^j)$ for $j = 1, \ldots, d$ according Eq. 4.5
4: Construct a Tensor $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$ for each $x \in M$ given U_t

$\hat{R_E} \in \mathbb{R}^{d\eta_t \times d\eta_t}$ is the respected submatrix of R_E. The main computational cost is the QR decomposition that is estimated to be $O\left(d^3\eta_t^2\left(t-\eta_t/3\right)\right)$. The computational cost is much more smaller compared to the full SVD cost $O\left(d^3t^3\right)$. The dictionary construction followed by patch to tensor embedding approximation process is described in Algorithm 1.

5 Image Segmentation via Dictionary Based PTE

The PTE [6] provide a general frame work that can be utilized to a wide span of data analysis tasks such as clustering, classification, abnormality detection and related manifold learning tasks. In this section we demonstrate the utilization of PTE to image segmentation based on the proposed dictionary construction.

Image segmentation aims to cluster pixels into image regions corresponding to individual surfaces, objects, or natural parts of objects. Image segmentation plays a key rule in many computer vision tasks such as object recognition, image compression, image editing, or image retrieval.

Under the PTE framework the image is viewed as a LPD super kernel constructed to reflect affinities between pixels and the projection of the related tangent spaces. The PTE construction translate the given pixel related features into tensors in the embedded space. The image segmentation is achieved through clustering the tensors in the embedding space into similar groups.

For the image segmentation examples we utilized pixel color information and its spatial (x,y) location multiplied by scaling factor $w = 0.1$. Hence, given an RGB image with $Ix \times Iy$ pixels we generated a $5 \times (Ix \cdot Iy)$ dataset X.

Algorithm 1 is used to construct an embedding of X in tensor space. The first step in Algorithm 1 include the construction of local patches. Each generated

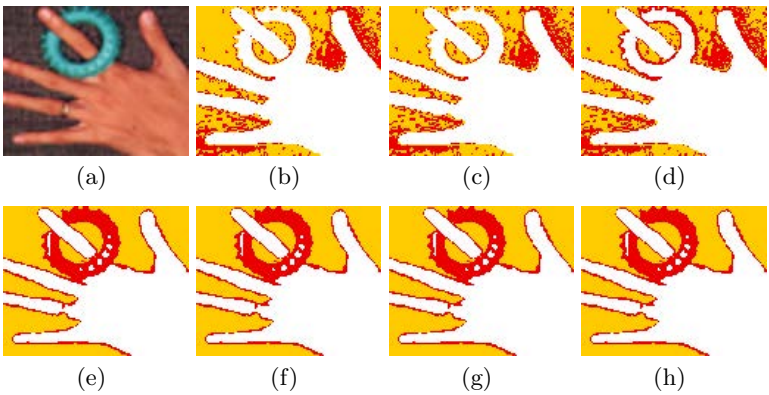
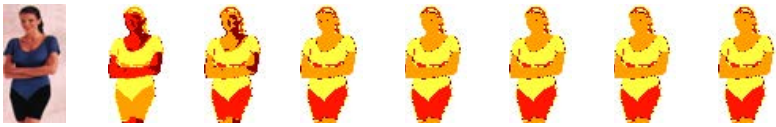


Fig. 5.1. The PTE segmentation result for image 'Hand' with $l = 10$ and $d = 10$, (a) original image, (b) is the segmentation with $t = 1$, (c) is the segmentation with $t = 2$, (d) is the segmentation with $t = 3$, (e) is the segmentation with $t = 4$, (f) is the segmentation with $t = 5$, (g) is the segmentation with $t = 6$, (h) is the segmentation with $t = 7$



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5.2. The PTE segmentation result for image 'Sport' with $l = 10$ and $d = 20$, (a) original image, (b) is the segmentation with $t = 1$, (c) is the segmentation with $t = 2$, (d) is the segmentation with $t = 3$, (e) is the segmentation with $t = 4$, (f) is the segmentation with $t = 5$, (g) is the segmentation with $t = 6$, (h) is the segmentation with $t = 7$

patch captures the relevant neighborhood and considers both color similarity and spatial similarity. Hence, patch are more likely to include attributes related to spatially close pixels. It is important to note that the affinity kernel is computed according to Eq. 2.2 with ε equals the mean Euclidean distance between all pairs in X. The PTE parameters l and ρ were chosen to generate the most homogeneous segments. The dictionary approximation tolerance μ was chosen arbitrary to be small as $\mu = 0.001$. The KMean algorithm with sum of square difference was used to cluster the tensors into similar groups. The final clustering result of the embedded tensors as a function of the diffusion time t are presented in the following figures.

Figures 5.1 - 5.2 present the PTE segmentation results based on the dictionary construction. For each figure, the first image (a) is the original image. The image respected size is detailed in Table 5.1. Each figure describe the segmentation results as a function of the diffusion parameter t. The impact of the diffusion time on the segmentation was significant for the 'Hand' image. For example, as can be seen from Fig. 5.1 the first three images, that corresponds to $t = 1$ and $t = 3$ respectively, provide poor segmentation results. As t increases, the segmentation is more homogenous and the main structures in the original image can be separated, for example $t = 4$ in sub image (e). Another interesting aspect of increasing the diffusion time parameter t is the smoothing effect on the pairwise distance between points in the embedding space. Increasing the t reduce the pairwise distance between similar tensors and increase the distance between dissimilar tensors. In the segmentation case, the result will be pixel label change.

Table 5.1. Comparing the estimated cost of performance of the dictionary, where **d** is the estimated intrinsic dimension, **SVD Cost - Full** G is the computational cost estimate for a full kernel decomposition, **SVD Cost - Approx.** G is the computational cost estimate for the decomposition of the approximated kernel according to Eq. 4.5 and **Dict. Size** is the number of dictionary members.

Image	**Size**	**d**	**SVD Cost - Full** G	**SVD Cost - Approx.** G	**Dict. Size**
Hand	104x128	2	$O\left(26624^3\right)$	$O\left(26624 \times 16\right)$	2
Sport	40x77	2	$O\left(6160^3\right)$	$O\left(6160 \times 16\right)$	2

The dictionary construction enable the utilization of the PTE for image segmentation. The computational cost was reduced from the SVD decomposition of the full super kernel into a SVD/QR of the extension coefficients E which can be efficiently computed.

6 Conclusions

The proposed construction extends the construction in [4]. This is done by efficient dictionary-based construction for the utilization of non-scalar kernels that measure non-scalar similarities between data points. The utilized dictionary contains patches of the underlying manifold, which are represented by the embedded tensors from [6], instead of individual data points. Therefore, it encompasses multidimensional similarities between local areas on the underlying manifold of the data, and it alleviates the computational costs of spectral analysis of such data. In future works, the dictionary-related approximation bounds will be tightened and additional dictionary constructions will be investigated.

Acknowledgements. This research was supported by the Israel Science Foundation (Grant No. 1041/10), the Eshkol Fellowship from the Israeli Ministry of Science & Technology and Agora Center, University of Jyväskylä, Finland.

References

1. Baker, C.: The Numerical Treatment of Integral Equations. Clarendon Press, Oxford (1977)
2. Coifman, R., Lafon, S.: Diffusion maps. Applied and Computational Harmonic Analysis 21(1), 5–30 (2006)
3. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall, London (1994)
4. Engel, Y., Mannor, S., Meir, R.: The kernel recursive least-squares algorithm. IEEE Transactions on Signal Processing 52(8), 2275–2285 (2004)
5. Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29, 1–27 (1964)
6. Salhov, M., Wolf, G., Averbuch, A.: Patch-to-tensor embedding. Applied and Computational Harmonic Analysis 33(2), 182–203 (2012)
7. Singer, A., Wu, H.: Orientability and diffusion maps. Applied and Computational Harmonic Analysis 31(1), 44–58 (2011)
8. Singer, A., Wu, H.: Vector diffusion maps and the connection laplacian. Communications on Pure and Applied Mathematics 65(8), 1067–1144 (2012)
9. Wolf, G., Averbuch, A.: Linear-projection diffusion on smooth Euclidean submanifolds. Applied and Computational Harmonic Analysis (2012), doi:10.1016/j.acha, 03.003

Where Are We Going? Predicting the Evolution of Individuals

Zaigham Faraz Siddiqui[1], Márcia Oliveira[2], João Gama[2], and Myra Spiliopoulou[1]

[1] University of Magdeburg, Germany
{siddiqui,myra}@iti.cs.uni-magdeburg.de
[2] LIAAD - INESC TEC and University of Porto, Portugal
jgama@fep.up.pt, mdbo@inescporto.pt

Abstract. When searching for patterns on data streams, we come across perennial (dynamic) objects that evolve over time. These objects are encountered repeatedly and each time with different definition and values. Examples are (a) companies registered at stock exchange and reporting their progress at the end of each year, and (b) students whose performance is evaluated at the end of each semester. On such data, domain experts also pose questions on how the individual objects will evolve: would it be beneficial to invest in a given company, given *both* the company's individual performance thus far and the drift experienced in the model? Or, how will a given student perform next year, given the performance variations observed thus far? While there is much research on how models evolve/change over time [Ntoutsi et al., 2011a], little is done to predict the change of individual objects *when the states are not known a priori*. In this work, we propose a framework that learns the clusters to which the objects belong at each moment, uses them as *ad hoc states* in a state-transition graph, and then learns a mixture model of Markov Chains, which predicts the next most likely state/cluster per object. We report on our evaluation on synthetic and real datasets.

Keywords: clustering, cluster transition, data streams, evolutionary data mining, label prediction, perennial objects.

1 Introduction

Stream mining is a mature research field, encompassing algorithms that learn a model from observed data, and adapt it as the data generating distribution changes. However, many real-world applications are not solely interested in capturing model change but also in understanding how individual *objects* evolve. Consider for example patients with a chronic disease or customers of a company. The treating physician, or the marketing department, is interested in understanding how each individual changes in response to observable measures (e.g., medication and marketing actions, respectively) and due to unobservable factors (health deterioration caused by the disease and changes in the customer's personal preferences, respectively). In view of the current economic downturn, the

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 357–368, 2012.
© Springer-Verlag Berlin Heidelberg 2012

ability to accurately predict corporate bankruptcy is also of utmost importance and one of the steps to avoid it. By analysing the information of the registered companies, at the end of each accounting period, it is possible to derive distinct company profiles, and generate probabilistic estimates of their evolution. Thus allowing to predict the likelihood that they will survive or go bankrupt. We propose a framework that captures the evolution of objects over time, derives 'ad hoc' states (as opposed to the *a priori* defined states of a Markov model), and predicts the transitions of objects from one state to another.

Intuitively, the individuals, whose evolution is to be predicted, can be modelled as *relational* objects: a patient is an entity accommodating personal information and is linked to further instances - medical tests, diagnoses and treatments. These instances constitute a stream, while the individuals themselves are *perennial*, in the sense that they are permanently stored and (ir)regularly enriched with stream instances that reference them. Relational stream mining encompasses methods that learn and adapt a model for such objects [9,10,3]. However, these methods do not predict the next state/cluster of an object.

There are many methods that study how clusters evolve in a concrete application, e.g. in the context of community evolution. However, as elaborated in the extensive literature discussion of [8], works on *modelling and monitoring* the phenomenon of cluster evolution are very sparse and essentially point back to the early framework MONIC [11]. MONIC encompasses a set of 'transitions' that a cluster may experience (split into many clusters, merge with other clusters, survival and disappearance), a set of measures and a cluster comparison mechanism using them to assess whether a cluster observed at some timepoint has survived, disappeared, merged or become split at the next timepoint. The frameworks MEC [8] and FINGERPRINT [7] build upon MONIC to explain evolution: they both model the clusters and their transitions as nodes, respectively directed edges, in an *evolution* graph, which they use to explain how the underlying population evolves. FINGERPRINT [7] summarizes paths on this graph to build a more concise representation of evolution. MEC [8] observes the evolution graph as a state-transition graph and applies Hidden Markov Models to explain the evolution of the underlying dynamic environment, and has been used experimentally to explain the evolution of Portuguese economic activity sectors on the basis of a recent dataset (cf. [8] for details). In this study, we use the concepts of MONIC [11] and extend MEC [8] by deriving a minimal number of Markov Chains that can explain the observed evolution graph.

Hidden Markov Models are also used by Laxman et al. in [5], the goal of which is to predict target event types over a categorical sequence stream of historical events. Laxman et al. first identify significant frequent episodes of event sequences in a time window, and then estimate a mixture of Hidden Markov Models (HMM), each one associated to a distinct episode. The estimated model is then used to predict future occurrences of a given target event type. However, they work in the context of supervised learning. We rather aim to identify Markov Chains that explain the unlabelled data.

There are also studies concentrating on how individual objects evolve over time. Gaffney and Smith [2] model the evolution of an object as a trajectory and cluster together objects that evolve similarly. Krempl et al. [4] extended [2] into the online algorithm TRACER that discovers and adapts the clusters as new observations of existing objects arrive and new objects appear. However, these algorithms concentrate on monitoring objects that evolve similarly, while our approach, similarly to MEC [8] and FINGERPRINT [7] concentrate on deriving models that explain the transitions of the clusters themselves.

We propose a framework that incrementally (a) derives and (b) adapts the states of perennial objects, and (c) predicts a perennial object's next state, whereupon the states are clusters that are learned and adapted at each timepoint. We then build a transition graph for the perennials, where a perennial experiences a 'transition' whenever it moves from one cluster to another. We use this transition graph to find the optimal number of Markov Chains that can predict the next state of each object, given the states learned thus far.

The paper is organized as follows. In the next section we formalize the problem of predicting the next state of a perennial object, when the states are themselves learned. Thereafter, we present our framework for learning the states and then learning a state transition model over them. We then report on our experiments in a synthetic data set with complex forms of drift and in a real dataset. The last section concludes our study.

2 Problem Description

Assume a stream of perennial objects that contains an infinite sequence of objects $\mathcal{O} = \langle x_1, \ldots, x_j, \ldots \rangle$ and a discrete sequence of timepoints $\langle t_1, \ldots, t_i, \ldots \rangle$. The objects from $\mathcal{O}$ are encountered at regular intervals with a different set of values, which is consistent with their most recent state. The objects arrive as a stream of objects $\mathcal{X} = \langle x_1^{t_1}, x_2^{t_1}, \ldots, x_1^{t_i}, x_2^{t_i}, \ldots \rangle$ with increasing number of timepoint t_i. For any two occurrences of $x_j^{t_i}$ and $x_j^{t_k}$ of an object x_j, $x_j^{t_i} \neq x_j^{t_k}$, if $t_i \neq t_k$. For multiple instances of x_j, the one with the higher t index represents the most recent instance of x_j, i.e., if $t_i > t_k$, then $x_j^{t_i}$ is its most recent instance.

Being perennial, the objects are dynamic. We investigate how these perennial objects *evolve* over time. Formally, for a given instance $x_j^{t_i}$ of an object x_j at timepoint t_i, what is the most probable instance $\hat{x}_j^{t_{i+1}}$, for x_j, at timepoint t_{i+1}? For the case where x_j has $d = 1$ dimension only, this is a single-value prediction. When $d > 1$ though, the prediction task becomes multi-valued and is not trivial (see left of Figure 1).

We reduce the problem to a single-valued prediction task by utilising the clustering structure over the data. In right of Figure 1, we show a clustering model ζ that is learned over the individual instances of the objects for a given time horizon. Each centroid/cluster in the model represents a possible state which an object can inhabit. For example, if centroid $C3 \in \zeta$ is closest to the instance x_j^i of an object x_j at t_i, then x_j is said to be in state $c_j^i = C3$ at t_i.

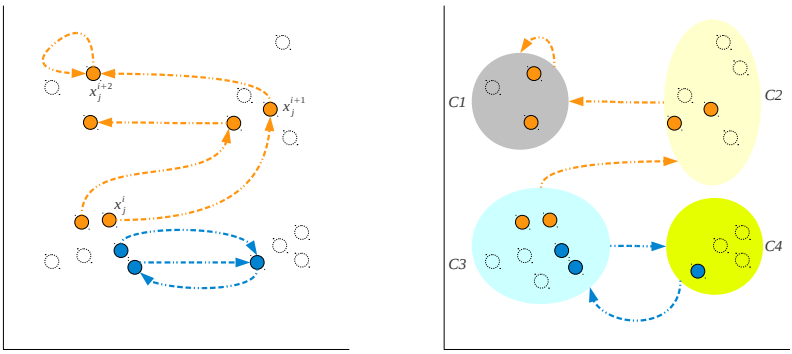


Fig. 1. (left) The original problem reduced to **(right)** a single-value prediction task. Different transition profiles are represented in different colours. (In the left sub-figure we mark the timepoint information for object x_j only. The rest can be interpreted on the basis of transitions.)

Therefore our question becomes, for a given object x and its state c^i at t_i, what is the most likely state $\hat{c}^{i+1}$ that x would acquire at t_{i+1}.

Our prediction task is of evolutionary nature and the transition profiles are defined over in terms of how objects change the cluster membership over time. This means that a cluster found at some timepoint serves as an 'ad hoc' learned state, so that an object moving from one cluster to another experiences a *state transition*. In clustering, there can be many different clusters that describe the objects. In the context of our framework, this means that there can be more than one type of evolutionary patterns on the objects, depending on the clusters.

Our method builds up on the MEC framework [8]. As discussed in Section 1, MEC traces evolution through the detection and categorisation of clusters transitions, such as *births*, *deaths*, *splits* and *merges*. The clusters are represented by enumeration (also known as *extensional definition of clusters*). In this kind of representation a cluster is defined by its members, i.e., by the observations that were assigned to it by a given clustering algorithm. It takes as input a set of clusterings, each one generated at a different timepoint. It performs pairwise mappings, between clusters obtained at timepoint t_i and at timepoint t_{i+1}. The *mapping* process explores the concept of conditional probability and is restricted by a user-defined threshold - the *survival threshold* τ, where $\tau \in [0.5, 1]$.

As in MEC, we trace evolution through the detection and categorisation of cluster transitions. However, we do not use the extensional definition of clusters: rather, the evolution of the objects results to new clusters, changing the nature of clusters and the subsequent transitions. We thus observe multiple evolutionary patterns in the data, not a single one as MEC. Hence, we aim to identify the optimal number of Markov Chains that explain the observed evolution.

3 Framework

Our proposed framework for mining and predicting over a stream of perennial objects consists of two components. First component comprises of a clustering algorithm that operates over stream with a *flattened* time horizon[1] and finds clusters over the individual instances of the objects. All clusters that exist within the given time horizon are discovered by the algorithm. Second component takes as input the clustering result. It learns a mixture of Markov Chains based on how objects change their cluster membership over time. Using this mixture of Markov Chains, we predict what is the next most likely cluster/ state an object is going to be in. We next describe our approach in detail.

3.1 Clustering Objects with Flattened Time-Horizon

First component of our framework operates over all the instances (including past instances) of an object and separates them into groups of similar instances, performing the steps described below.

ID_{old}	Timepoint	Values		ID_{new}	Values
j	i	$\text{tuple}(x_1^{t_1})$	$\rightarrow$	$j \oplus i$	$\text{tuple}(x_j^{t_1})$
j	$i+1$	$\text{tuple}(x_1^{t_{i+1}})$	$\rightarrow$	$j \oplus i+1$	$\text{tuple}(x_j^{t_{i+1}})$
$\ldots$	$\ldots$	$\ldots$	$\rightarrow$	$\ldots$	$\ldots$
j	k	$\text{tuple}(x_1^{t_k})$	$\rightarrow$	$j \oplus k$	$\text{tuple}(x_j^{t_k})$

Fig. 2. Flattened out timepoint information

Time Flattening. Before the clustering can be performed, we first flatten out the timepoint information. This means that multiple instances $x_j^{t_i}, x_j^{t_{i+1}}, \ldots$ of an object x_j are initially treated as distinct objects: we create their identifiers by concatenating the timepoint information and the identifier of x_j. Timepoints are not exploited for the individual instances-turned-objects, as we illustrate in Figure 2. The timepoint information is not discarded, though: we later use it when we re-construct the original objects and their evolution as sequence of cluster membership transitions.

Constructing Cluster Transition Sequences. Once the instances of the objects are flattened out, the instances are fed to a clustering algorithm[2]. In Figure 3, we depict the flattened instances and the clusters learned over them. It is important to note that the clusters span across the whole time horizon. There might be some clusters that contain instances that belong to only a subset of timepoints, while others contain instances from the complete set of timepoints.

[1] We describe how we flatten the time horizon later in the section.

[2] We have experimented with various clustering algorithms and have found EM clustering [1] to work best with the datasets that are available to us. The parameters for the clustering, i.e., K (#clusters) can be discovered by experimentation.

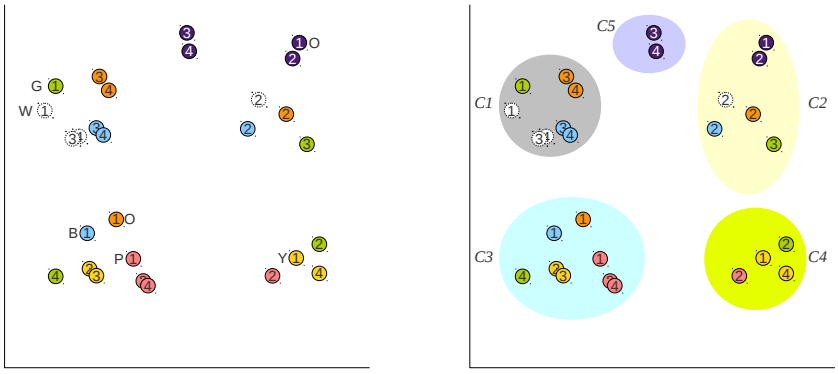


Fig. 3. (left) Flattenned instances, **(right)** Clustering over instances

Before the transition model can be learned, the results are first converted into sequences. First, the concatenated identifiers are un-winded and the object and timepoint data are linked to the individual instances. These data are then used to construct sequences for each individual object. The sequence represents how the individual instances of an object underwent a change, how the object changed clusters and evolved over time. For a sequence seq_x of object x, the i-th element of the sequence holds the cluster id for the instance x^i at t_i. In the left of the Figure 4 we depict the constructed sequences.

3.2 Learning the Cluster Transition Model for Perennial Objects

From the constructed sequences, the cluster memberships for an object x can be determined for each timepoint. For two consecutive timepoints t_i and t_{i+1}, the object x is said to make a transition from cluster c^i at t_i to c^{i+1} at t_{i+1}. However, there can be more than one type of transition profile exhibited by the objects. Before we can learn a Markov Chain, we separate the sequences for the objects into groups. Each group of sequences serves as a seed and a separate Markov Chain is learned over it.

In order to separate the sequences into groups we used Levenshtein distance [6]. Unlike other measures (i.e., Euclidean distance and cosine similarity) the groups are not prototype-based. First, all the unique sequences are identified and the $benefit()$ is computed for each sequence whether it can serve as a good seed or not. The formula for benefit is given in the Equation 1.

$$benefit(s) = \sum_{s' \in seqs} \left\{ \begin{array}{ll} dist_{max} - d & d < dist_{max} \\ 0 & d >= dist_{max} \end{array} \right\} \tag{1}$$

where, $d = distance(s, s')$ and $seqs$ are the constructed sequences for the objects.

In order to select the seeds for the mixture model, we use following heuristic. We sort the sequences on the $benefit()$ value and compute the average benefit

ID	Sequence	ID	Sequence
B	3211	U	2255
O	3211	X	3211
M	1211	Y	3211
P	3433	Z	4334
Y	4334		
G	1423		
V	2255		

Unique	Contributors	Benefit
3211	3211:4, 1211:1	9
~~1211~~	1211:1, 3211:4	6
4334	4334:2, 3433:1	5
~~3433~~	3433:1, 4334:2	4
2255	2255:2	4
~~1423~~	1423:1	2
#seeds=3	$\frac{b_{avg}}{2}$=3	b_{avg}=6

Fig. 4. (left) Objects reconstructed as sequences of clusters they belong to and **(right)** sequence sorted according to the benefit values. The benefit is computed with $dist_{max} = 2$. Sequences with $benefit() > \frac{b_{avg}}{2} = 3$, serve as potential seeds. Sequences that have already contributed to a Markov Chain, e.g. `1211`, get removed from the potential seeds.

b_{avg}. We mark all sequences with $benefit() > \frac{b_{avg}}{2}$ as potential seeds. Starting with the first sequence s with the highest benefit, we train a Markov Chain over all sequences s' such that $distance(s, s') < dist_{max}$. In other words, if a sequence s' contributed to the benefit value for s, it is incorporated into the Markov Chain for seed s. Once a sequence s' contributes to some Markov Chain of sequence s, it is unmarked as potential seed and we do not learn a separate Markov Chain for it. We keep learning the Markov Chain until there are no more potential seeds left or until we have learned k_{MC} Markov Chains[3]. The process of sequence separation is shown in right of Figure 4 for the running example. At the end of this process, some sequences might have been left, which do not contribute to any Markov Chain (e.g., seq "`1423`"): they are assigned to the Markov Chain with the highest log-likelihood.

4 Evaluation

We have evaluated our framework on a real and a synthetic datasets described below. The objective of the evaluation was to learn the transition profiles with the least number of Markov Chains, while observing least number of examples.

4.1 Data Description

European Companies (EC) Dataset To evaluate our framework in a real world scenario, we extracted publicly available dataset[4], comprising financial and accounting information for a set of European individual companies, for the time horizon that goes from 2003 to 2007. After a pre-processing stage, which involved the removal of missing values, duplicates and outliers, we selected only the subset of companies with information reported for all timepoints under analysis. We obtained a set comprised of 836 distinct companies, each one characterized

[3] k_{MC} parameter explicitly sets the number of Markov Chains in the mixture. However, setting it is not mandatory, it is an *optional* parameter.

[4] `http://pages.stern.nyu.edu/~adamodar/New_Home_Page/data.html`

by 7 continuous variables. We considered this dataset appropriate to test our approach due to two main reasons:

1. Financial and accounting information resulting from companies activity is dynamic since it is collected at regular timepoints, usually on a yearly basis.
2. One of the basic assumptions in Management is the *principle of continuity* that claims that a company's goal should be to develop its activity continuously and with unlimited duration, avoiding the interruption or cessation of its activity. Therefore, it is highly likely that over time any profit-driven organization will go through a series of different stages during its life cycle.

Synthetic Dataset The data generator[5] takes as input the number of clusters and number of transition profiles. It first initialises Gaussian distributions for each cluster. Transition profiles are defined in terms of how objects jump from one Gaussian cluster to another. The transitions are represented as a matrix and are generated in a semi-controlled way: each object adheres to certain transition profile. T initial cluster for each object is chosen randomly; a transition matrix is used to determine subsequent clusters. Individual instances for each object are generated depending on the cluster the object was in at t_i.

4.2 Evaluation Settings

Baseline. We compare our strategies for the real dataset against a baseline algorithm we call MEC+; it is built upon MEC [8]: it uses the graph generated by MEC and learns a single Markov Chain, which we use to predict the next state of the perennial objects.

As pointed out in Section 3.1, we have used various clustering algorithms: K-Means, EM with Gaussian Mixtures and DBSCAN[6]. To find out the number of clusters K, we tried silhouette coefficient at distinct timepoints. For the complete flattened-out data, the separation was low. Hence, not only was it difficult to find an appropriate value of K, but the quality of K-Means and DBSCAN was also affected. Since our objective is not to build a "crisp clustering" but to discover regions in which objects move, and since EM can find overlapping clusters (which we verified through experimentation as well), we chose it as the base clustering algorithm and vary K according to values of silhouette at individual timepoints.

To compute the performance of the learned models, we initially used two measures: *predictive accuracy* and *perplexity*. Accuracy tell us whether the predicted states match the actual states. Thus, models yielding higher accuracies are assumed to have better performance. However, accuracy does not take into account neither the indeterministic nature of the system nor the probabilistic nature of the outcome of a model. To incorporate this information in our evaluation we consider perplexity, which is a widely used measure to evaluate the generalization performance of a probabilistic model. Intuitively, we can understand perplexity as the degree of surprise of a trained model to unseen data.

[5] `http://omen.cs.uni-magdeburg.de/itikmd/software/index.html`

[6] For DBSCAN we mostly get a big cluster and the remaining is noise

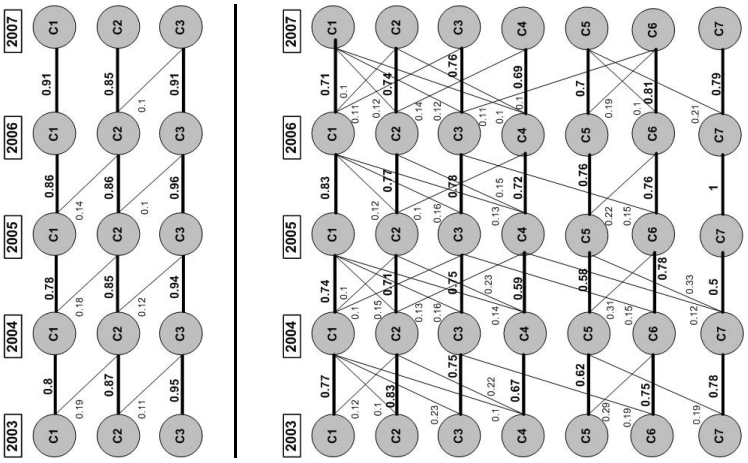


Fig. 5. Bipartite graphs for the European individual companies data, corresponding to time span [2003, 2007]. **(Left)** Setting the number of clusters to $k_t = 3$, $\forall_{t\in[2003,2007]}$. **(Right)** Setting the number of clusters to $k_t = 7$, $\forall_{t\in[2003,2007]}$.

Therefore, better probabilistic models tend to offer smaller perplexities on test data. When Markov Chains are used for prediction, perplexity measures how well the model fits the underlying unknown process distribution being represented by independent test samples. We use the following formula to compute perplexity:

$$Perplexity(M) = b^{-\frac{1}{N}\sum_{i=1}^{N} log_b P(x_i|M)} \tag{2}$$

where x_i is a test example drawn from the test sample $x_1, x_2, ..., x_N$, with size N, b is the base of the logarithm, given by the number of states in the distribution, and $P(x_i|M)$ is the probability of the most likely outcome for a test example x_i $(i = 1, ..., N)$, provided by the probabilistic model M.

4.3 Experimental Results

Setting the K is an unsolved problem in general. For the concrete scenario, we experiment with different values of K and study how homogeneity increases with the number of clusters. First we show the clustering results over EC dataset along with the transitions (see Figure 5). For a low value of $K = 3$, the clusters discovered are abstract. For most objects there were not a lot of inter cluster transitions. It is apparent that for such a case, the baseline can approximate the transitions well by using only one Markov Chain. While for $K = 7$, algorithm discovered clusters at a higher resolution. The algorithm was able to discover local transitions which otherwise were obscured at $K = 3$. Because of this we would expect our algorithm to show better performance than baseline at higher values of K. For lower values the performance is expected to be competitive.

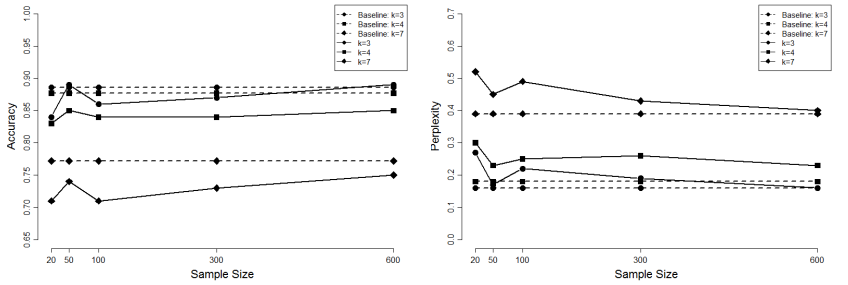


Fig. 6. (left) Accuracy and (right) Perplexity plots for the EC dataset. The mixture model learned for years 2003-2006 with varying sample size for training. The mixture model was used for predicting the clusters for test data for the year 2006. The dotted line represents the baseline MEC for which $k_{MC} = 1$.

In Figure 6 we show the performance for the first set of experiments. The lines depict the different strategies with different parameters, namely the number of clusters $K = 3, 4, 7$ found in the initial clustering (cf. Section 3.1). The strategies were allowed to determine the parameter k_{MC}. Each strategy was compared against MEC+, which was learned at the maximum sample of 600, i.e., number of training examples provided. Varying the sample size has very little effect on the strategies in this set. We see that for each value of K, the baseline ($k_{MC} = 1$) shows better performance than strategies with more than one Markov Chain. This indicates that a single Markov Chain suffices to explain the data.

In Figure 7 we show the results for $K = 3, 4, 7$ clusters. The model was trained for a period from 2003-2006 and the predictions were done for 2007. We see that the performance of strategies with a smaller K is better and their baselines also show better performances. This is partially expected, as there are less cluster

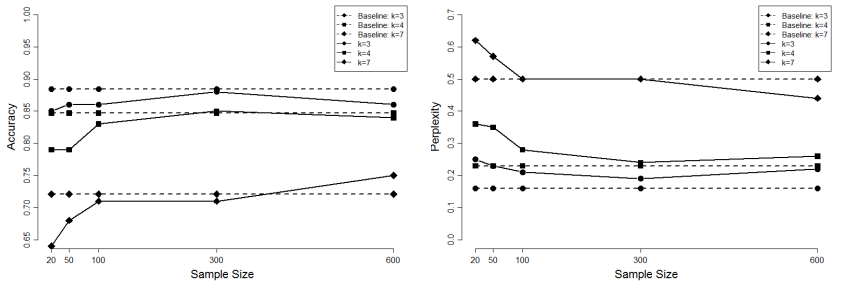


Fig. 7. (left) Accuracy and (right) Perplexity plots for the EC dataset. The mixture model learned for years 2003-2006 with varying sample size for training. The model was used for predicting the clusters for test data for the year 2007, i.e., future.

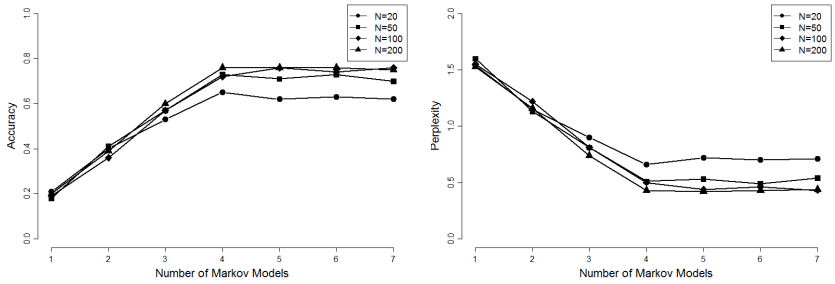


Fig. 8. (left) Accuracy and (right) Perplexity plots for synthetic dataset. The measures are plotted against the number of Markov Chains used in the mixture.

candidates for a transition, so the probability for each one is higher. For $K = 7$, baseline is inferior to the strategy that use more than one Markov Chain, i.e., $k_{MC} > 1$. This is consistent with the earlier discussion, where clustering with $K = 7$ showed complex transitions.

In Figure 8 we show the results on a synthetic dataset with 4 transition profiles and 300 objects. In this set of experiments our strategies used different sample sizes for training. In this experiment, our baseline is not explicitly visible: it can be found at $k_{MC} = 1$. For low k_{MC}, the mixture model was unable to approximate the different transition profiles; for $k_{MC} = 1$ we record the worst performance. As k_{MC} increases, the accuracy of the prediction improves until the right value for k_{MC} is reached. The strategy with the smallest sample size shows the worst performance, since the small fraction of the data makes it difficult to find the appropriate seeds for learning the Markov Chains. In contrast, if right number of transition profiles is not known in advance, sample sizes hardly have any effect on the performance.

5 Conclusion

In the paper we have presented a framework for predicting the evolution of perennial objects. We have used clustering over the timestamped data to derive *states* of the objects, then performed cluster matching and derived *transitions* as migrations from one cluster to another. On this basis, we developed methods that identify different types of transition profiles and learn a mixture of Markov Chains to predict transitions, whereby the objective is to identify the optimal number of MCs needed to describe the transitions. We have presented the first results from the framework on one real and one synthetic dataset. In the experiments, we have found that the number of clusters K influences the number of MCs needed. So, we intend to investigate methods for fixing K by monitoring abrupt changes in cluster homogeneity, and also to use clustering methods that do not require a fixed number of clusters as input.

In this study, we have concentrated on learning ad hoc states with clustering. A future step is the extension of the framework for the supervised case. In particular, we are interested in tackling the issue of label change: the label of a perennial object can change [10], similarly to any other variable. Predicting label change would involve a more elaborate definition of state, and replacing Markov Chains with Hidden Markov Models.

Acknowledgements. First author was funded by the German Research Foundation project SP 572/11-1 "IMPRINT: Incremental Mining for Perennial Objects". Second author was funded by FCT, under the PhD grant SFRH/BD/81339/2011, by the ERDF through the COMPETE Programme, and by National Funds through FCT within the project FCOMP-01-0124-FEDER-022701. This work is also funded by FCT through the project KDUS - Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIAEIA/098355/2008).

References

1. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977); Series B (Methodological)
2. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: 5th Int. Conf. on Knowledge Discovery and Data Mining, pp. 63–72 (1999)
3. Ikonomovska, E., Driessens, K., Dzeroski, S., Gama, J.: Adaptive windowing for online learning from multiple inter-related data streams. In: Workshop on Learning and Data Mining for Robots (LEMIR 2011@ICDM 2011), pp. 697–704 (December 2011)
4. Krempl, G., Siddiqui, Z.F., Spiliopoulou, M.: Online Clustering of High-Dimensional Trajectories under Concept Drift. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 261–276. Springer, Heidelberg (2011)
5. Laxman, S., Tankasali, V., White, R.W.: Stream prediction using a generative model based on frequent episodes in event sequences. In: Procs of the 14th ACM Int. Conf. on Knowledge Discovery and Data Mining, pp. 453–461. ACM (2008)
6. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10(1), 707–710 (1966)
7. Ntoutsi, I., Spiliopoulou, M., Theodoridis, Y.: Summarizing Cluster Evolution in Dynamic Environments. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part II. LNCS, vol. 6783, pp. 562–577. Springer, Heidelberg (2011)
8. Oliveira, M., Gama, J.: A framework to monitor clusters evolution applied to economy and finance problems. Intelligent Data Analysis 16(1), 93–111 (2012)
9. Siddiqui, Z.F., Spiliopoulou, M.: Combining Multiple Interrelated Streams for Incremental Clustering. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 535–552. Springer, Heidelberg (2009)
10. Siddiqui, Z.F., Spiliopoulou, M.: Tree Induction over Perennial Objects. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 640–657. Springer, Heidelberg (2010)
11. Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., Schult, R.: MONIC – modeling and monitoring cluster transitions. In: 12th Int. Conf. on Knowledge Discovery and Data Mining, pp. 706–711. ACM (2006)

Use of General Purpose GPU Programming to Enhance the Classification of Leukaemia Blast Cells in Blood Smear Images

Stefan Skrobanski[1], Stelios Pavlidis[2,*], Waidah Ismail[2], Rosline Hassan[3], Steve Counsell[2], and Stephen Swift[2]

[1] Arithmetica Ltd, London, UK
[2] Department of Information Systems and Computing, Brunel University, London, UK
[3] Department of Haematology, School of Medical Sciences, Universiti Sains Malaysia
Stefan.Skrobanski@arithmetica.com,
{Stelios.Pavlidis,Waidah.Ismail,Steve.Counsell,
Stephen.Swift}@brunel.ac.uk

Abstract. Leukaemia is a life threatening form of cancer, which causes an uncontrollable increase in the production of malformed white blood cells, termed blasts, inhibiting the body's ability to fight infection. Given the variety of leukaemia types and the disease's nature, prompt diagnosis is essential for the choice of appropriate, timely patient treatment. Currently, however, the diagnostic process is time consuming and laborious. To target this issue we propose a methodology based on an existing system, for automated blast detection and diagnosis from a set of blood smear images, utilising a mixture of image processing, cellular automata, heuristic search and classification techniques. Our system builds upon this work, by employing General Purpose Graphical Processing Unit programming, to shorten execution times. Additionally, we utilise an enhanced ellipse-fitting algorithm for blast cell detection, yielding more information from captured cells. We show that the methodology is efficient, producing highly accurate classification results.

Keywords: General Purpose GPU Programming, Image Processing, Leukaemia, Evolutionary Programming, Classification.

1 Introduction

Acute Myeloid Leukaemia (AML) is a disease of as yet unknown cause, characterised by elevated production of abnormal cells, termed blasts, by the bone marrow. The official World Health Organisation classification system recognises eight distinct AML types, known as M0 to M7 [1, 2]. Importantly, the early diagnosis of the correct type of AML is essential for appropriate treatment [3]. In common medical practice this is achieved through manual microscopic inspection of blood smear images by

* Corresponding author.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 369–380, 2012.
© Springer-Verlag Berlin Heidelberg 2012

trained haematologists and requires identification and calculation of the number of blast cells present in a sample [4]. Given that the diagnostic process is laborious and time consuming, lasting up to 5 days, the development of methods for automatic diagnosis, to aid the clinician, will be highly beneficial.

Naturally, digital image processing for this purpose has drawn considerable attention. In brief, image processing is performed using transformations, taking an image as input, and returning a new image or data describing characteristics of the input image [5]. A variety of methodologies have been proposed, which aim to segment and classify white blood cells present in blood smear images, although none are used specifically for segmenting blast cells. In [6] the authors discuss the application of a number of image processing techniques, including Contrast Stretching and Edge Detection, in order to segment the nucleus of the white blood cells. They utilise features extracted from the nucleus to classify the dataset via a Neural Network. This approach is inappropriate for our purposes, since features of both the cell nucleus and the cell membrane are required for classification of AML subtypes. Elsewhere, a method for segmentation of white blood cells, using Feature Space Clustering is presented; the method employs Scale-Space Filtering and Watershed Clustering in order to locate several features in the white blood cells [7]. The use of Gradient Flow Vector Snakes to detect the boundaries of white blood cell nucleus has also been examined [8]. Both of the aforementioned methods may produce good results whenever white blood cells are well defined and separated from other features in the image, but their efficiency diminishes substantially in cases of overlapping cells.

In another publication, the authors use a simple form of thresholding to segment the white blood cells, and employ Principle Component Analysis for feature extraction and classification [9], while in [10] Watershed Transformation and Snakes to detect the cell nuclei is discussed. Borovicka, proposes a method for automatic circle detection using the Hough Transform [11], however, since leukaemia cells are not perfectly circular this method may not yield the desired results. In [12] a method for ellipse fitting of arbitrary shapes using a linear least squares method is discussed. Again this approach seems impractical for detecting blast cells due to noise within the image. Consequentially, the application of digital image processing, with the ultimate purpose of achieving automatic diagnosis of leukaemia, to aid morphologists in their task, remains an issue that is open to investigation.

In this paper we present an empirical study investigating the potential gains of employing General Purpose GPU programming within a system designed to locate blast cells within blood smear samples and classify them according to AML subtypes. The project is based on previous work by Ismail and Swift, proposing the employment of image processing, cellular automata, and heuristic search techniques for blast cell classification [13, 14]. In addition to speeding up the processing time, we propose an ellipse fitting algorithm that better captures blast cells present in the images, increasing classification accuracy.

The rest of the paper is organised as follows; Section 2 presents an overview of the methodological approach, including some detailed discussion of the distinct steps and the specifics of the algorithm. Section 3 presents the obtained results, while section 4

concludes the manuscript with some general comments and plausible directions for future work.

2 Materials and Methods

2.1 Dataset Description

The dataset used throughout this project consists of 322 images of blood smear samples from different patients, suffering from four distinct subtypes of AML, including M1, M2 M3 and M5 types. The images were supplied by Hospital University Science Malaysia (USM), Kota Bahru, Lelantan. Figure 2 shows a characteristic example of such an image.

2.2 Methodology

As aforementioned, we implement general purpose GPU programming to boost the speed of the proposed image processing approach. Graphical Processing Units (GPUs) constitute a type of co-processor designed for high performance rendering of 2D and 3D graphics. GPUs are a form of what's known as 'many-core' processors where each GPU may have hundreds of processing cores as opposed to CPUs, which typically have up to 4 cores. Over the last decade the processing power of GPUs has increased at unprecedented levels with the latest GPUs capable of processing speeds of up to 1 teraflop (TFLOP), dwarfing the processing power of modern CPUs by about 10 to 1. The main bottle-neck with GPU programming is the process of transferring data to and from the host memory to the GPU memory. In this paper this problem has been dealt with by copying all of the images over to the GPU card all at once.

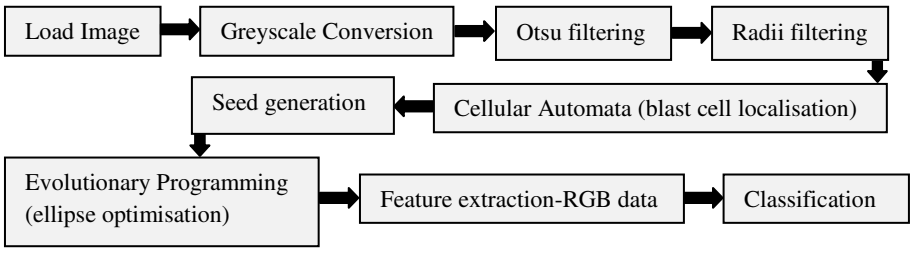


Fig. 1. System Workflow

The application presented here was developed using Microsoft Visual Studio 2010, in Microsoft Visual C++. The CUDA C source code was developed using NVidia's Parallel NSight plugin for Visual Studio, which provides code colouring, IntelliSence and code compilation using NVidia's nvcc compiler. Parallel NSight is also supplied with a memory profiling tool to allow for visual profiling of memory and processing loads throughout the execution of a piece of software running on the GPU. The application uses a number of third party Libraries and Application Programming Interfaces (APIs), which

are summarised on table 1. The CPU used for the test was an Intel iQ7 processor which has 4 cores, the CPU software was written in sequentially executing native C++ and did not use multithreading of any sort. Figure 1 provides a brief summary of the workflow followed in the proposed system. A more detailed discussion of each step is presented in the following sections.

Table 1. Utilised third party Libraries

The Standard Template Library (STL)	Consists of containers, iterators, and algorithms designed to aid development in the C++ programming language [15]
LibTIFF	Provides support for the reading and writing of the Tag Image File Format (TIFF) image format [16]
NVidia CUDA Runtime Library	C++ API which provides a set of wrapper functions to expose the NVidia CUDA Driver API as familiar C++ template functions with overloading, references, and default arguments [17]
Thrust	C++ parallel algorithms template library which is designed to resemble the C++ STL. Uses the GPU and multicore CPUs to perform algorithms in parallel, and integrates seamlessly with CUDA applications, aiding productivity [18]
CURAND	Random number generation library. Used in the blast cell detection software within the heuristic search component [19]
Windows GDI+	C/C++ based API used for simple 2D graphics and formatted text. GDI+ is part of the Microsoft Win32 API. GDI+ is used within the blast cell detection software to annotate the resultant images [20].

2.2.1 Otsu

In our analytical approach, a grey scale filter is first applied to an image and the resulting image processed with Otsu's method of Image Thresholding [21], a nonparametric, unsupervised method of automatic threshold selection for image segmentation. Consequentially, the greyscale image is discretised, converted into a binary (black and white) image, where the sum of the pixels in each class is at a minimum. In the binary image pixels are often described as being part of the "foreground" or "background" depending on which class they belong to. Following that, blast cells are detected by the radii filtering step (Figure 1).

2.2.2 Cellular Automata

A Cellular Automata (CA) method is applied to the image produced by the Otsu filter in order to find the centres, and radii of candidate blast cells, and then segment the image into a series of images corresponding to individual blast cells. A CA can be visualised as a grid of cells, each one having a finite number of possible states, two in the simplest case (e.g. "true" and "false"). The CA operates by iteratively scanning through the grid and changing the state of each cell depending on the state of a "neighbourhood" of cells and a predefined set of rules. A single instance of the grid is known as a generation [22]. In the implementation discussed here, at the initial stage

or first generation, foreground pixels are considered "alive", holding a value of 1, while background pixels "dead", with a value of 0. Our CA follows a simple rule, where pixels with any "dead" neighbours, are killed off. The process continues until all pixels "die", while the generation at which this occurs is recorded for each pixel, on a separate "survival" matrix of the same dimensions as the original image. The process is similar to image dilation, apart from recording pixel transition generations. Given that white blood cells are larger on average then red blood cells with diameters of 6-8μm and 7-21μm respectively, it is assumed that the last pixels to be killed off by the CA are likely to reside at the centre of white blood cells present on the slide.

Next, a set of bounding rectangles that encapsulate the cells is defined. This is achieved by first scanning each horizontal line to check whether it contains data. We can thus define a set of horizontal bands containing data, each of which have a specified height. Next the vertical lines of each band are scanned, resulting in a set of rectangles in each band that encapsulate the islands of data. These rectangles can then be shrunk so that their dimensions describe the extremities of each island. The approximate centre and radii of the blast cells are then used as seeds in a heuristic search algorithm to find the ellipse best approximating the blast cell. An example of the produced seeds, corresponding to blast cells, before optimisation is shown in Figure 2A. Importantly, a CA is naturally a parallel algorithm. In theory the survival of each point/pixel can be computed in parallel.

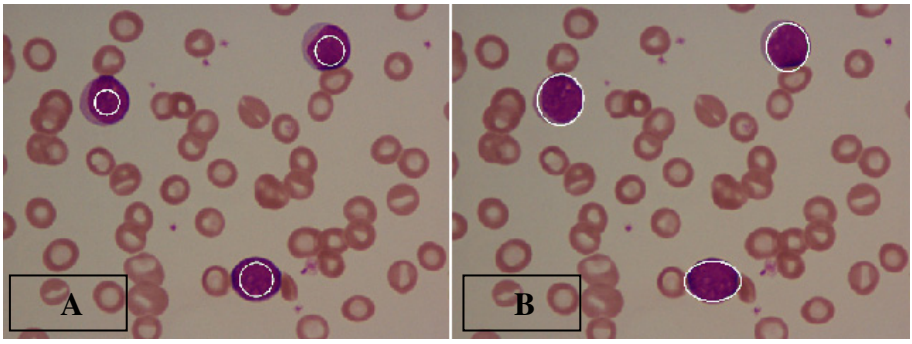


Fig. 2. Blood smear image showing seeds produced via image processing and cellular automata (A) and the subsequent application of heuristic search (B)

2.2.3 Evolutionary Programming

Ismail and Swift implement a selection of heuristic search methods to produce circles of near-optimal approximation to the position and radii of blast cells [13, 14]. The work discussed here builds upon this research by using a different search algorithm to produce ellipses which better approximate the position, dimensions and orientation of blast cells. In particular, we developed an Evolutionary Programming (EP) Algorithm, a type of heuristic search in the Evolutionary Algorithm family [23]. The algorithm works with a population of candidate solutions, a mutation operator using self-adapting parameters, and a tournament selection to achieve a near optimal solution. The EP was chosen as a means of optimising the blast cell geometric approximations

over a GA due to its lack of a crossover mechanism conferring it better suited for parallel implementation. That is, each individual in the population can be mutated and then evaluated in parallel.

More precisely, the EP starts with a population of N individuals, each comprising of a number of genes. Each individual is used to produce an offspring, with mutated gene values. All individuals are then evaluated with respect to a fitness function, competing for survival into the next generation in a stochastic tournament. Each individual is compared to a number of randomly chosen opponents and assigned points depending on the number of opponents with lower fitness. The N individuals with the highest scores are selected for the next generation, through tournament selection. The process is repeated for a set number of generations and the individual with the highest fitness is selected as the solution. Self-Adapting Parameters (SAPs) can be used by the EP as a mutation operator. SAPs allow for more diverse mutations at the beginning of the EP process as a means of escaping local optima [24].

An individual can be denoted as set of n parameters$\{g_1, g_{2,} \cdots, g_n\}$. In our implementation, there are 5 parameters needed to describe each ellipse, the centre coordinates, the major and minor radii and the ellipse's angle (Figure 3). Consequentially, $n = 5 \times m$ where m is the number of ellipses (or blasts) in an image, determined by the CA. An individual also has a set of n self-adaptive parameters $\{\sigma_1, \sigma_2, \cdots, \sigma_n\}$ each of which is the standard deviation used when creating a new mutated offspring (equation (1)).

$$g_i' \sim N(g_i, \sigma_i) \tag{1}$$

$N(\mu, \sigma)$ is a Gaussian random variable with mean μ and standard deviation σ. After being used to create new parameters for the offspring, the self-adapting parameters are themselves subject to mutation, using the equation (2).

$$\sigma_i' = \sigma_i e^{N(0,\tau_1)+N_i(0,\tau_2)} \tag{2}$$

Where $\tau_1 = \frac{1}{\sqrt{2n}}$ and $\tau_2 = \frac{1}{\sqrt{2\sqrt{n}}}$. It should also be noted that $N(0,\ \tau_1)$ is sampled once per individual, while $N_i(0, \tau_2)$ is resampled for each parameter.

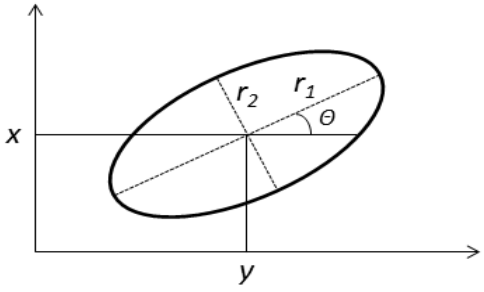


Fig. 3. Ellipse, with r_1 major, r_2 minor axis and angle Θ, at x,y coordinates

As aforementioned, the objective is to produce ellipses which cover as much of the blast cell as possible, with as little of the background or red blood cells encompassed as possible, maximising fitness. To calculate the fitness of each ellipse it is set against the data produced by the Otsu filter, the number of foreground and background pixels contained within the ellipse is counted and the fitness value calculated according to equation (3).

$$fitness = w\left(\frac{w}{w+b}\right)^{k} \tag{3}$$

Here w is the number of white (alive) pixels and b the number of black (dead) pixels within each ellipse, while k is a sensitivity parameter. The overall fitness is then estimated based on the fitness of all ellipses in an image. Figure 2B shows the input image complete with optimised ellipses.

2.2.4 Classification

Once we have the set of optimal ellipses for each blast cell in a given image we extract information from the RGB data within these ellipses. The data consists of attributes for each colour channel, including the Minimum Value, Maximum Value, Median, Mean and Standard Deviation, while the leukaemia subtype is also recorded. These attributes are used for image classification, with a number of relevant algorithms, implemented in WEKA [25]. This stage was not implemented in parallel.

In order to investigate the effectiveness of the system, the software was used to batch process all images in the dataset. The system was tested in both a CPU and GPU configuration, using the same execution parameters and results were compared. In total 2544 ellipses were extracted from the 322 images. Overall, performance was compared based on the time taken to complete the batch processing using each workflow. Measurements of execution time of each step of the workflow for each image were also recorded to provide an insight as to where the greatest performance increases took place. Furthermore, to investigate any difference in quality, the resultant annotated images were visually compared to detect any obvious differences. The resultant ellipse image data was used to train a set of classifiers, and the observed classification accuracies used as a metric to compare the quality of the input data.

2.2.5 Computational Complexity

Otsu's method complexity is $O(M+L^2)$ [26], where L is the number of greyscales and $M = X_{max} \times Y_{max}$, for an image of size X_{max} by Y_{max}. Regarding the CA, given that the time complexity of running the CA one iteration is $O(X_{max}\ Y_{max})$, i.e. processing each pixel at each iteration, the final time complexity is $O(X_{max}\ Y_{max}) = O(MK)$, where $K = X_{max} + Y_{max}$ [27]. Naturally, the complexity of the Heuristic Search depends on the ellipse fitness function, hence, the number of ellipses being fitted and the number of pixels in each circle. Therefore, its complexity is $O(IM)$, where M is the maximum possible number of ellipses and I the number of fitness calls. Thus, the overall complexity of the methodology is equal to O(Otsu) + O(CA) + O(Heuristic Search) and therefore equal to $O(M+ L^2) + O(MK) + O(IM) = O(M(K+I)+ L^2)$.

3 Results

3.1 Processing Speed

The population size used by the EP algorithm was set to 100, with tour size equal to 10 and run for 50 generations. The GPU workflow was also run using a larger population size designed to utilise each core. The computer used for testing used NVidia GeForce GT 330M GPU with 48 CUDA cores and warp size of 32, hence the total number of threads available for execution was 1536. It was impractical to batch process the images with such population size, using the CPU workflow, as it is estimated that this would require around 10 days. Not surprisingly, the approach utilising GPU exhibits a significant performance increase, as displayed on Table 2. With a population size of 100, a near 13-fold execution speed increase was observed.

Table 2. Total processing times for CPU and GPU workflows

CPU 100	56709 sec
GPU 100	4364 sec
Speedup Factor	**12.99x**
GPU 1536	6956 sec

Table 3. Total processing times for CPU and GPU workflows

	Greyscale	Otsu	CA	Radii Filter	Ellipse Fitting
CPU 100	0.02 sec	0.01 sec	0.24 sec	0.01 sec	175.84 sec
GPU 100	0.01 sec	0.00 sec	0.04 sec	0.01 sec	13.05 sec
Speedup Factor	**2.97x**	**2.08x**	**5.42x**	**1.16x**	**13.48x**
GPU 1536	0.01 sec	0.00 sec	0.04 sec	0.01 sec	21.54 sec

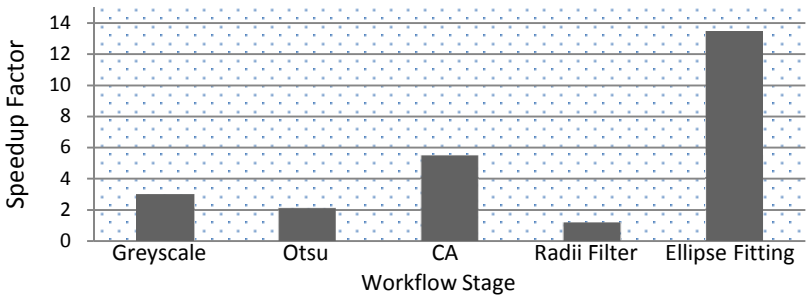


Fig. 4. Mean Speedup Factors achieved during each stage of the workflow

As table 3 shows, a performance increase is observed at each stage of the workflow when using the GPU. However, the image processing operations only equate to a very small proportion of the total execution time. On the other hand, the Ellipse Fitting element achieves a large performance increase, saving on average 162.79 seconds (2.71 minutes) per image. Figure 4 visually depicts performance increase.

3.2 Blast Cell Detection

After batch processing the images using both the CPU and GPU workflows, the images were inspected, and no major differences in the ellipses were observed. However, the system performance exhibited some variability. Figures 5A and B constitute an example of the two extremes, with respect to the quality of the produced ellipses.

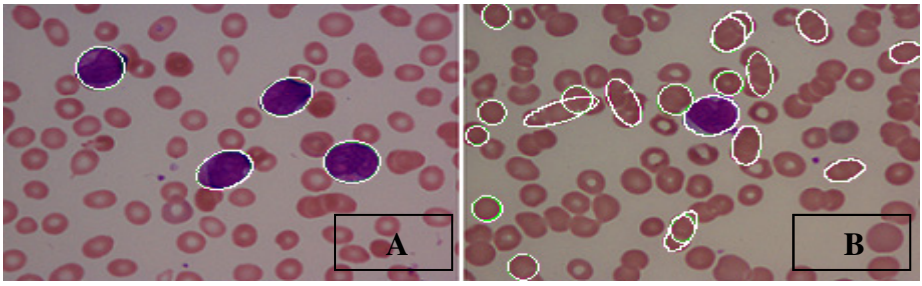


Fig. 5. Correct (A) and wrong (B) blast cell detection

In Figure 5A each of the four blast cells present has been successfully located and segmented, but in Figure 5B a number of red blood cells have also been segmented. Such false positive results are often due to the fact that clusters of red blood cells are of similar size to the larger white blood cells, and are not eliminated by the radii filtering step. It should also be noted that complex situations, such as white blood cells surrounded with clumps of red blood cells are rare in the dataset. The CA and ellipse fitting would often fail whenever they occur. There is room for further investigation to deal with such cases.

3.3 Classification Accuracy

Two distinct classification approaches were implemented, utilising WEKA. Firstly classification was attempted for all four of the AML subtypes present in the dataset, treating each as a distinct class. Second classification into either M3 AML class or other types, grouped together, in separate class, was implemented. This is a sensible approach given that M3 leukaemia requires a different form of treatment than the other types. A variety of classification algorithms, which were deemed appropriate, were tested. Accuracies from both approaches, produced by the most successful methods, are presented on tables 4 and 5. Naturally, the larger population the better the EP performs, leading to more accurate classification.

Table 4. Classification Accuracies for all four types of AML

	CPU (100)	GPU (100)	GPU (1578)
Naïve Bayes	47.75%	58.06%	57.72%
Multilayer-Perceptron	**88.69%**	**86.32%**	**87.92%**
Decision Tree	84.69%	83.14%	82.94%
Random Tree	72.09%	72.09%	74.01%
K-Star	81.54%	80.03%	79.28%

As table 4 shows, accuracies exhibited by different methods show substantial variation, when all four types of AML are considered. The best set of results is produced by the Multilayer-Perceptron, using 10-fold cross validation.

Table 5. Classification Accuracies for the M3/other approach

	CPU (100)	GPU (100)	GPU (1578)
Naïve Bayes	84.96%	89.86%	78.78%
Multilayer-Perceptron	**97.84%**	**97.80%**	**98.38%**
Decision Tree	96.61%	96.03%	95.83%
Random Tree	90.58%	90.88%	94.17%
K-Star	95.53%	95.40%	95.66%

As table 5 shows, a much higher accuracy is observed when images are classified into either M3 or other AML types. Similarly to the results on table 4, the best classification accuracy, equal to 98.38%, is achieved by the Multilayer-Perceptron, using 10-fold cross validation. This is a significant improvement over the classification accuracy of 93.81% reported by Ismail and Swift [14]. Interestingly, accuracies vary slightly between the CPU and GPU implementations, which might be due to small variations in the resultant ellipses in the CPU and GPU workflows. In a few occasions results varied substantially between separate executions. For example, when blast cells are located amongst a large cluster of red blood cells, the algorithm may occasionally converge to encapsulate the cluster instead of just the blast cell.

4 Conclusions

In this paper we presented an empirical investigation into the potential performance increase achieved by employing General Purpose GPU programming within a system to detect blast cells from blood smear samples. In [13, 14] it was shown that with the intelligent utilisation of image processing, cellular automata, heuristic search and machine learning methods, it is possible to automatically detect and classify blast cells of a particular type of leukaemia with very high levels of accuracy. The study presented in this manuscript builds on these findings, employing General Purpose GPU programming in conjunction with an enhanced ellipse fitting technique for blast cell detection. The proposed methodology is more practical and efficient, exhibiting

greater levels of accuracy. The results provide a proof of concept, suggesting that clinical software may have significant applications in this field of medicine, to facilitate clinical decision making.

There are a number of plausible directions for future research left to be explored, that may lead to enhancements of the proposed methodological approach. These include, but are by no means limited to, the following; improving the accuracy of the results yielded by the system, by employing techniques such as Multi-Level Image Thresholding, to segment white blood cells from both red blood cells and the image background. Image clustering techniques can also be employed to avoid processing ellipses wrongly encompassing red blood cells. Utilising more of the GPU processing power by processing multiple ellipses in parallel may shorten processing times even further. In addition, implementation of classification algorithms using GPU programming can be used to fully automate the system and contribute further to processing times reduction. Importantly, the use of feature selection from the RGB data, within the blast cell ellipses, to improve classification accuracies, holds potential. Naturally, analysis of a larger selection of sample images, corresponding to a greater variety of leukaemia types, to build a larger classification model, will be beneficial.

Acknowledgments. This work is based on the final year project of Stefan Skrobanski, in the School of Information Systems, Computing and Maths, Brunel University. Data was provided by the Hospital University Science Malaysia (USM), Lelantan. We would like to acknowledge the EPSRC for providing funding (grant: EP/H019685/1).

References

1. Hassan, R.: Molecular Diagnosis in Acute Leukaemia: Hospital University's experience. In: International Joint Symposium Frontier In Biomedical Sciences: From Genes to Applications, Faculty of Medicine, Gadjah Mada University, Yogyakarta, Indonesia (2008)
2. Vardiman, J.W., Harris, N.L., Brunning, R.D.: The World Health Organization (WHO) classification of the myeloid neoplasms. Blood 100(7), 2292–2302 (2002)
3. Nipon, T.U., Gader, P.: System-level training of neural networks for counting white blood cells. IEEE Trans. SMS-C 32(1), 48–53 (2002)
4. Yi, D.H., Rashid, S., Cibas, E.S., Arrigg, P.G., Dana, M.R.: Acute unilateral leukemic hypopyon in an adult with relapsing acute lymphoblastic leukemia. Am. J. Ophthalmol. 139, 719–721 (2005)
5. Petrou, M., Petrou, C.: Image Processing, The Fundamentals, 2nd edn. John Wiley and Sons, Ltd, Chichester (2010)
6. Piuri, V., Scotti, F.: Morphological Classification of Blood Leucocytes by Microscope Images. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Boston, MA (2004)
7. Jiang, K., Liao, Q.M., Xiong, Y.: A novel white blood cell segmentation scheme based on feature space clustering. Soft Computing - A Fusion of Foundations, Methodologies and Applications 10(1), 12–19 (2006)
8. Zamini, F., Safabakhsh, R.: An unsupervised GVF Snake Approach for White Blood Cell Segmentation based on Nucleus. In: International Conference on Signal Processing Proceedings (ICSP), Beijing (2006)

9. Yampri, P., Pintavirooj, C., Daochai, S., Teartulakarn, S.: White Blood Cell Classification based on the Combination of Eigen Cell and Parametric Feature Detection. In: 1st IEEE Conference on Industrial Electronics and Applications, Singapore (2006)
10. Zerfass, T., Schlarb, T., Elter, M.: Segmentation of leukocyte cells in bone marrow smears. In: 23rd International Symposium on Computer-Based Medical Systems (CBMS), Perth, WA (2010)
11. Borovicka, J.: Circle Detection Using Hough Transform. University of Bristol, Bristol
12. Mulchrone, K.F., Choudhury, K.R.: Fitting an Ellipse to an Arbitary Shape: Implications for Strain Analysis. Journal of Structural Geology 26, 143–153 (2004)
13. Ismail, W., Hassan, R., Swift, S.: Detecting Leukaemia (AML) Blood Cells Using Cellular Automata and Heuristic Search. In: Cohen, P.R., Adams, N.M., Berthold, M.R. (eds.) IDA 2010. LNCS, vol. 6065, pp. 54–66. Springer, Heidelberg (2010)
14. Ismail, W., Swift, S.: Detecting Leukaemia (AML) blood cells using Genetic Algorithms. In: SCOR 2010. Brunel University, London (2010)
15. Microsoft, n.d., Standard Template Library, `http://msdn.microsoft.com/en-us/library/c191tb28.aspx` (accessed March 23, 2012)
16. LibTIFF - TIFF Library and Utilities, `http://www.libtiff.org/libtiff.html` (accessed March 23, 2012)
17. NVIDIA CUDA C Programming Guide. NVIDIA Corporation, Santa Clara (2011)
18. Wen-Mei, H.W.: GPU Computing Gems Emerald Edition, 1st edn. Morgan Kaufmann, San Francisco (2011)
19. NVidia, CURAND Library. 1st edn. NVidia Corporation (2010)
20. Microsoft, GDI+, `http://msdn.microsoft.com/en-us/library/ms533798(v=vs.85).aspx` (accessed March 23, 2012)
21. Otsu, N.: A Threshold Selection Method from Gray-Level Historgrams. IEEE Transactions on Systems, Man, and Cybernetics SMC-9(1), 62–67 (1979)
22. Bandini, S.: Cellular Automata. Future Generation Computer Systems 18(7) (2002)
23. Fang, L.: The New Adaptive Evolutionary Programing. In: IEEE Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao (2010)
24. Ghozeil, A., Fogel, B.D.: Discovering Patterns in Sptial Data Using Evolutionary Programming. In: GECCO Genetic and Evolutionary Computation Conference. MIT Press, Cambridge (1996)
25. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1), 10–18 (2009)
26. Zhu, N., Wang, G., Yang, G., Dai, W.: A fats 2D Otsu Thresholding Algorithm Based on Improved Histogram. In: CCPR Chinese Conference on Pattern Recognition, Nanjing (2009)
27. Ismail, W.: Automatic Detection and Classification of Leukaemia Cells. PhD Thesis, School of Information systems, Computing and mathematics, Brunel University, London, UK (2012)

Intelligent Data Analysis by a Home-Use Human Monitoring Robot

Shinsuke Sugaya[1], Daisuke Takayama[1], Asuki Kouno[2], and Einoshin Suzuki[1]

[1] Department of Informatics, ISEE, Kyushu University, Fukuoka, Japan
{shinsuke.sugaya,takayamaD}@gmail.com, suzuki@inf.kyushu-u.ac.jp
[2] Graduate School of System Life Sciences, Kyushu University, Fukuoka, Japan
asu00798@gmail.com

Abstract. In this paper, we argue that a home-use autonomous mobile robot is a platform for a new kind of Intelligent Data Analysis (IDA). Recent advancement of hardware and software for robotics have enabled us to construct a small yet powerful, autonomous mobile robot from components in low cost. Such a robot is able to perform machine learning and data mining in the real world for a long period, which opens a new avenue for IDA. This paper improves and studies one of our monitoring robots in detail to reveal promising directions and challenges inherent in the new kind of IDA.

1 Introduction

The rapid progress of hardware and software for robotics have enabled us to construct a small yet powerful, autonomous mobile robot from components in low cost. For instance, powerful MPUs such as PandaBoard with a CPU of clock speed 1GHz and 1GB RAM, can be purchased with several hundred USD. Two pairs of Li-Fe batteries, which cost only 132 USD but exhibits 6.6V and 4.6Ah, allows for instance one of our robots function for 120 min with charging time 30 min. OpenCV (http://code.opencv.org/), an open-source library for computer vision, provides a robot and its developer a powerful vision capability and a fast software development environment, respectively. Various kinds of software for SLAM (Simultaneous Localization and Mapping) [1, 2] are available for free of charge on the Web, allowing a robot to navigate and explore in an unknown environment. We define a home-use autonomous mobile robot as an autonomous mobile robot which costs approximately 100K JPY, i.e., 1100 USD, with its size approximately 20cm $\times$ 20cm $\times$ 20cm.

We argue that such a home-use autonomous mobile robot is a platform for a new kind of Intelligent Data Analysis (IDA) due to three reasons. First, it is able to perform an IDA with its powerful sensors such as high resolution cameras and its powerful MPU. PandaBoard may be safely considered as powerful as a PC of a decade ago, when data mining and machine learning programs were executed smoothly on a PC for huge data. Secondly, such a robot is a typical example of physical computing, i.e., it is a computing entity which functions in the real world, which brings various problems and possibilities not considered in

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 381–391, 2012.
© Springer-Verlag Berlin Heidelberg 2012

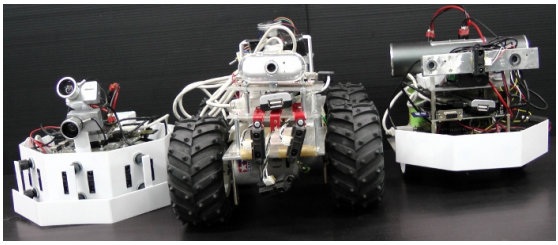


Fig. 1. Our robot platforms: from left to right, the spy robot, the large robot, and the monitoring robot

the standard IDA. Thirdly, such a robot is able to function for a long period, say several months or even a few years, with an appropriate measure such as self charging, which brings new aspects in the current IDA.

Our team has developed several dozens of such autonomous mobile robots in recent years and has been working on various applications [3–7]. To establish the new kind of IDA, we need to clarify promising directions and challenging issues in it. We review our endeavors for home-use autonomous mobile robots, especially the human monitoring application, for the purpose.

The rest of this paper is structured as follows. We explain our mobile robot platform used in this study and our past attempts in Section 2. Section 3 introduces an improved version of our solution for the human monitoring application. We evaluate the effectiveness of our improved approach and clarify the promising directions and the challenging issues in the new kind of IDA in Section 4.

2 Home-Use Autonomous Mobile Robots

2.1 Our Platforms

A home-use autonomous mobile robot may be constructed either from components as we did, or a commercial kit such as LEGO Mindstorm (`http://mindstorms.lego.com/en-us/`). Another option would be to use a commercial product such as iRobot Roomba (`http://www.irobot.com/`). The last option is unpromising due to the lack of convenient commercial product for our purpose. Such a robot either violates our cost requirement or necessitates a hard labor for conversion as they do not satisfy most of our demands. The second option is unpromising either due to similar reasons. On the contrary the first option, which we have adopted, is quite common in robotics.

In about 2 years we constructed 3 kinds of home-use autonomous mobile robots as shown in Fig. 1. They are 7 large robots [5], 10 spy robots [3, 6, 7], and 5 monitoring robots [4]. The large robot costs about 110,000 JPY with its size 22.5cm × 32.0cm × 22.5cm. It is equipped with 3 infrared sensors and 2 high resolution cameras. Its main MPU is BeagleBoard (600MHz clock speed and 128MB RAM) and a 4GB SD card is used for storage. The spy robot also costs

about 110,000 JPY but with its size 22cm × 18cm × 18cm. It is equipped with 8 infrared sensors and 2 high resolution cameras. Its main MPU is PandaBoard and a 16GB or 32GB SD card is used for storage. The monitoring robot costs about 95,000 JPY with its size 20.0cm × 20.0cm × 17.5cm. It is equipped with 1 infrared sensor and 2 high resolution cameras. The two cameras form a stereo vision system and can be moved from downward to upward with a servo motor. Its main MPU is again PandaBoard and a 16GB or 32GB SD card is used for storage.

We explain the monitoring robot [4] in detail. The mounted stereo cameras are used to generate a disparity image for detecting a human and navigating in the office. Most of the computation for an image and motion processing runs on MPU PandaBoard while two motors and I/O sensors except the cameras are controlled with a microprocessor Arduino. PandaBoard runs Linux OS Ubuntu installed in the SD card. Arduino has a clock speed of 16MHz with 8KB SRAM memory and runs a program written in C language installed in on-board 256KB flash memory.

Input sensors are the two cameras for taking a disparity image, the infrared sensor for measuring a distance, and six touch sensors for detecting a collision. The two cameras are Sanwa Supply CMS-V24SETSV and are attached on an aluminum plate with a distance of 10cm from each other. The cameras are connected to PandaBoard, images from the cameras are loaded by a software library V4l2, and the lightness and the focus is controlled by libwebcam library. A disparity image is generated from two images of the stereo cameras by a block matching algorithm of OpenCV. The infrared sensor is Sharp GP2Y0A21YK. Arduino converts an input voltage from the sensor into a distance and thus the sensor is able to detect an obstacle placed in a distance from 5cm to 50cm. The touch sensor is Omron A2A switch, and four and two of them are placed behind the front bumper and behind the two cameras, respectively. When a touch sensor detects a collision, Arduino stops the motors in less than 1msec.

Actuators are two motors, one servomotor, one LED unit, and one loudspeaker. The driving unit consists of Daisen Electronic Industrial RA250100-58Y91 as a motor, Tamiya 70105 as a motorsport tire, and Toshiba TA7291P as a motor controller. By communicating with Arduino, PandaBoard controls the velocity in 20 steps with the maximum speed 20cm/sec. The servomotor controlled by Arduino is used to adjust the perpendicular angle, i.e., the pitch angle, of the camera from 0 degree, i.e., the horizontal direction, to 90 degrees, i.e., the perpendicular direction toward the sky, with 1-degree step. The LED unit is Avago Technologies LED ASMT-MT00-00001, which is able to emit light in either red, blue, green, yellow, purple, orange, or white. The loudspeaker is Sanwa Supply MM-SPIP2.

The batteries, two pairs of two LiFePO4 A123s connected in series, exhibit 4.6Ah - 6.6V and allow our robot to function for at least 2 hours. A DC-DC converter, TDK-Lambda CC6-0505SF-E, transforms the voltage of the batteries to 5V specified by Arduino and PandaBoard. Two battery chargers, HIPERION EOS0606i, allow us to fully charge the batteries in about half hour.

laboratory workshop

Fig. 2. The person and the robot (taken from [4])

2.2 Human Monitoring Task

We have tackled, with the platforms in the previous section, several tasks including human detection [7], human detection and avoidance [5], novel object detection [6], human pursuit [3], and human monitoring in an office [4]. For instance, we equipped the spy robot a human detection method using Histograms of Oriented Gradients (HOG) [8] and Support Vector Machines (SVM) [9] in [7]. The method uses two kinds of SVM classifiers based on an estimation of the distance to the human and exhibited an F value of 0.93 in the experiments. Another instance is the large robot which not only detects humans but avoids them in an office [5] . The robot is equipped with an SVM classifier trained on a PC from subimages taken by the camera mounted toward the ceiling.

Among the tasks, human monitoring [4] seems most promising because it would allow as to develop monitoring robots for aged people at home, which would be a promising consumer electronics product in a rapidly aging country such as Japan. In [4], the monitoring robot tackles a state prediction problem at two different places in our office, one at the side of the desk of the person and the other at the side of another desk in a workshop. The robot moves between the two monitoring places, which are located 10.6m apart. The office is subject to different lightning conditions.

As shown in Fig. 2, the robot predicts the person continuously, mostly for more than 60 min and at most for 100 min. The states to be predicted are related with the productivity of the person. In [4], we defined the state as either stressed, relaxed, usual, or absent as shown in Fig. 3.

2.3 Related Work

There are various works for activity recognition using a fixed camera.Cutler [10] tackled a recognition problem of human activity from airborne video, which is a challenging problem due to image noise, poor contrast and motion blur. To recognize the activity, they defined an approach to encode it as a finite state machine.

Fiore [11] presented a system which tracks pedestrians across a scene of interest and recognizes a set of human activities. The also developed a framework

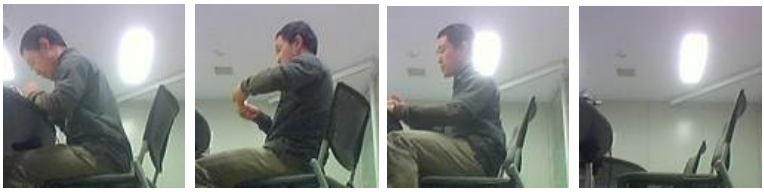


Fig. 3. The 4 states

for the placement of multiple cameras to observe a scene. Moreover, an active dual camera system for task recognition at multiple resolutions was developed. All of these systems were tested under real-world conditions, and were shown to produce usable results.

Zouba [12] described an approach to recognize a set of interesting activities of daily living for elderly at home. They proposed 3D key postures to recognize a set of human activities, which is environment-independent. Using 3D key postures, their system modeled video events from video sequences. They applied the video event recognition algorithm to video sequences with 4800 frames, and the primitive states were well recognized by video cameras in different rooms.

Ayers [13] described a system to recognize human activities, such as entering a room and opening a cabinet, from video sequences. They modeled human activities by a state machine with states and transitions. In addition, their system generated a textual description of them and a set of key frames from video sequences. They recognized human activities in several video sequences.

Most previous works in the area of monitoring activities often use fixed cameras. Few researchers used a small robot probably because it is difficult to monitor a human from a low position.

3 Our Human Monitoring Method

3.1 Overview

We briefly explain our method for the human monitoring [4] and propose several improvements. Unlike [4], we restrict ourselves to the prediction problem of the four states of the person and do not tackle the navigation problem between the two monitoring points.

The postures and the positions of the human vary and so do the monitoring points of the robot. Thus the robot should estimate the region of the human for each of the image it takes. SVM is one of the most successful classifiers for prediction problems from images, as SVM is quite robust against overfitting in classification with a large number of features and most of the image classification is free from class noise. Our method detects the probable region of the human as an adjusting window then applies SVM classifiers to the window.

We will give a detailed explanation on the adjusting window W_{human} in the next section and focus on the classification process. W_{human} is converted to a

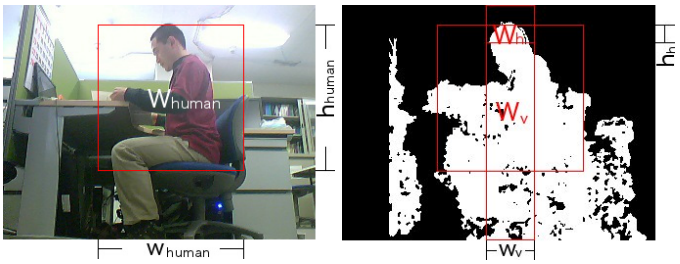


Fig. 4. Adjusting window detection. (a) The original image I and the detected window W_{human}. (b) detection window W_{v} and W_{h} in the binary image I_{b}.

gray-scaled image and then a feature vector $\mathbf{x}$ which consists of the brightnesses of individual pixels [4]. HOG is a robust method against the change of illumination, thereby it allows us to cope with a wide variety of human shapes mainly at various environments. The combination of HOG and SVM is known as the most successful approach for human detection [7]. Thus in this paper we use HOG features in $\mathbf{x}$.

As we stated we tackle a four-class classification whereas SVM solves a binary classification. We adopted a heuristic method in [4] while in this paper we adopt one-against-one approach among the standard methods such as one-against-all [14, 15]. The one-against-one approach regards the k-class problem as a collection of binary classification problems: $k(k-1)/2$ SVM classifiers for each pair of classes are constructed and then the class is determined in a manner of knockout tournaments.

3.2 Inferring the Region of the Human

Extracting the adjusting window is a non-trivial problem due to the presence of various kinds of noise and the change of illumination. We employ a disparity image generated from the stereo cameras for robustness: we detect the human by assuming that he or she is located within a predefined range of distances. To generate the disparity image, we adopted the block matching algorithm in [4]. In this paper, we use the semi-global block matching (SGBM) algorithm [16], which is more precise and almost as fast as the standard block matching algorithm.

We previously estimated the adjusting window based on the horizontal center of the pixels within the region [4]. Here we improve it by also using the vertical center and skin color detection if possible. Our method extracts the window W_{human} of size $w_{\mathrm{human}} \times h_{\mathrm{human}}$ including the human from the image I of size $w_{\mathrm{orig}} \times h_{\mathrm{orig}}$ as shown in Fig. 4 (a). First, it generates a disparity image I_{d} from the image I of stereo cameras by SGBM of OpenCV and converts the disparity image I_{d} to a binary image I_{b} of which pixel has white color if it is located within the predefined range of a distance. A block of white pixels in I_{b} is discarded as noise if the number of white pixels is under t_{b}. An image I_{b} is shown in Fig. 4

(b). Next, a vertical detection window W_{v} of size $w_{\mathrm{v}} \times h_{\mathrm{orig}}$ traverses the binary image I_{b} in the horizontal direction and looks up the position x_{v} that contains the maximum number of white pixels in W_{v}. After finding x_{v}, to find the head of human in W_{v}, a horizontal detection window W_{h} of size $w_{\mathrm{v}} \times h_{\mathrm{h}}$ traverses W_{v} at the position x_{v} in the vertical direction and a position y_{h} is decided if the number of white pixels in W_{h} exceeds a threshold t_{h} for the first time. Moreover, to refine the position of human in W_{human}, x_{v} is modified with a skin color detection to locate the head of human on the center of the upper part of W_{human}. A range of HSV color space ($0 \leq H \leq 50$ and $0.23 \leq S \leq 0.68$) is often used to a skin color detection [17]. However, since in the preliminary experiments the noise and illumination affected our images in the skin color detection, we expanded the range to $H \leq 330 \vee H \leq 50$ and $0.1 \leq S \leq 0.68$. Assuming that the face is located in the upper half of $W_{\mathrm{human}} \cap W_{\mathrm{v}}$, a skin color detection window W_{skin} of size $w_{\mathrm{s}} \times h_{\mathrm{s}}$ is set to the window with the largest number of skin color pixels provided that the number is no smaller than a threshold t_{s}. In this case, x_{v} is modified to be the center of W_{skin} along the horizontal detection. Finally, with x_{v} and y_{h}, we obtain the position $(x_{\mathrm{human}}, y_{\mathrm{human}})$ as $(x_{\mathrm{v}} + \frac{w_{\mathrm{v}} - w_{\mathrm{human}}}{2}, y_{\mathrm{h}})$.

4 Experiments

4.1 First Series

We tested the state prediction on a PC by monitoring the third author in two different places. The distance between the robot and the person was about 1.4m and the cameras of the robot were headed 16 degrees up from the horizontal line. The robot took full-color stereo images at intervals of one second for nearly 8 hours, and the number of images is 28,148. We classified each of 334 images of them to one of the states, which are either stressed, relaxed, usual or absent. The corresponding numbers of examples for the four classes are 132, 51, 108, and 43, respectively. In [4] we used 280 examples and the monitored person assigned the states to images. A size of the image is 320 × 240 pixels and the method described in Section 3.2 determines a location of 150 × 150 window in the original image for the prediction. For parameters of the method, w_{v}, h_{h}, t_{b}, w_{s}, h_{s} and t_{s} were set to 50, 20, 768, 30, 30 and 360, respectively.

We applied SVM and HOG to the dataset of 150 × 150 window. We used SVM-Light [18] with its default setting with and without the polynomial kernel function. Parameters of HOG are a cell size, a block size, and the number of gradient orientations and we performed several combinations of the parameters. For comparison, we also tested a straightforward method with gray-scaled disparity image, which defines one pixel as one attribute of the feature vector. We used 10-fold cross validation with the experiments on PC.

Table 1 shows the results of the experiments. We see that HOG for the cell size of 10 gives the best result, i.e., 91.9%. For the state prediction of usual, stressed and absent, we achieved high accuracy predictions. On the other hand, the accuracies for relaxed are much lower than those for other classes for all approaches. We consider that the number of examples of relaxed is smaller than

Table 1. Results of the accuracy for the state prediction. For HOG(x, y), x is a cell size, y is a block size and the number of gradient orientations is 9. kd represents the use of polynomial kernel of degree d.

	State				
	Usual	Relaxed	Stressed	Absent	Total
HOG (10,3)	96.2	76.5	91.7	97.7	91.9
HOG (10,5)	95.5	78.4	91.7	97.7	91.9
HOG (10,5)k2	96.2	76.5	91.7	95.3	91.6
HOG (10,5)k3	95.5	78.4	92.6	93.0	91.6
HOG (10,5)k4	94.7	78.4	90.7	93.0	90.7
HOG (10,5)k5	93.9	70.6	91.7	86.0	88.6
HOG (10,5)k6	93.2	54.9	93.5	81.4	85.9
HOG (15,3)	94.7	76.5	87.0	97.7	89.8
HOG (15,5)	93.2	70.6	88.0	97.7	88.6
Gray-Scaled	94.7	41.2	75.9	93.0	80.2

others and it is a difficult problem to discriminate relaxed from usual even for a human. The total accuracies of the cell size of 10 is 91.9%. Comparing the block size of 3 and 5 for them, the numbers of attributes of the feature vector are 13689 and 27225, respectively. Since a lightweight processing is suited for a small mobile robot, the combination of the cell size of 10 and the block size of 3 is proper for our robot. By investigating misclassified examples, we found that illumination from a ceiling light affected the detection of disparity images and the gradients. Therefore, to improve the accuracy, we consider that a more robust image processing against illumination is required.

4.2 Second Series

We collected new data in which the second author is monitored and we report preliminary results in this section. The additional images were collected in spring, a few months later than those in the previous section. The numbers of examples are 97, 93, 78, 49, and 317 for usual, relaxed, stressed, absent, and total, respectively.

Table 2 shows the results of the experiments. The method in the previous section gave the first two results, which are poor. Fig. 5 shows three difficult cases, which are over-illumination and blur, out of sight, and illumination misrecognized as a part of the head. These examples show the challenging aspects of long-term monitoring in the real world.

Further inspection revealed that the parallax information at the bottom of the person is more inaccurate in the new data than in the old data. Thus we adjusted the horizontal length of W_{v} by reducing it 20 %. This adjustment, though insufficient, improved the performance as shown in the bottom two lines in Table 2. We found that further features such as the positions of hands should be considered. The new data is more difficult than the old data.

Table 2. Preliminary results of the accuracy for the state prediction for the new data, where -a represents the adjustment for the new person

	State				
	Usual	Relaxed	Stressed	Absent	Total
HOG (10,3)	73.2	62.4	57.7	85.7	68.1
Gray-Scaled	63.9	48.9	21.8	89.8	53.0
HOG (10,3)-a	80.4	69.9	66.7	95.9	76.3
Gray-Scaled-a	67.0	51.6	29.5	91.8	57.1

Fig. 5. Problematic cases in the new data: from the left to the right, over-illumination + blur, out of sight, and illumination mis-recognized as a part of the head

5 Directions and Challenges for the New IDA

The experiments in Section 4 demonstrated the difficulty in observing and predicting online and in real time in the real world. The difficulty increases substantially if we require learning, discovering, and acting, which would be the most obvious direction and challenge in the new IDA.

The experiments also taught us the importance of robustness and the world modeling to successfully accomplish a set of given tasks for a long period under different conditions. Coping with noise becomes much more difficult but is just the beginning of the whole problems. Not only the method but also the goal should be modified, which indicates a similarity to the KDD (Knowledge Discovery in Databases) process [19]. As an autonomous robot, a very challenging aspect would be to automate the process, which corresponds to adaptation at the strategic level to the dynamic environment.

For our application, using Kinect (`http://www.xbox.com/en-US/kinect`) [20] would facilitate the task, as it estimates the human postures and provides relatively accurate depth images as shown in Fig. 6. To researchers of structural data mining, the figure gives an idea of mining patterns from spatio-temporal trees. New powerful sensors make current problems easier but at the same time open the doors to more significant and challenging ones in IDA.

Special attention must be payed to the limited resource such as the batteries of a robot, which adds a new research topic to the conventional IDA. Using multiple robots that communicate each other link the new IDA with multi-agent research. Note also that such a group of robots may be regarded as a generalization of the sensor network research [21], as a robot acts unlike a typical sensor, which

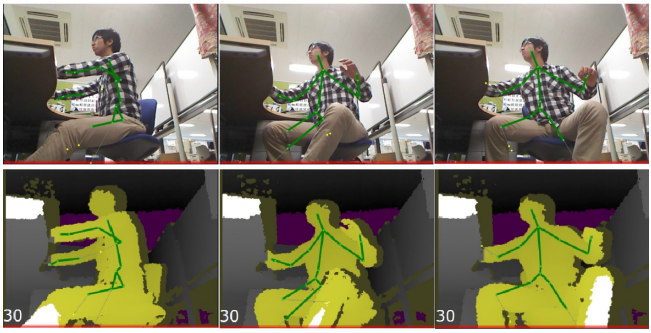


Fig. 6. Examples of information obtained with Kinect

closes our argument that a home-use autonomous mobile robot is a platform for a new kind of IDA.

Acknowledgment. A part of this research was supported by Strategic International Cooperative Program funded by Japan Science and Technology Agency (JST) and JSPS KAKENHI Grant Number 24650070. We thank Kouhei Takemoto for providing us Fig. 6.

References

1. Durrant-Whyte, H., Bailey, T.: Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. IEEE Robotics and Automation Magazine 13(2), 99–110 (2006)
2. Bailey, T., Durrant-Whyte, H.: Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. IEEE Robotics and Automation Magazine 13(3), 108–117 (2006)
3. Boubou, S., Kouno, A., Suzuki, E.: Implementing Camshift on a Mobile Robot for Person Tracking and Pursuit. In: Proc. Eleventh IEEE International Conference on Data Mining Workshops (ICDMW 2011), pp. 682–688 (2011)
4. Kouno, A., Takayama, D., Suzuki, E.: Predicting the State of a Person by an Office-use Autonomous Mobile Robot. In: Proc. of IAT (accepted, 2012)
5. Matsumoto, E., Sebag, M., Suzuki, E.: Avoiding Humans with SVM - a Case of Small Autonomous Mobile Robot in an Office. In: Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences (ISCIS 2011), pp. 283–287 (2011)
6. Takano, S., Suzuki, E.: New Object Detection for On-Board Robot Vision by Lifting Complex Wavelet Transforms. In: Proc. Eleventh IEEE International Conference on Data Mining Workshops (ICDMW 2011), pp. 911–916 (2011)
7. Takemoto, K., Takano, S., Suzuki, E.: Human Detection by a Small Autonomous Mobile Robot. In: Extraction et Gestion des Connaissances (EGC 2012), pp. 531–536 (2012)
8. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Computer Vision and Pattern Recognition (CVPR 2005), pp. 886–893 (2005)
9. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)

10. Cutler, R., Shekhar, C., Burns, B., Chellappa, R., Bolles, R., Davis, L.S.: Monitoring Human and Vehicle Activities Using Airborne Video. In: Proc. SPIE, vol. 3905, pp. 146–153. Int. Soc. Opt. Eng. (2000)
11. Fiore, L., Fehr, D., Bodor, R., Drenner, A., Somasundaram, G., Papanikolopoulos, N.: Multi-camera Human Activity Monitoring. Journal of Intelligent & Robotic Systems 52(1), 5–43 (2008)
12. Zouba, N., Boulay, B., Bremond, F., Thonnat, M.: Monitoring Activities of Daily Living (ADLs) of Elderly Based on 3D Key Human Postures. In: Caputo, B., Vincze, M. (eds.) ICVW 2008. LNCS, vol. 5329, pp. 37–50. Springer, Heidelberg (2008)
13. Ayers, D., Shah, M.: Monitoring Human Behavior from Video Taken in an Office Environment. Image and Vision Computing 19(12), 833–846 (2001)
14. Hsu, C., Lin, C.: A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks 13(2), 415–425 (2002)
15. Weston, J., Watkins, C.: Support Vector Machines for Multi-class Pattern Recognition. In: Proc. Seventh European Symposium on Artificial Neural Networks (ESANN 1999), pp. 219–224 (1999)
16. Kosov, S., Thormählen, T., Seidel, H.-P.: Accurate Real-Time Disparity Estimation with Variational Methods. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Wang, J.-X., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnação, M.L., Silva, C.T., Coming, D. (eds.) ISVC 2009, Part I. LNCS, vol. 5875, pp. 796–807. Springer, Heidelberg (2009)
17. Phung, S., Bouzerdoum, A., Chai, D.: Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. IEEE Trans. PAMI 27(1), 148–154 (2005)
18. Joachims, T.: Making Large-Scale SVM Learning Practical. In: Advances in Kernel Methods - Support Vector Learning, pp. 169–184. MIT Press (1999)
19. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining, pp. 1–34. AAAI/MIT Press, Menlo Park, Calif. (1996)
20. Shotton, J., Fitzgibbon, A.W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-Time Human Pose Recognition in Parts from Single Depth Images. In: 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011), pp. 1297–1304 (2011)
21. Dargie, W., Poellabauer, C.: Fundamentals of Wireless Sensor Networks. John Wiley & Sons, Chichester (2011)

Sleep Musicalization: Automatic Music Composition from Sleep Measurements

Aurora Tulilaulu[1], Joonas Paalasmaa[1,2], Mikko Waris[2], and Hannu Toivonen[1]

[1] Department of Computer Science and HIIT, University of Helsinki, Finland
tulilaulu@gmail.com, hannu.toivonen@cs.helsinki.fi
[2] Beddit.com Ltd, Espoo, Finland
{joonas.paalasmaa,mikko.waris}@beddit.com

Abstract. We introduce data musicalization as a novel approach to aid analysis and understanding of sleep measurement data. Data musicalization is the process of automatically composing novel music, with given data used to guide the process. We present Sleep Musicalization, a methodology that reads a signal from state-of-the-art mattress sensor, uses highly non-trivial data analysis methods to measure sleep from the signal, and then composes music from the measurements. As a result, Sleep Musicalization produces music that reflects the user's sleep during a night and complements visualizations of sleep measurements. The ultimate goal is to help users improve their sleep and well-being. For practical use and later evaluation of the methodology, we have built a public web service at `http://sleepmusicalization.net` for users of the sleep sensors.

1 Introduction

Understanding data is a central goal of statistics and data mining. Numerical or textual representations can convey very exact results of data analysis, but they are not always the most convenient form for the consumer of the results. Different charts, and data visualization in more general, have proven to be a powerful way of illustrating general properties, distributions, trends and patterns in data, even if details may be missed. But how to represent data analysis results so that they give the user a deeper feeling of the data, perhaps even an emotional one?

In this paper, we consider sleep analysis as a data analysis problem, and propose a novel data analytical approach for it: *data musicalization*. We define data musicalization as the process of automatically composing music from given data, with the goal of perceptualizing it. Figuratively speaking, the data is used to inspire a composition algorithm that produces a novel piece of music. This is in contrast to classical data sonification where data is just mapped to sounds with little if any intention to generate music, and typically with no control at all over the musicality of the result.

Measuring and tracking one's sleep over extended periods of time can help improve sleep and well-being. Sleep analysis methods were previously developed

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 392–403, 2012.
© Springer-Verlag Berlin Heidelberg 2012

for clinical use to help patients with sleep disorders. State-of-the-art sensor technology now allows unobtrusive sleep analysis at home, with two implications for data analysis. First, deriving high-level sleep data, e.g., sleep stages from the physiological sensor data is a complex data analysis task. Second, the users are ordinary people who want to get a more analytical feeling or their sleep, not sleep doctors specialized in reading hypnograms (charts of sleep stages) and actigrams (charts of movements).

The main contribution of this paper is to propose data musicalization as a tool to aid data analysis and understanding. The proposed methodology, called *Sleep Musicalization*, consists of non-trivial data analysis methods for sleep analysis on one hand, and of automatic composition of music from the sleep analysis results on the other hand. For instrumentation, we use a modern, commercially available mattress sensor[1] that can be used to detect sleep stages as well as respiration, heart rate and movements. We have described these sleep analysis methods in detail elsewhere [1,2] and in this paper we just give a brief overview of them.

Based on the sleep information, a piece of music is automatically composed. The goal is that the music reflects the structure of sleep during the night and gives the user a feeling of her sleep. We give stochastic composition algorithms to produce the harmony, melodies, and rhythm of music, based on musicological principles. Their novelty is not as much in the quality of the music *per se* as in making the composition reflect the given attributes of sleep.

We have set up a web service, `http://sleepmusicalization.net`, that supports end-to-end sleep analysis and musicalization. This service also allows us to test the approach with real end users. Some example compositions shared by the users of the service are publicly available.

This paper is structured as follows. We start by reviewing background and related work in Section 2. The methods we use for both sleep data analysis and automatic composition are described in Section 3. We discuss the methods and initial results in Section 4. Section 5 contains concluding remarks. An appendix contains informal definitions of some musical terms.

2 Background and Related Work

2.1 Sleep Measurement

The aim of medical sleep measurement is to divide sleep into *sleep stages* and diagnose various sleep disorders. The standard method for measuring sleep stages for medical purposes is *polysomnography*, which involves attaching biopotential electrodes to the head. Based on the measured signals, sleep can be divided into five sleep stages: wakefulness, REM (rapid eye movement) sleep and three categories of non-REM sleep: N1, N2 and N3 [3]. Stages N1 and N2 are called light sleep and N3 deep sleep. The classification into sleep stages, also known as *scoring*, is normally done manually by a trained sleep technician. Additional

[1] `http://www.beddit.com`

respiration sensors are typically used for diagnosing sleep-related breathing disorders. Polysomnography costs hundreds of euros per measurement night and is not commonly available.

The other widely used medical sleep measurement method is actigraphy, where the patient wears a wrist accelerometer typically for one week, 24 hours a day. Actigraphy measurement is convenient for the patient, but has limited diagnostic capability. It can assist in determining sleep patterns in healthy adult populations and diagnosing circadian sleep-wake disorders [4].

There also are non-medical sleep measurement products for self-help use. *Zeo Sleep Manager* measures *electroencephalography* (EEG) as well as actigraphy with a wireless sensor on the forehead, and classifies sleep into wakefulness, REM sleep, light sleep and deep sleep. The device has been on the market for several years and has validated accuracy [5]. *Fitbit*, *BodyMedia FIT*, *Jawbone UP*, *SleepTracker*, *LARK* and *WakeMate* are wrist actigraphy devices for sleep-wake classification. Classifying sleep periods into REM, light and deep varieties is not performed, because the wrist measurement provides insufficient information.

The sensor that we use for sleep measurement, Beddit, differs from other self-help devices in that the measurement is fully unobtrusive: the data is acquired with a thin force sensor placed under the mattress. Other devices require wearing a sensor in the wrist or on the forehead, which may deteriorate sleep quality.

With most of the self-help devices, sleep information is presented visually to the user in a smartphone interface and a web service. We are not aware of other attempts to produce music from sleep data.

2.2 Music Generation from Physiological Signals

Sonification, i.e., the use of sounds to convey information is a wide area of research [6]. We here focus on methods that sonify physiological signals (such as the ones used to measure sleep).

Research on generating music from physiological measurements mostly deals with EEG and *electrocardiography* (ECG). The first device to generate music from physiological measurements was the "encephalophone", from the 1940s [7]. It generates audio from measured brain waves for both medical and musical purposes. Various artists have since made similar instruments. For example, Krzysztof Penderecki's *Polymorphia* (1963) was based on encephalographic pitch notation derived from EEG data.

The first commercial biomusic production environment, BioMuse, was introduced in late 1980s [8]. Notable contemporary performances include, to name a few: underwater-EEG-ECG *DECONcert* [9], *Multimodal Brain Orchestra* [10] as well as *Baroque Duet for Cello, Violin, 2 Hearts (ECG) and 2 Minds (EEG)* by Barrass and Whitmer, performed at the Sound and Music Computing Conference in 2010.

2.3 Automatic Composition of Music

Algorithmic composition of music is a much older phenomenon than computers, and many kinds of formal processes have been proposed to generate music. Already around year 1026, Guido d'Arezzo introduced a technique for generating melodies to accompany a text [11]. One of the most famous examples of automatic composition is Mozart's Musikalisches Würfelspiel, a dice game for generating waltzes. Prior to computers, some mechanical composition machinery also existed [12].

The first computer composition, *The Illiac Suite for String Quartet*, was generated by a program by Hiller and Isaacson in 1956. Current state-of-the-art includes David Cope's program called *Emily Howell* who composes complex music in her own style [13].

Mapping data to sounds or music is an old idea, too. Several attempts have been made in biology. For instance, *Protein music* was obtained by translating coding sequences into sound [14]. A contemporary example with CDs on sale is *Your DNA Song*[2], a service for turning DNA into music by mapping the 22 amino acids to 22 different notes and then playing out the result with 3000 nucleotides per minute. These systems sonify data essentially by translating it to sound. In contrast, we are interested in having music composed automatically, with the data given just for guidance. This way several musical characteristics are under the control of the composition algorithm, not just the data.

Music can be composed using many different approaches and their combinations. Typical examples include cellular automata, fractals, grammars, constraints, pattern matching, and many different stochastic processes, like Markov chains and different uses of distributions. A review of these approaches is outside the scope of this paper, but see, e.g, [12] for an overview. We use relatively simple Markov chains and random walks, and the novelty is in making the music reflect sleep.

Approaches to automated composition can also be classified by their interactivity with the user. Some systems generate accompaniments for the user's music, or generate short parts of melody [12]. Some systems also accept feedback from the user [12]. One of the better known examples of this is the *band in a box* software that generates accompaniments with the chosen chords in a style the user prefers. Other systems, like the one to be described in this paper, do the whole composition process autonomically.

3 Methods

We next present the Sleep Musicalization methodology in three parts. First, we describe the data analysis methods we use for sleep measurement. Since these methods have been reported elsewhere, we here just give an overview of them. Second, we present the composition algorithm that produces music from sleep

[2] `http://www.yourdnasong.com/`

measurements. Third, we describe the implementation of Sleep Musicalization as a web service.

The Python source codes of the musicalization[3] and web service[4] components are available for download.

3.1 Sleep Measurement

Sensor Data Collection The raw signal is acquired using a thin, flexible piezoelectric force sensor by Beddit, measuring 70 by 4 cm, placed under the mattress topper or mattress. The signal from the sensor is sampled at 140 Hz. An excerpt of the measured signal is shown in Figure 1.

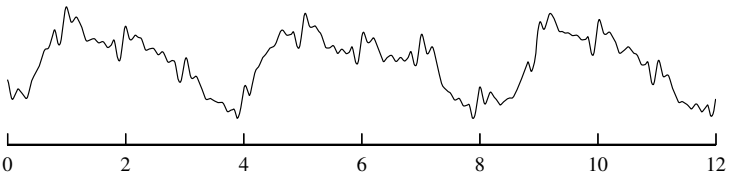


Fig. 1. A 12-second signal excerpt from the Beddit sensor. Respiration is the cyclic low-frequency waveform with 4-second period. The heartbeats occur around every second.

A tailor-made miniature Linux computer receives the signal, performs analog-to-digital conversion, and sends the signal to a web service for analysis. Around 300 kilobytes of compressed data is sent per a measured hour of sleep. In addition to the force signal, information related to the sleeping environment is measured every minute: temperature, noise level and brightness.

Data Analysis for Physiological Parameters. The force signal contains heartbeats as well as respiratory and movement activity. In Figure 1, the overall trend corresponds to respiration cycles, while heartbeats can be detected from the finer-grained fluctuations of the force signal. Sleep analysis greatly depends on heart rate variability, and this can be obtained from the signal in the form of inter-beat intervals. The measurement of heart rate from a force signal is called *ballistocardiography*, and it is an established technique [15]. We have shown elsewhere that heartbeat intervals can be detected accurately from our current sensor data [2]. The low-frequency phenomenon of the force signal is also analyzed [1], in order to detect the length of each respiration cycle and variations in them. This information is also needed in the sleep analysis.

Sleeper movements are detected by analyzing individual abrupt changes in the force signal. The resolution of movement detection is set at three seconds, i.e., a movement is recorded to occur at most every three seconds.

[3] `https://github.com/Tulilaulu/Sleep-musicalization`
[4] `https://github.com/beddit/sleep-musicalization-web`

Data Analysis for Sleep Stages Now, based on heart rate and respiration variability as well as movement information, the time spent in bed is segmented into periods of wakefulness, REM sleep, light sleep and deep sleep.

The classifier is based on a generative model of heart rate, respiration and movement features given sleep stage. Transitions between sleep stages are described with Markov models. On a high level, the generative model works as follows. The levels of heart rate and respiratory variation are highest during wakefulness and REM, slightly lower in light sleep, and lowest in deep sleep. Wakefulness has a high level of movement activity, REM and light sleep stages permit some movements, while in deep sleep the level of activity is very low.

A full validation of the classifier is in progress; the details of the classifier and its validation will be published elsewhere.

The result of sleep analysis thus consists of the structure of the sleep (hypnogram: segments of different sleep stages) as well as information about the heart-rate and movements of the sleeper during the night. Figure 2 is a screenshot from the Beddit service illustrating what the analysis results look like when visualized.

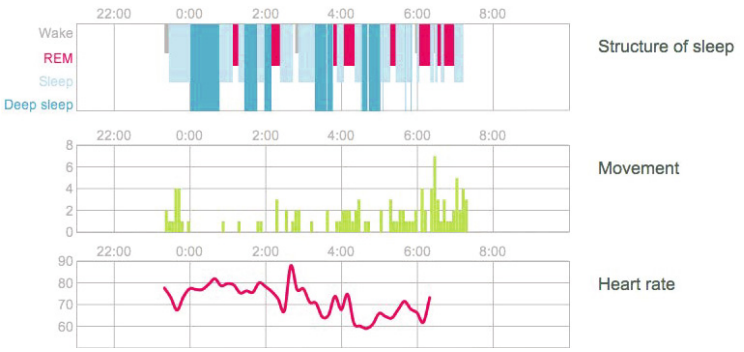


Fig. 2. The key sleep information for one night

3.2 Automatic Composition of Music

In Sleep Musicalization, a novel piece of music is automatically composed from the sleep measurements of one night at a time. We currently use the following information: sleep stages, heart rate, and movements. The composition method has been designed to make these aspects clearly audible in the music, as will be described below. The piece is roughly compressed to ratio 1:120, i.e., one second of music corresponds to two minutes of sleep or, in other words, eight hours of sleep results in a song of about 4 minutes.

Preparations. Before the actual composition starts, sleep stages that are very short are removed to give more stability to the music.

A scale[5] is chosen at random by picking the starting point of the scale and its type, a major scale or a harmonic minor scale, the most common Western scales of music. The default is major, however pieces in minor can be generated if the user so wishes.

The method next randomly decides if the bars will have 3 or 4 beats. The length of a bar is significant musically; the method also processes music one bar at a time and associates one chord with each bar.

Generation of a Chord Sequence. The sequence of chords is first generated with a Markov chain. It is hand-coded to have meaningful transition probabilities between chords. (Note that the Markov chains used to generate chords or other musical aspects below have nothing to do with the Markov models of sleep stages used in sleep analysis.)

For simplicity, the Markov model has maximum degree 2, i.e., it only takes into account two previous chords when generating the next one. Not all pairs of two last chords have a transition distribution assigned to them and in those cases the next chord is chosen using only the previous chord. At the moment we only use chords that consist of exactly three notes that are in the chosen scale. (In music in general, chords are often used that are not native to the key. We only use 7 elementary chords per piece to cut back on complexity.)

The probabilities of the chords have in general been chosen to reflect their frequencies and dominance in music using the given key. Exceptions to this rule are transition probabilities in the Markov chain that make sure that chords that need to be resolved will be resolved (i.e., from the dissonant seventh chord of the key, it is best to move to the first chord). Once the sequence of chords has been generated for the piece of music, the method moves on to compose a melody.

Melody Generation. Intervals can be classified into steps and skips. Major and minor seconds and unison are steps. Skips, in turn, are any larger intervals of up to an octave up or down. This division to steps and skips is useful because small intervals appear more commonly in music than big ones.

The intervals that form the melody are generated using another Markov chain. The Markov chains are constructed so that the total probability of steps is roughly equal to the total probability of skips. Additionally, the probabilities of transitions between some specific intervals have been set. If the last interval was a skip upward, the next is either a step or a skip downward and if it was a skip downward, it is followed by a skip upward or a step. If the resulting note is not in the scale or is in bad dissonance with the current chord it is rejected, and the Markov chain is used to generate a new interval instead.

In order to have some internal coherence in the music, the melody generation procedure uses musical themes. Different sleep stages have a different theme each, so they will have more clear identities in the music. (As a special case, the state of being absent from bed generates silence.) As the very first step of melody generation, a theme of two bars is generated for each sleep stage for later use.

[5] See Appendix for a glossary of some musical terms.

The actual melody is composed for one sleep stage segment at a time. The melody for a segment starts with the theme of the sleep stage, after which the rest of the segment is filled with the melody generation method.

Generation of Rhythm. Note lengths are generated with a Markov chain of degree 1, but the degree will be increased later to get better results. The values are chosen so that they favour multiples of the same length as it is common for especially shorter note lengths to appear in groups. They also favour rhythmic patterns that match the beats. For instance, if the last note is 1.5 beats long, the next one is likely to be 0.5. However, if the remaining length of the segment at hand is less than 2.5 beats, then the note will last until the end of the segment. This avoids the practical problem of going over the segment length, but it is also a good way to give the last note of the segment a bigger probability of being longer. The rhythm also slightly varies with respiration. Higher respiration rate make short rhythms sligthly more likely and slower respiration rate longer rhythms a bit more likely.

Generation of Accompaniment. The accompaniment is also generated sleep stage segment by segment. It also changes according to the sleep stage: for each stage, there is a different generating function. The awake accompaniment just consists of short notes on the base of the chord. The one for REM sleep consists of quite a fast pattern using all the notes of the chord. The accompaniment of light sleep is a generic up-and-down pattern of the notes of the chord, slightly slower than in REM sleep. Finally, the deep sleep accompaniment consists of long notes in a slow pattern. All of the generators use only notes of the current chord. In the accompaniment, slight emphasis is added on the first beat or all beats, depending on sleep stage.

Volume to Reflect Sleeper Movements. The actigram data (movement information) is preprocessed so that it contains information on the density of the movements for each beat of music. If there are a lot of movements, the volume rises and if there are very little movements, the volume goes down. (In addition, there can be other changes to volume, such as emphasis on the beats.)

Regulation of Tempo. The tempo (beats per minute) varies during the music, reflecting the heart rate. The heart rate is transformed to make changes more pronounced. First, it is approximated in a piece-wise constant manner so that tempo changes are discrete. This is implemented by taking a cumulative moving average and comparing it to the current constant approximation. If the absolute difference exceeds a pre-determined threshold, then the current value is taken as the next constant approximation. Second, the starting times of each point in the averages and the tempos attached to them are processed so that slower tempos get less beats and faster tempos get more beats to keep the song moving through the sleep stages at a constant pace. Then, the relative magnitude of changes is amplified. The heart rate estimate is multiplied by four and 180 is subtracted form the result. Another interesting option for regulation of tempo would be to use respiration rate, which is also available from the sleep analysis.

Sound Synthesis. We use Kunquat[6] for the actual sound generation, given the composed music. Kunquat uses an event-based approach to represent music. There are different events for starting a note, stopping a note, changing the tempo and changing the force of the note. The pitch of a note is represented as a number where 0 is equal to 440 Hz and an addition or subtraction of 100 moves the pitch a minor second. The program also does not deal with time as seconds, but uses beats as its main time unit. Our implementation uses these units internally. In Kunquat, music is represented in tracks, where only one note can be playing at a time. For this reason, the melody and accompaniment are generated to their own tracks. After the composition of music is ready, it is given to Kunquat which produces an audio file.

3.3 Implementation as a Web Service

We have implemented a web service (`http://sleepmusicalization.net`) where sleep music can be generated and accessed conveniently by users of the Beddit device. This is a valuable tool also for research, as it will allow end user studies of the Sleep Musicalization methodology and the music generated. At the web site, others interested can listen to music published by Beddit users.

In the end-to-end system, a user of the Beddit sleep tracking service first grants the sleepmusicalization service access to her data (Figure 3). This authentication is implemented using OAuth2 protocol. When the user wants to listen to her sleep as music, she initiates the composition process at either the sleepmusicalization or Beddit site. The sleepmusicalization service then retrieves sleep measurement data from the Beddit service and runs the music composing algorithm. The generated music can then be listened to on a web page, or published for others to listen and share in social media.

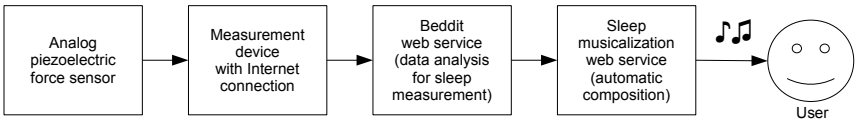


Fig. 3. The end-to-end flow of sleep information, from an analog force signal to sleep measurements to automatically composed music

4 Results and Discussion

The goal of Sleep Musicalization is to provide a novel way to perceptualize sleep measurements, complementing their visualizations. The Sleep Musicalization method as described above has been implemented and can be freely used at `http://sleepmusicalization.net`, where example results are also available for listening.

[6] `http://kunquat.org`

At this phase of the research, it is too early to draw conclusions about the value and success of the methodology. Our own, subjective opinion is that the key characteristics of sleep, especially the sleep stages, are already perceptualized well with the current method. An objective evaluation and validation is yet to be carried out.

The quality of the music could be improved a lot. The current composition algorithms are relatively simple and, as a result, the musicality of the melodies is still questionable. A large body of literature on composition algorithms exist (see, e.g., [12]) and could be used to implement better methods. It is likely that mixing other methods with the ones used now will result in better compositions. One possibility could also be to program the system to learn from feedback or other compositions, as has been implemented elsewhere [13].

There are currently some practical limitations which could in principle be easily circumvented. For instance, we only use three tracks, i.e., at most three notes are playing at the same time. For the quality of the result this is not a crucial point, however. Many of the choices we have made in the music composition have viable alternatives, like adding other scale types, allowing a bar to have some other number of beats than 3 or 4, or accepting notes that are not in the scale.

The current set of sleep measurements contains the most important characteristics of sleep, but additional data would also be available. The most relevant additional physiological measurements are heart rate variation and respiration rate variation. We plan to incorporate such information, e.g., in the average pitch or volume of the music, or in the frequency of the rhythm.

The Beddit device, of which the force sensor is just one part, also provides information about the environment. Ambient levels of light, noise, and temperature could be useful additions, as possible explanations of some sleep events.

From a data mining and learning perspective, a fascinating challenge would be to learn a musical style from a given corpus of musical scores. Given a set of songs, say, of Beatles, the system should learn to produce new music in the style of Beatles, and be able to adjust it according to sleep measurements. This would allow, in principle, the user to select the style of music in which her sleep is musicalized.

5 Conclusion

We have proposed data musicalization as a novel way to represent data or data analysis results to the user. We presented Sleep Musicalization, a methodology combining state-of-the-art sleep measurement sensors and methods with automated composition algorithms. Sleep Musicalization produces music that reflects the user's sleep during a night, with the aim of complementing existing visualization methods for sleep measurements. The ultimate goal is to help users understand and track their sleep, in order to improve their sleep and well-being. We hope that musicalization of sleep measurements can make this tracking process more attractive and fun for some users.

Data musicalization is based on composing music, not just mapping data to sounds as is usual in sonification. In this paper, we presented stochastic composition algorithms that are constructed specifically for sleep measurements.

The next phase in this work will be validation of the Sleep Musicalization approach. How much and what kind of information about sleep is actually conveyed to the users? Does it provide some additional value to the users, complementing current graphical results? The web service will allow us to gather actual user experiences and feedback. Other future work will include improving composition methods.

The sleepmusicalization.net service has now been designed for the Beddit sensor. It should be conceptually relatively easy to adapt the composition algorithm to work with sleep data from sources other than Beddit, e.g., Zeo Sleep Manager.

An interesting research question is if the data musicalization approach is applicable to other data sets and application areas. In the current composition algorithm, we already perceptualize discrete state variables (sleep stages), continuous variables (heart and respiration rates), as well as event data (sleeper movements). Could similar ideas and methods be applied to perceptualize other temporal data sets?

Acknowledgments. We would like to thank Tomi Jylhä-Ollila for help with the Kunquat system.

This work has been supported by the Algorithmic Data Analysis (Algodan) Centre of Excellence of the Academy of Finland (Grant 118653).

References

1. Paalasmaa, J., Leppakorpi, L., Partinen, M.: Quantifying respiratory variation with force sensor measurements. In: 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2011, pp. 3812–3815 (2011)
2. Paalasmaa, J., Waris, M., Toivonen, H., Leppakorpi, L., Partinen, M.: Unobtrusive Online Monitoring of Sleep at Home. In: 34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2012, pp. 3784–3788 (2012)
3. Iber, C., Ancoli-Israel, S., Chesson, A., Quan, S.F.: The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications. American Academy of Sleep Medicine (2007)
4. Morgenthaler, T., Alessi, C., Friedman, L., Owens, J., Kapur, V., Boehlecke, B., Brown, T., Chesson Jr., A., Coleman, J., Lee-Chiong, T., Pancer, J., Swick, T.: Practice parameters for the use of actigraphy in the assessment of sleep and sleep disorders: An update for 2007. Sleep 30(4), 519–529 (2007)
5. Shambroom, J.R., Fabregas, S.E., Johnstone, J.: Validation of an automated wireless system to monitor sleep in healthy adults. Journal of Sleep Research 21(2), 221–230 (2012)
6. Kramer, G., Walker, B.N.: Sound science: Marking ten international conferences on auditory display. ACM Transactions on Applied Perception (TAP) 2(4), 383–388 (2005)

7. Henry, T.K.: Invention locates hurt brain cells. New York Times 21 (March 2, 1943)
8. Knapp, R.B., Lusted, H.S.: A bioelectric controller for computer music applications. Computer Music Journal 14(1), 42–47 (1990)
9. Mann, S., Fung, J., Garten, A.: DECONcert: bathing in the light, sound, and waters of the musical brainbaths. In: ICMC 2007: International Computer Music Conference (2007)
10. Le Groux, S., Manzolli, J., Verschure, P.F.M.J.: Disembodied and collaborative musical interaction in the multimodal brain orchestra. In: NIME 2010: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 309–314 (2010)
11. Kirchmeyer, H.: On the historical constitution of a rationalistic music. Die Reihe 8, 11–24 (1968)
12. Roads, C.: The computer music tutorial. The MIT Press (1996)
13. Muscutt, K.: Composing with algorithms: An interview with David Cope. Computer Music Journal 31(3), 10–22 (2007)
14. King, R.D., Angus, C.G.: PM – protein music. Bioinformatics 12(3), 251–252 (1996)
15. Pinheiro, E., Postolache, O., Girao, P.: Theory and developments in an unobtrusive cardiovascular system representation: Ballistocardiography. Open Biomedical Engineering Journal 4, 201–216 (2010)

Appendix: Glossary of Some Musical Terms

scale a series of notes differing in pitch according to a specific scheme; major scale and harmonic minor scale are typical Western scales

key a major or minor scale

minor second (or semitone) the smallest interval in use in Western music, e.g., the distance between two closest keys in a piano

major second (or whole tone) an interval as big as two semitones

octave an interval as big as 12 semitones, the distance between two notes of the same name (for example c to c).

chord a combination of a minimum of three notes; the chords in this paper are made by taking a base note for the chord and adding the notes that are 2 up and 4 up from it in the given scale.

Engine Parameter Outlier Detection: Verification by Simulating PID Controllers Generated by Genetic Algorithm

Joni Vesterback[1], Vladimir Bochko[1], Mika Ruohonen[1], Jarmo Alander[1], Andreas Bäck[2], Martin Nylund[2], Allan Dal[2], and Fredrik Östman[2]

[1] Department of Electrical Engineering and Energy Technology, University of Vaasa
FIN-65101 Vaasa, Finland
[2] Wärtsilä Finland Oy, FIN-65101 Vaasa, Finland

Abstract. We propose a method for engine configuration diagnostics based on clustering of engine parameters. The method is tested using simulation of PID controller parameters generated and selected using a genetic algorithm. The parameter analysis is based on a state-of-the art method using multivariate extreme value statistics for outlier detection. This method is modified using a variational mixture model which automatically defines a number of Gaussian kernels and replaces a Gaussian mixture model.

Keywords: multivariate outlier analysis, PID controller, genetic algorithm, simulation.

1 Introduction

Gas and diesel engines working in power plants, vehicles, and ships have high maintenance costs. Nowadays engine manufacturers use a condition monitoring approach for engine health monitoring system to determine the engine state under certain operating conditions [6]. Condition monitoring and similar approaches are efficiently used in such applications as gas-turbine aerospace engines [17], patient health monitoring [9], anomaly detection in transportation corridors [1], PID control and automatic tuning depending on engine type and application in trucks [13], and study of performance of a turbocharged diesel engine operating under transient load conditions [15]. Thus, intensive studies have been made to reveal the mechanical condition of engines.

In this paper a method for engine configuration diagnostics is developed. The method includes visualization of parameter sets for PID controllers, data clustering and detection of outliers. The feasibility of the proposed method is tested by stochastically generated parameters for PID. This is done by using the genetic algorithm (GA) for which we introduce a multiobjective index of goodness.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 404–415, 2012.
© Springer-Verlag Berlin Heidelberg 2012

2 Related Work

The problem of engine configuration diagnostics can be solved using approach known as detection of outliers. The approach determines test data as normal or abnormal by a model of normality designed from training data. This technique is especially useful if a large number of samples of normal behavior are available while samples of abnormal behavior are absent or rare. Therefore the best way to build the outlier detector is to use the knowledge of normal group of engines because the information about configuration of these engines is usually available from the technical support department of companies. There is an overview of many publications including classification, clustering, statistical, nearest neighbor-based techniques related to the field of detection of outliers [5]. However, a typical problem of those methods is the heuristic selection of novelty threshold separating the samples as normal or abnormal. This threshold has no probabilistic interpretation and in addition not suitable for multimodal and multivariate data. One possible solution can be found using an approach based on the classical Extreme Value Theory (EVT) [16]. However, the classical EVT technique is inaccurate in the case of multidimensional and multimodal data.

The recent work by Clifton et al. [7] introduces the Multivariate Extreme Value Statistics (MEVS) which overcomes the disadvantages of classical EVT and makes EVT use possible with multivariate and multimodal distributions. The approach uses Gaussian Mixture Model (GMM) fitting the data to create the probabilistic model and MEVS to define the threshold for outliers. The MEVS approach uses the model of normality to perform novelty detection in the probability space. There are two variants of MEVS approach for multivariate and multimodal problem: one that uses sampling and numerical computation of the threshold, and another that uses an approximative solution. The latter approach, the most computationally attractive, is based on fast multivariate, unimodal closed form solution that is an approximation of the multimodal problem.

We use this approximative multivariate, multimodal solution to find MEVS threshold for solving the task of outlier detection [7]. We replaced GMM by Variational Mixture Model (VMM) [4], [18] that gives the following advantages: automatic adjustment of the number of clusters, no need to use cross-validation and no problems related to singularities. The analyzed data, both normal or abnormal, is modeled using GA and a set of criteria. For more references on GAs for controllers see e.g. bibliography [2], for GAs in outlier detection see e.g. bibliography [3], and for GAs in testing software see e.g. review [11].

To the authors' best knowledge this is the first study related to the analysis of engine setup parameters for engine configuration diagnostics. The main contribution of the paper is two-fold: we propose a modified variant of MEVS using VMM that simplifies and improves the algorithm and a multiobjective criterion for GA giving a realistic simulation results for PID data.

3 Engine Parameter Modeling

3.1 Index of Goodness

We introduce the index of goodness that helps us to estimate the overall goodness of the model parameters and identify them as normal and abnormal.

First, we consider an input or reference signal described by the ideal step function and a response, reaction of the device on this input signal. Then several criteria describe the relationship between the reference signal and response [20]: overshoot, settling time, steady-state error, and stability. Overshoot happens when the response exceeds the reference signal. The settling time is the elapsed time from the moment when the reference signal starts to the time when the device response is close to the final value. Settling time tells how fast the system reacts to change [8]. The difference between the signals at the end of the settling time is characterized by a steady-state error. Stability is needed to exclude output divergence.

Our index of goodness consists of the sum of absolute error or difference error e_{dif}, i.e. the sum of absolute differences between the reference signal and response, maximum error e_{max}, i.e. the maximum absolute error between response and reference after the first response-reference intersection, and final error e_{fin}, i.e the steady-state error. Settling time is accounted by a difference error and a final error. To resolve the stability problem we decided to choose only stable parameters, hence any unstable parameter is discarded. According to [14] the genetic algorithm tend to converge towads stable population in this problem setting.

The difference error, maximum error and final error are also mentioned in many works as good ways to estimate the performance of a PID controller [8], [20]. The difference error would give an estimate on how well the system worked overall and an indication of the stationary error. The maximum error is important, its too high value might even cause breaking of the controlled machine [8].

As the index of goodness I_g we use one similar to that given in [19]:

$$Ig = \frac{1}{ae_{dif} + be_{max} + ce_{fin}}, \tag{1}$$

where a, b, and c are constant weight coefficients. We select their values in agreement with [19] as follows: $a = 0.02$, $b = 100$, and $c = 1$. The value for a must be small, because the sum of absolute error is usually much greater than the other factors. The value for b should be large due to the importance of the maximum error. A large value of maximum error may break a machine or cause other problems for the system. Though the final error tells us how well the controller works overall, it also relates to an unstable system.

The GA uses the index of goodness in the fitness function to give a fitness value for each individual. The individuals are selected by the GA based on these values. The index of goodness is made so that the GA would produce a group of overall good and realistic parameters.

The index of goodness takes into account the steady state error, the response time using the sum of absolute error and steady state error, and the maximum error. The maximum error is there because large errors from the reference value costs more. Since the GA uses a single fitness function we take a weighted mean from all of these values. A similar function was used in the study, where the GA combined with fuzzy logic was employed to configure PID controllers [19]. This function produced good results to find overall good parameters in that work.

GA uses the index of goodness to compute PID parameters used as candidates for further selection of parameters and partition them into two groups: normal and abnormal. Therefore for each PID parameter triplet computed by GA we build the plot showing the reference signal and response. Finally, the analysis of these plots suggests data partition.

For assessing normality using plots we used a second set of criteria: overshoot, settling time and the sum of absolute error (Fig. 1). Each criterion had its own threshold. If a controller would exceed at least one of these thresholds, it would be considered abnormal.

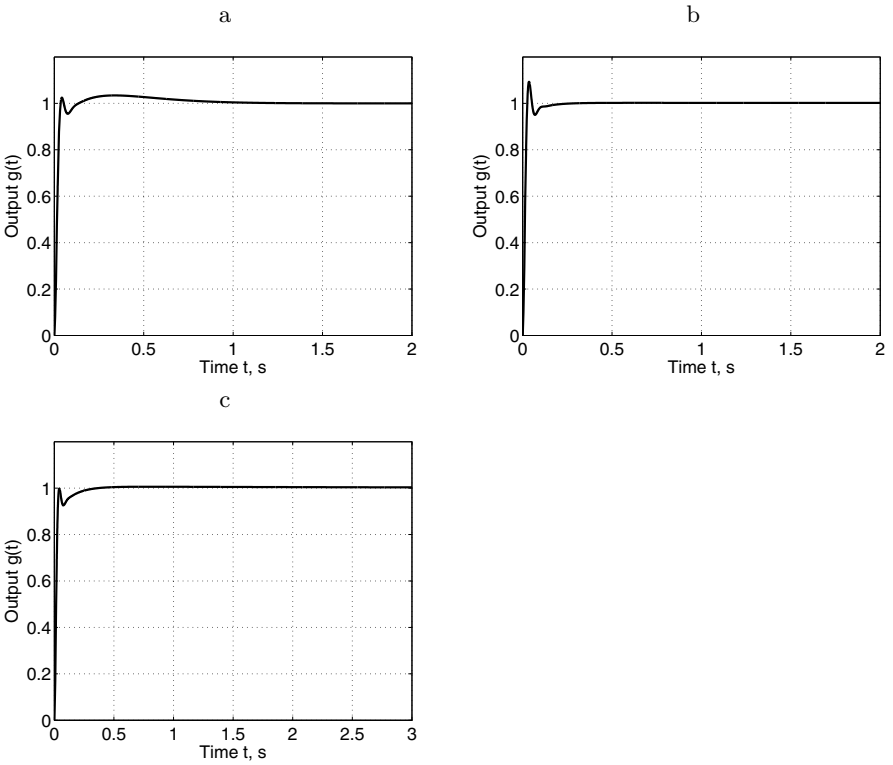


Fig. 1. Second set of criteria. a) Settling time value 0.61s. b) Overshoot value 0.093. c) sum of absolute error 417.87.

We defined the thresholds to be reasonable and efficient. The maximum overshoot was set to 10% [20], settling time was put to reach 2% [10], [20] within 0.75s and the sum of absolute error was set to 450 [8]. We arrived at these by comparing the plots from the PID parameters and engineering intuition. The values can be changed for different applications, depending on the user's preference.

3.2 Simulation

GA uses a way of ranking individuals. We do it by simulation using Simulink which is a tool for Matlab and has several different libraries of blocks for drawing graphical models for simulation purposes. The model used by GA for simulation is presented in Fig. 2. The model consists of `step`, `PID`, `process`, `clock`, `abs`, `mux`, and `simout` blocks.

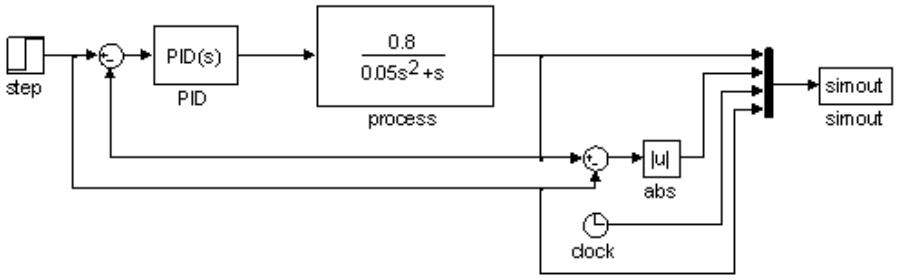


Fig. 2. Simulink model used by GA

The `step` block represents the reference value, which the process tries to achieve. It produces the unit step function and has its own sampling time, which was set to zero, i.e. continuous time.

The `process` block represents the engine model. It has a given second order transfer function [12]:

$$\frac{0.8}{0.05s^2 + s}, \tag{2}$$

where s is the Laplace variable. We use this simple transfer function throughout all the simulations. The real process behind this model is a diesel engine with an actuator.

The parameters for the `PID` block are generated by the GA separately for each simulation. The `PID` block uses a default value of 100 for the derivative filter coefficient. A parallel controller form is used in all simulations.

The `clock` gives the time value at each sampling point. The absolute value block $|u|$ provides the absolute value of input. The `muxer` block is shown by the black thick vertical line segment with four arrows going in. These four input signals are presented as vectors with a size 50001. The `muxer` combines the four vectors into an array.

The `simout` block receives the data from the `muxer`. The four vectors packed into an array are as follows: system output $g(t)$, absolute error $e_{abs}(t)$, time t for each sampling point, and reference $y(t)$. The model was run in continuous time mode but the `simout` block had a sampling time of 10^{-4}s. Simulation time was 5s, which produced 50001 data points for each simulation. GA uses the data from the `simout` block to calculate the index of goodness. Then the maximum overshoot is a maximum of $g(t) - y(t)$. The maximum error is a maximum of $e_{abs}(t)$ after the first response-reference intersection. The sum of absolute error was calculated by summing all of the data points from $e_{abs}(t)$. The settling time was found by searching the time moment when the values of $e_{abs}(t)$ are within 2% of the reference point and never going over it again during the simulation.

Fig. 3 shows how the Simulink model (Fig. 2) is used by the GA. In the initialization step, the GA is given: population size, number of generations, step time, step size, initial value, sample time for step, process transfer function, derivative filter coefficient and sampling time for `simout` block. The GA optimization includes generation of a new population, calculating fitness and then selection of PID parameters.

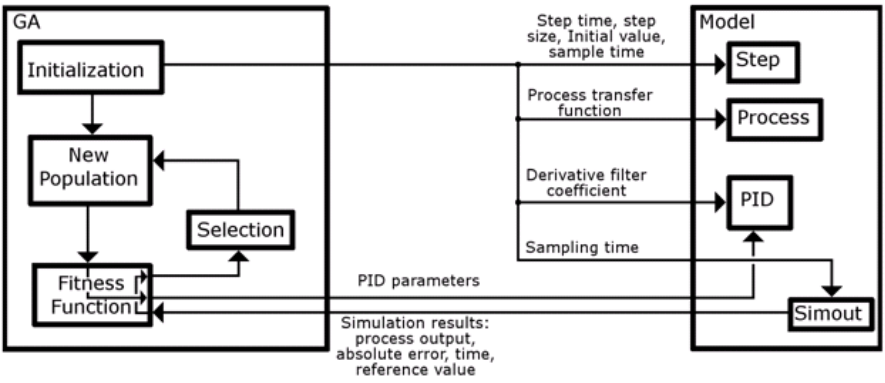


Fig. 3. Flow diagram for GA and Simulink model

During the first step the genetic algorithm using the index of goodness finds representative parameters which are stable. The normal and abnormal parameters were taken from the representative parameters using the second set of criteria. If all criteria are satisfied then the parameters are normal otherwise they are abnormal.

The GA using the index of goodness was run for 50 generations. The population size was 1000. As a result about 22 000 unique configurations (PID triplets) were obtained. Then using the second set of criteria the normal and abnormal parameters were finally selected. We obtained 104 PID triplets (normal) for training, and 104 PID triplets (normal) and 104 PID triplets (abnormal) for testing. The parameter values are taken at resolution 0.5 and lie in the range $[0.5, 1500]$ for P and $[0, 1500]$ for I and D.

4 Data Analysis

4.1 Mapping to the Probability Space

Our feature space is obtained by mapping the PID (3D) values into the subspace spanned by the first two eigenvectors, $n = 2$. This dimensionality is well suited for visualization and keeps the data topology.

The key idea of the MEVS approach is to modify the univariate EVT for multivariate data [9]. For that a mapping from the feature space to the probability space is made. First, we estimate the probability density function of the input data $f_n(\mathbf{x})$ in the feature space and, then the probability is considered as a new random variable. If we denote the input data, i.e. feature vector, $\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$, then the EVT is used in the probability space $G_n(f_n(\mathbf{x})) \in \mathbb{R}$, where G_n is a new probability distribution defined over probability density values. Our task is to detect the extrema of input data.

4.2 Variational Mixture Model

The mixture model VMM is used to model complex probability distributions. The mixture model consists of components which represent simple distributions. Such a simple probability density is the multivariate normal density $\mathcal{N}(\mathbf{x}/\boldsymbol{\mu}, \mathbf{T})$, where $\boldsymbol{\mu}$ is a mean, and $\mathbf{T}$ is an inverse covariance. The mixture density function is as follows:

$$f_n(\mathbf{x}) = \sum_{i=1}^{M} \pi_i \mathcal{N}(\mathbf{x}/\boldsymbol{\mu}_i, \mathbf{T}_i), \tag{3}$$

where π_i are mixing coefficients, the sum of which is equal to unit, and M is the number of components. Then the probability of the observed data is a function of three sets: $\{\pi_i\}_{i=1}^{M}, \{\boldsymbol{\mu}_i\}_{i=1}^{M}, \{\mathbf{T}_i\}_{i=1}^{M}$, and is called the likelihood function.

The mixture coefficients are optimized using the type 2 maximum likelihood method [4]. This approach finds the relevant number of components and also gives explicit values for means and covariances from the variational parameters. Thus, the approach is particularly useful for density estimation.

4.3 Method

The VMM, i.e. the Gaussian mixture model with the automatically defined number of clusters, fits the input data. After fitting we find $f_n(\mathbf{x})$. Next, we will omit the argument and use f_n instead of $f_n(\mathbf{x})$. Considering the probability densities f_n values as samples of a random variable y ($y = f_n$), the probability distribution is as follows [7]:

$$G_n(y) = \int_{f_n^{-1}(]0,y])} f_n(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{4}$$

where $f_n^{-1}(]0, y])$ is the preimage of $]0, y]$ under f_n.

In our case, when VMM fits the multimodal multivariate data, G_n has not any analytical form. Therefore, the unimodal multivariate model is used for fitting the multimodal multivariate data to provide an approximative solution [7], [9]. The approximative solution defines the probability distribution K_n corresponding to a single Gaussian kernel that accurately estimates G_n for the small values of y that agrees with the large magnitude (extrema) of $\mathbf{x}$. For introducing a novelty score, one may write:

$$1 - K_n(y) = \exp\left[- \left(\frac{y}{c_m} \right)^{\alpha_m} \right]. \tag{5}$$

The scale and shape parameters are as follows:

$$c_m = K_n^{\leftarrow}\left(\frac{1}{m} \right), \tag{6}$$

$$\alpha_m = c_m \frac{k_n(c_m)}{K_n(c_m)}, \tag{7}$$

where $K_n^{\leftarrow}\left(\frac{1}{m}\right)$ is the $1/m$ quantile of K_n, m is the total number of points, i.e. the number of input parameters and k_n is the probability density related to a single Gaussian distribution and obtained by the approximative solution. Eq. 6 is solved by numerical optimization as follows:

$$\arg\min_c \left[K_n(c) - \frac{1}{m} \right]^2. \tag{8}$$

From Eq. 5 we obtain a threshold value as follows:

$$y_m = c_m \left[- \ln(1 - K_n) \right]^{\frac{1}{\alpha_m}}. \tag{9}$$

Eq. 9 shows that given a threshold at $1 - K_n = 0.999$, we find the threshold value y_m for f_n and may draw a contour representing the novelty threshold in the 2D visualization (feature) space. All points (parameters) located inside the region defined by the contour are normal otherwise the points are abnormal.

5 Implementation

We developed and completed the analyzer of diesel engine parameters using Matlab for Wärtsilä Finland Oy. The analyzer is intended to detect outliers during an engine configuration setup. In our primary experiments we used real engine parameters extracting the group of engines with similar parameters and comparing them with engines having different parameters. The results were rather good. We had several engine configuration files containing parameters for more than ten engines each. All parameters used were normal and we had not available abnormal data. Therefore, we simulated normal and abnormal data in our

experiments. Since our analyzer is based on MEVS, it is ready for use, even when we have not any abnormal data. The analyzer works with data which has complicated encoding. The data contains the values of PID parameters and additional numerical and textual information related to these parameters. The user can click the Mouse button on the point that is an outlier in a given visualization panel and see the engine identification. In addition, the user may save the result as an Excel file with full information on outliers.

To avoid the generation of complex data structure and to use a typical number of PID triplets in our experiments we replace the real engine data of the PID triplets by synthetic ones without changing the rest real data structure. For example, for the particular engine parameter presented by an eight element vector where each element is a PID triplet and 13 engines, the number of synthetic PID parameters is 104. This number is used in our experiments. Since the MEVS technique builds the threshold separating normal and abnormal points using the number of available input parameters (Eq. 6), our method is flexible with respect to the number of PID triplets.

At the moment we use a simple strategy to adjust the parameters to obtain the desired performance. After finding outliers we compute the correction values, i.e. the differences between the outliers and the center of mass of the normal data from the training set. For further study we will use more sophisticated approaches.

6 Experiments

The conducted experiments include simulation of parameters using GA with proposed sets of criteria and data analysis. We generated 104 PID triplets (normal) for training, and 104 normal and 104 abnormal PID triplets for testing. Examples of the output signals corresponding to normal and abnormal data are shown in Fig. 4. For example, for abnormal data behavior the output signal has overshoots over 10 percent, settling time too long, and too high overall error.

The simulated data was tested using a modified MEVS approach. Using a training set and a threshold value 0.999 we obtained an area of normal data surrounded by a threshold curve (Fig. 5a). Fig. 5 shows the space spanned by the first two eigenvectors where the principal components (or scores) are scaled to have unit variance. All points inside the area bounded by the threshold curve are normal, outside they are abnormal. Fig. 5a shows the results after training. During fitting VMM may produce varying although similar results. Therefore we run the algorithm several times and select only those results which are obtained with the simplest fitting model, i.e. the model which has the smallest number of Gaussian kernels.

The test results with normal and abnormal data are shown in Fig. 5b and c, respectively. The two normal points were recognized as abnormal. Their PID values are $56.5, 185, 4$ and $62, 211, 4$. The two abnormal points were recognized as normal. All test data recognized as abnormal is abnormal with probability $P > 0.999$.

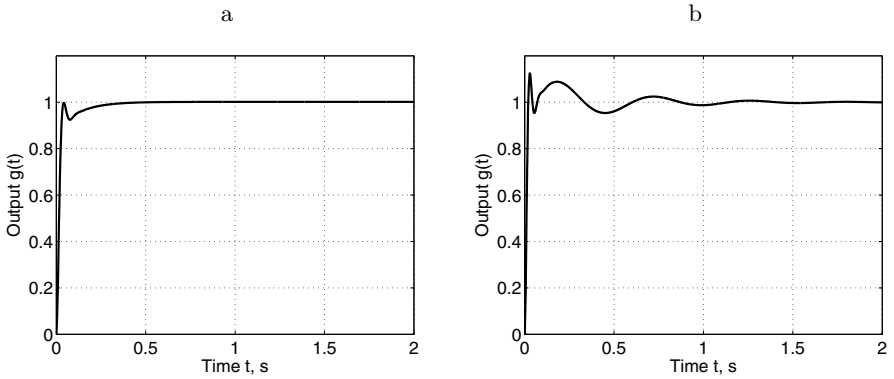


Fig. 4. Simulation results suggested by GA. a) The output corresponds to the best parameters used in the normal set, $P = 47.5$, $I = 3$, $D = 4.5$. b) The example of output corresponding to the parameters used in the abnormal set, $P = 43$, $I = 1382.5$, $D = 8.5$. The output signal has overshoots over 10 percent, settling time too long, and too high overall error.

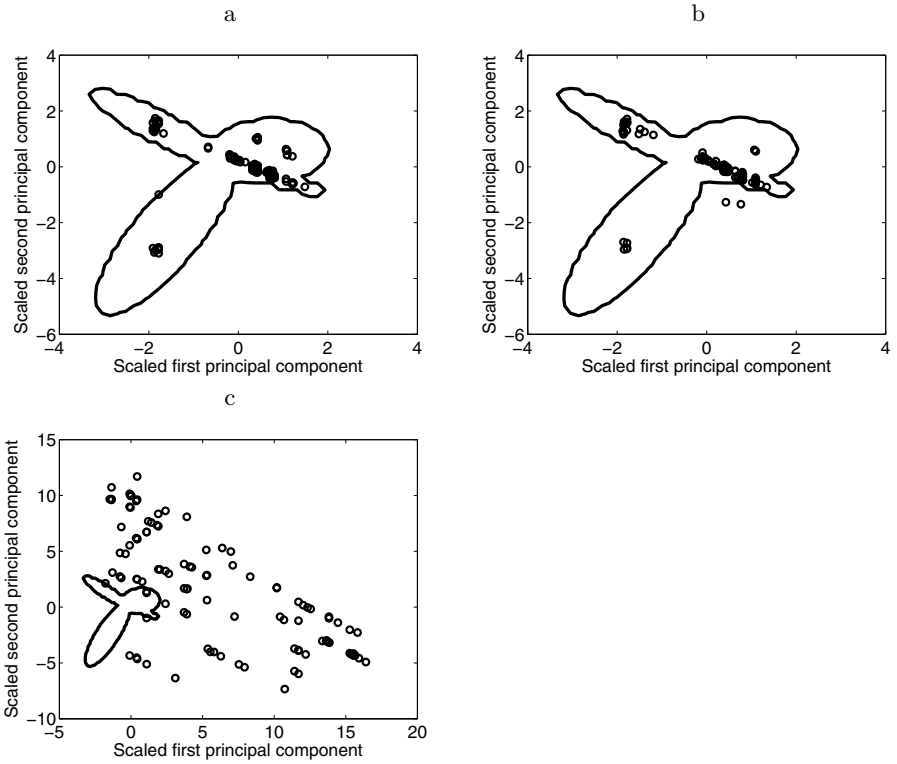


Fig. 5. The PID parameter analysis using a MEVS approach. The data points are shown by circles. a) Training. b) Test with normal data c) Test with abnormal data.

The confusion matrix is as follows: $\begin{pmatrix} 102 & 2 \\ 2 & 102 \end{pmatrix}$. We obtain the sensitivity $TPR = 0.98$, false positive rate $FPR = 0.02$ and accuracy $ACC = 0.98$, which means that the proposed method predicts well.

7 Conclusions

We propose an outlier analyzer for diesel engine configuration diagnostics using a modified MEVS approach. We tested the analyzer with PID data generated by a genetic algorithm modeling PID controller - engine system and simulated with Simulink and Matlab. The analyzer showed good results in detection of outliers for the test set of parameters.

We note that the Gaussian distributions used in VMM are not robust and sensitive to a small number of solitary points. Therefore, VMM frequently overestimates the number of Gaussian kernels. This problem can be resolved by replacing the Gaussian distributions by the Student-t distributions which are more robust. For a future work we will consider the robust variant of a mixture model, also GAs might be used with VMM to find an optimal model.

Acknowledgments. The authors greatly appreciate the support, advise and help given by Dr. David A. Clifton from University of Oxford, UK, and helpful suggestions given by Dr. Janne Koljonen and Dr. Petri Välisuo from University of Vaasa, Finland. The authors would also like to thank the anonymous reviewers for their valuable comments to improve the quality of the paper.

References

1. Agovic, A., Banerjee, A., Ganguly, A.R., Protopopescu, V.A.: Anomaly detection in transportation corridors using manifold embedding. In: Proceedings of the First International Workshop on Knowledge Discovery from Sensor Data, ACM KDD Conference, San Jose, CA (2007)
2. Alander, J.T.: Indexed Bibliography of Genetic Algorithms in Control, Report No. 94-1-CONTROL, University of Vaasa, Department of Information Technology and Production Economics, University of Vaasa (1995), `http://lipas.uwasa.fi/~TAU/reports/report94-1/gaCONTROLbib.pdf`
3. Alander, J.T.: Indexed Bibliography of Genetic Algorithms in Machine Learning, Report No. 94-1-ML, Department of Electrical Engineering and Automation, University of Vaasa (2007), `http://lipas.uwasa.fi/~TAU/reports/report94-1/gaMLbib.pdf`
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2007)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A Survey. ACM Computing Surveys 41(3), 1–72 (2009)
6. Clifton, D.A.: Condition monitoring of gas-turbine engines. Transfer Report, Department of Engineering Science, University of Oxford (2005)
7. Clifton, D.A., Hugueny, S., Tarassenko, L.: Novelty detection with multivariate extreme value statistics. Journal of Signal Processing Systems 65, 371–389 (2011)

8. Haugen, F., Fjelddalen, E., Dunia, R., Edgar, T.F.: Demonstrating PID control principles using an Air Heater and LabVIEW. CACHE News (Computer Aids for Chemical Engineering) (Winter 2007)
9. Hugueny, S., Clifton, D.A., Tarassenko, L.: Probabilistic Patient Monitoring with Multivariate, Multimodal Extreme Value Theory. In: Fred, A., Filipe, J., Gamboa, H. (eds.) BIOSTEC 2010. CCIS, vol. 127, pp. 199–211. Springer, Heidelberg (2011) (Invited article, from IEEE Biomedical Engineering Systems and Technologies)
10. Lewis, P.H., Yang, C.: Basic Control Systems Engineering. Prentice-Hall Inc. (1997)
11. Mantere, T., Alander, J.T.: Evolutionary Software Engineering, a Review. Applied Soft Computing 5(3), 315–331 (2005)
12. Mohamed, F.A., Koivo, H.N.: Diesel engine systems with genetic algorithm self tuning PID controller. Technical Report, Control Engineering Lab, Helsinki University of Technology (2005)
13. Olsson, J.: Automatic tuning of control parameters for single speed engines. Master's Thesis. Stockholm, The Royal Institute of Technology, November 22 (2004)
14. Pedersen, G.K.M.: Towards automatic controller design using multi-objective evolutionary algorithms. Ph.D. Thesis, Department of Control Engineering, Aalborg University (2005)
15. Rakopoulos, C.D., Giakoumis, E.G.: Availability analysis of a turbocharged diesel engine operating under transient load conditions. Energy 29, 1085–1104 (2004)
16. Roberts, S.J.: Extreme value statistics for novelty detection in biomedical data processing. In: IEE Proceedings - Science, Measurement and Technology, vol. 147, pp. 363–367 (2000)
17. Sundaram, I.S., Strachan, I.G.D., Clifton, D.A., Tarassenko, L., King, S.: Aircraft engine health monitoring using density modeling and extreme value statistics. In: Proc. 6th Intern. Conference on Condition Monitoring and Machinery Failure Prevention Technologies, Dublin, Ireland, pp. 919–930 (2009)
18. Variational Bayesian Expectation Maximization for Gaussian Mixture Models, `http://www.cs.ubc.ca/~murphyk/Software/VBEMGMM/index.html`
19. Törmänen, P.: Evaluating the benefit of fuzzy logic for PID-control by means of genetic algorithms - case: frequency controller. University of Vaasa. Master's thesis 31-32 (1997)
20. Åström, K., Hägglund T. H.: PID controllers: Theory, Design and Tuning. Instrument Society of America (1995)

Unit Operational Pattern Analysis and Forecasting Using EMD and SSA for Industrial Systems

Zhijing Yang, Chris Bingham, Wing-Kuen Ling, Yu Zhang, Michael Gallimore, and Jill Stewart

School of Engineering, University of Lincoln, Lincoln, LN6 7TS, UK
{zyang,cbingham,wling,yzhang,mgallimore,jstewart}@lincoln.ac.uk

Abstract. This paper studies operational pattern analysis and forecasting for industrial systems. To analyze the global change pattern, a novel methodology for extracting the underlying trends of signals is proposed, which is based on the sum of chosen intrinsic mode functions (IMFs) obtained via empirical mode decomposition (EMD). An adaptive strategy for the selection of the appropriate IMFs to form the trend, is proposed. Then, to forecast the change of the trend, Singular Spectrum Analysis (SSA) is applied. Results from experiment trials on an industrial turbine system show that the proposed methodology provides a convenient and effective mechanism for forecasting the trend of the operational pattern. In so doing, it therefore has application to support flexible maintenance scheduling, rather than the traditional use of calendar based maintenance.

Keywords: Operational pattern analysis, trend extraction, empirical mode decomposition, signal forecasting, singular-spectrum analysis (SSA).

1 Introduction

Preventive maintenance (PM) planning has received significant attention in both manufacturing industry and the manufacturing systems & operations research literature [1]. To improve the performance of PM, a key issue is to analyze the operational pattern of the units being considered [2].

Here then, the paper proposes a number of techniques to analyze operational patterns and operational forecasting for industrial systems. To analyze the global change pattern, a novel methodology for extracting the underlying trends of signals based on empirical mode decomposition (EMD), is proposed. EMD has attracted significant recent attention due to its ability to decompose any signal into basic components that are inherently useful for trend extraction [3]. Indeed, the underlying trend of some signals can be approximated using only the last IMF obtained through EMD, making it extremely powerful for analyzing nonlinear and nonstationary signals [4]. However, since the final IMF is always monotonic, extracting the underlying trend can be severely limited in circumstances where it is not monotonic [5]. Moreover, adaptively choosing the appropriate IMFs to describe the underlying trend in such circumstances remains an unsolved problem. In order to address these deficiencies, this paper presents an adaptive strategy for the selection of the appropriate IMFs to form the final trend based on the separation of the IMFs' Hilbert marginal spectrums.

J. Hollmén, F. Klawonn, and A. Tucker (Eds.): IDA 2012, LNCS 7619, pp. 416–423, 2012.
© Springer-Verlag Berlin Heidelberg 2012

After obtaining the trend, a forecast of the change of the trend is calculated that is shown to be useful for designing maintenance plans using Singular Spectrum Analysis (SSA) [9]. The SSA technique has been previously used in a variety of fields such as signal processing [6], mathematical statistics [7], and especially in time series forecasting [8]. Unlike the traditional parametric methods, such as linear Auto-Regressive moving average (ARMA) [10], the SSA is a powerful technique for nonparametric time series analysis and forecasting. Here, SSA is used to forecast the change of the trends of operating time of industrial generator units.

2 Problem Identification

The paper therefore considers the operational duty patterns of industrial generator units, as a mechanism for predictive maintenance scheduling. The problems studied here are the trend extraction and prediction of the operational duty. Specifically 3 gas turbine units, denoted as 1, 2, and 3, are used which have located on the same site. All units are used as generators and have intermittent operational duties. Four time series signals are considered: the total running hours per day of each unit and the total running hours per day of all units combined. The time period covers 8 years from 2004 to 2011. The four time series signals are shown in Figs. 1-2. It can be seen that the daily duty of each unit varies significantly.

3 Operational Analysis by Trend Extraction

The underlying trend of the operational duty of the units is determined using a new trend extraction method based on EMD and the Hilbert marginal spectrum.

In principle, EMD decomposes a signal into intrinsic mode functions (IMFs), from which the instantaneous frequencies can be analyzed using the Hilbert transform [3]. When the decomposition is complete, the signal, $x(t)$, can be expressed as

$$x(t) = \sum_{i=1}^{n} c_i(t) + r_n(t) \cdot \tag{1}$$

The final $r_n(t)$ is also considered as an IMF, denoted as $c_{n+1}(t)$, such that

$$x(t) = \sum_{i=1}^{n+1} c_i(t) \cdot \tag{2}$$

Having decomposed a signal into a finite number of IMFs, the resulting time-frequency distribution can be obtained using the Hilbert transform [3]. The Hilbert transform, $\tilde{H}[\cdot]$, is initially applied to each IMF, $c_i(t)$, to obtain an analytic representation of the signal,

$$z_i(t) = c_i(t) + j\tilde{H}[c_i(t)] = a_i(t)e^{j\theta_i(t)} \cdot$$

Clearly,

$$a_i(t) = \sqrt{(c_i(t))^2 + (\tilde{H}[c_i(t)])^2} \text{ , and } \theta_i(t) = \arctan(\frac{\tilde{H}[c_i(t)]}{c_i(t)}), \tag{3}$$

From (3), the instantaneous frequency of $c_i(t)$, $\omega_i(t)$, is defined as

$$\omega_i(t) = \frac{d\theta_i(t)}{dt}.$$

The time-frequency distribution of amplitude is the Hilbert spectrum [3], denoted by $H(\omega,t)$ where $H(\omega,t)=0$ except $H(\omega_i(t),t)=a_i(t)$. For practical purposes the sampling frequency is given by $\omega_m = m\Delta\omega$ for $m \in Z$, and discrete time $t_n = n\Delta t$ for $n \in Z$. Then, the Hilbert marginal spectrum (HMS) is given by

$$h(\omega_m) = \sum_n H(\omega_m, t_n),$$

and is calculated for each IMF (IMF_i):

$$h_i(\omega_m) = \sum_n H(\omega_i(t_n), t_n). \tag{4}$$

Adaptively choosing the IMFs corresponding to the underlying trend of a signal remains an unsolved problem. However, here we propose an adaptive mechanism based on the separation of the respective Hilbert marginal spectra. The absolute value of the correlation coefficient is used to measure the separation of two consecutive IMFs in the Hilbert marginal spectrum, and an adaptive strategy for extracting the underlying trend is proposed, as follows:

Algorithm 1 (Trend Extraction based on EMD)
Step 1: Capture a (noisy) signal $y(t)$ where $t \in R$, and define an optional error tolerance $\varepsilon > 0$.

Step 2: Decompose $y(t)$ into a sum of IMFs by EMD, that is, $y(t) = \sum_{i=1}^{N} c_i(t)$, where N is the total number of IMFs of $y(t)$.

Step 3: Compute the Hilbert marginal spectrum of $c_i(t)$ according to equations (3)-(4), denoted as $h_i(m\Delta\omega)$, $i = 1, \cdots, N$;

Step 4: Compute the absolute value of the correlation coefficient of the Hilbert marginal spectrums of two consecutive IMFs, that is,

$$\rho_i = \left| \frac{\sum_m (h_i(m\Delta\omega) - \bar{h}_i)(h_{i+1}(m\Delta\omega) - \bar{h}_{i+1})}{\sqrt{\sum_m (h_i(m\Delta\omega) - \bar{h}_i)^2 \sum_m (h_{i+1}(m\Delta\omega) - \bar{h}_{i+1})^2}} \right|, \qquad i = 1, \cdots, N-1. \tag{5}$$

Step 5: Find the largest number, n_0, such that the following inequality is satisfied:

$$\rho_i \geq \varepsilon, \qquad \forall\, i \geq n_0.$$

Step 6: Define the underlying trend of the signal as $\Gamma(t) = \sum_{i=n_0}^{N} c_i(t)$.

To analyze the pattern of operation of the units, Algorithm 1 is initially applied to the running hours of 3 units to extract the global characteristics. The results are shown in Figs. 1-2, respectively.

From Fig. 1, it can clearly be seen that the proposed method can track the global change of the total running hours. It can be seen in this instance that the total operational time has a global minimum in 2008, and there exists an annual pattern hidden in the data that shows the total running hours peak during winter months.

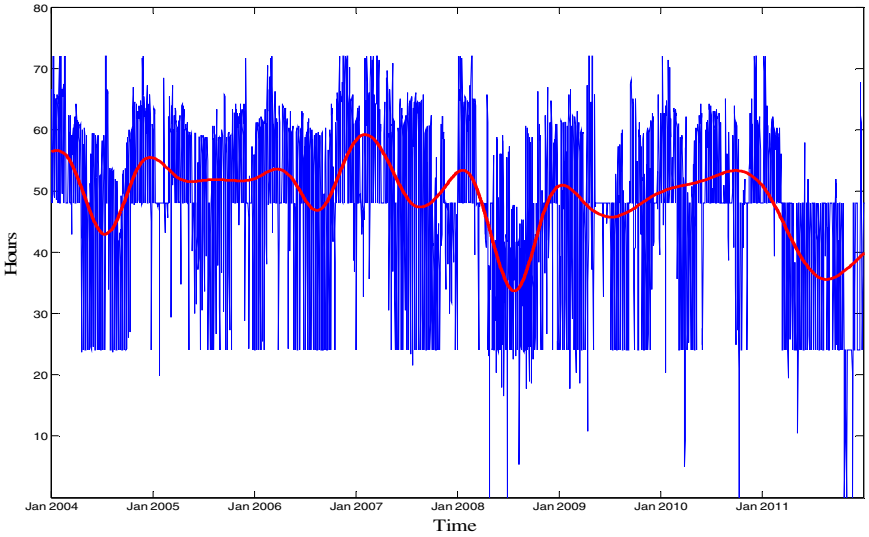


Fig. 1. The original signal and the trend of the total operating hours of 3 units

From Fig. 2, it can be seen that unit 1 is used less in 2009, and that more recently, the duty of the unit is significantly decreasing. For unit 2, it appears that the operational duty is less than the others, but all characteristics show an annual periodic trend. Unit 3 shows increased utilization as time progresses.

Thus far, only past trends have been considered. For predictive maintenance scheduling, the most significant benefits come from being able to forecast the units operational characteristics based on past 'behavior'. For this, the authors now propose the use of Singular Spectrum Analysis.

4 Trend Forecasting by Singular Spectrum Analysis

Singular Spectrum Analysis (SSA) has previously been highlighted as a very effective technique for time series analysis and forecasting [9]. Consider $x(n)$, $n=1,\cdots,N$ to be a time series. Given the window length $L\,(1<L<N)$, construct the L-dimensional vectors $X_n=(x(n),\cdots,x(n+L-1))^T$, $n=1,\cdots,K=N-L+1$. The resulting K vectors X_n are used to form the $L\times K$ *trajectory matrix:*

$$X=[X_1,\cdots,X_K]. \tag{6}$$

Now, let $S=XX^T$, with $\lambda_1\geq\cdots\geq\lambda_L\geq 0$ being the eigenvalues of the matrix S, $d=\max\{j:\lambda_j>0\}$, $U_1,\cdots,U_d$ being the corresponding eigenvectors, and

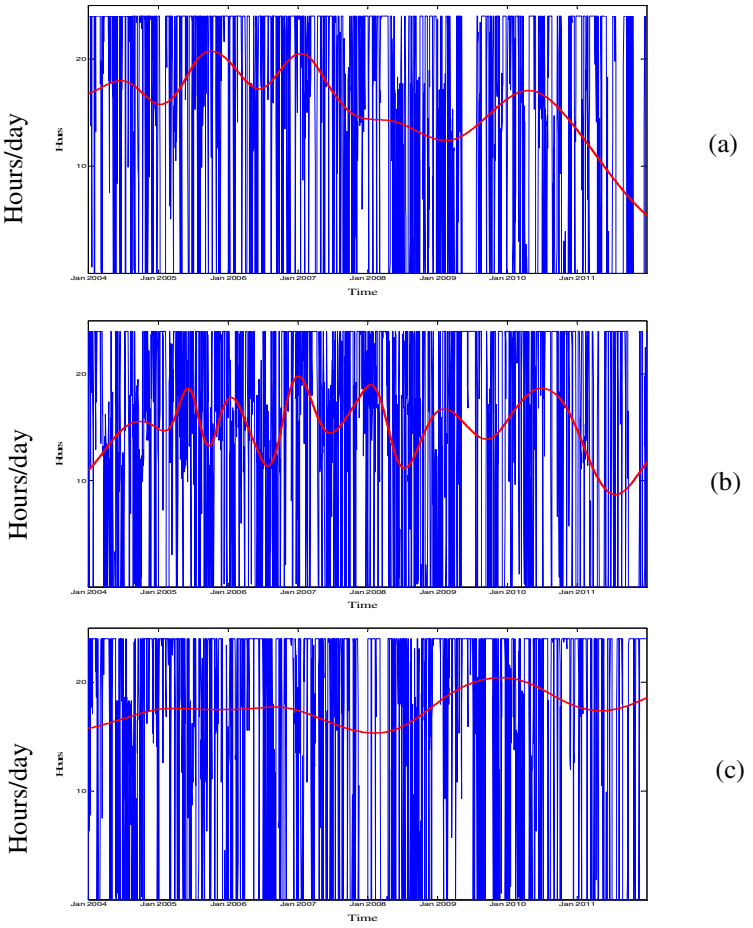


Fig. 2. The original signal and the trend of the operating hours (a) of unit 1; (b) of unit 2; (c) of unit 3

$V_j = X^T U_j / \sqrt{\lambda_j}$, $j = 1, \cdots, d$, being factor vectors. Denote $\tilde{X}_j = \sqrt{\lambda_j} U_j V_j^T$. Finally the SVD of the trajectory matrix X is represented by

$$X = \tilde{X}_1 + \cdots + \tilde{X}_d . \tag{7}$$

Partition the indices set $\{1, \cdots, d\}$ into m disjoint subsets $I_1, \cdots, I_m$, and let $I = \{i_1, \cdots, i_I\}$. The resultant matrix $\tilde{X}_I$ corresponding to the group I is defined as $\tilde{X}_I = \tilde{X}_{i_1} + \cdots + \tilde{X}_{i_I}$, leading to the new expansion of (7) as

$$X = \tilde{X}_{I_1} + \cdots + \tilde{X}_{I_m} . \tag{8}$$

The final step, termed *diagonal averaging*, transforms each resultant matrix in (8) into a new one-dimensional signal (reconstructed) of length N by a hankelization-like procedure (details can be found in [9]).

The resulting SSA procedure can be used to forecast, since it can identify the underlying structure of the time series. Assuming this structure is preserved for the time period to be predicted, a model of the characteristics can be constructed. The model is expressed as a *Linear Recurrent Formulae* (LRF) applied to the last $L-1$ terms of the original signal. That is

$$x(N+1)=\sum_{k=0}^{L-2} a_k x(n-k), \tag{9}$$

where the coefficients of the LRF, $R=(a_{L-2},\cdots,a_0)^T$, are given by

$$R=\frac{1}{1-\upsilon^2}\sum_{i=1}^{r}\pi_i U_i^{\nabla}, \tag{10}$$

and $\upsilon^2=\sum_{i=1}^{r}\pi_i^{\,2},(r<L)$, π_i is the last component of vector U_i, and $U_i^{\nabla}\in R^{L-1}$ is the vector consisting of the first $L-1$ components of U_i. The same principle is applied to $x(N+2)$, and so on.

Remarks: The window length L is set as $L=\frac{[N]}{2}$, where the notation $[\cdot]$ indicated rounding towards nearest integer.

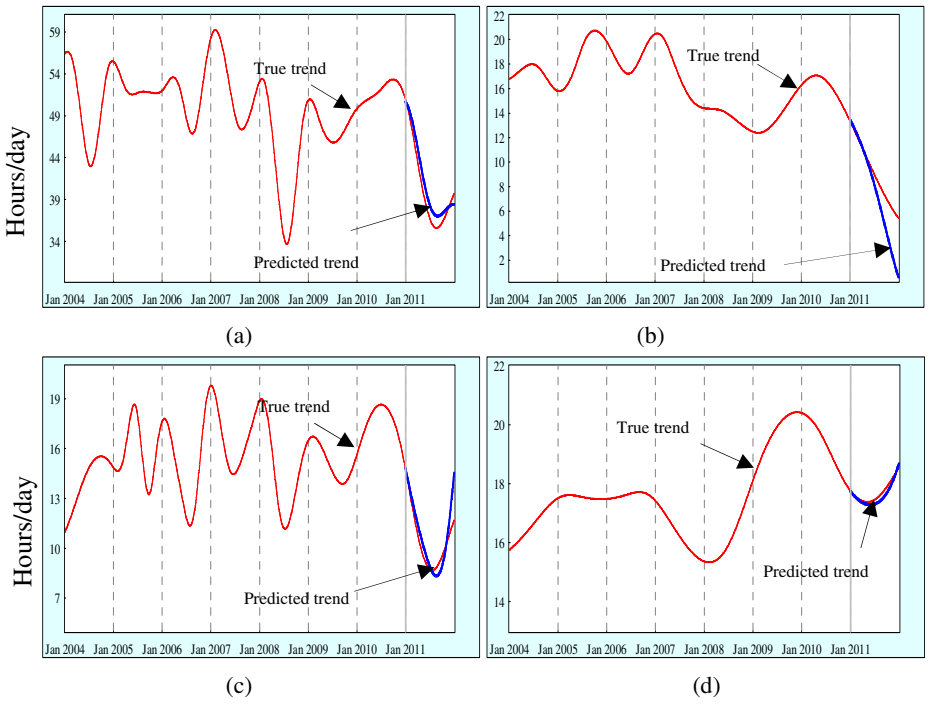


Fig. 3. The true trend and the predicted trend for 2011 of the operational hours (a) of 3 units combined; (b) of unit 1; (c) of unit 2; (d) of unit 3

Here then, the SSA forecasting method is applied to predict the future unit operation from the trends obtained in Section 3 (i.e. in Figs. 1-2) with a view to providing input to maintenance planning and operational guidance for the customers.

To check the reliability of prediction results, initially the first 7 years' data of the trend (2004-2010) is used to train the model, i.e. apply SSA to the first 7 years' data to obtain the LRF model; and then the model is used to predict the trend in 2011. After applying the methodology, the results for the predicted total operational hours of the 3 units for 2011, are shown in Fig. 3(a)-(d), respectively.

By comparing the true trends and the predicted trends in 2011 as shown in Fig. 3(a)-(d), it can be seen that they provide very reliable predictions of duty at least 6 months in advance, giving the customer key information as to the future condition of the units. Based on these considerations, SSA is now applied to the currently available 8 years of data (2004-2011) to train the model, and is then used to predict the operational trend in 2012. The predicted trends again are shown in Fig. 4(a)-(d), respectively.

From Fig. 4(a)-(d), the change of operational trends of each unit can therefore be forecast. Specifically, it can be seen, for instance, that unit 3 is expected to be utilized more during 2012 (around 22 hours per day), and may required advance maintenance. Moreover, the total running hours of the 3 units is expected to reduce significantly in the summer and increase dramatically in the winter of 2012. Again this gives useful maintenance information to the provider.

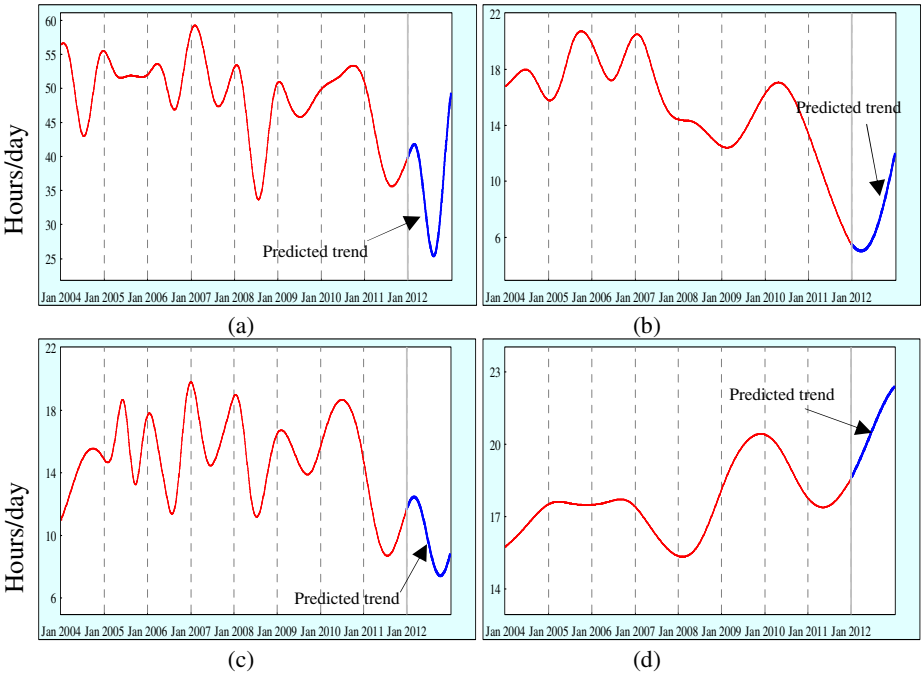


Fig. 4. The true trend of 8 years for training and the predicted trend for 2012 of the operational hours (a) of 3 units combined; (b) of unit 1; (c) of unit 2; (d) of unit 3.

5 Conclusion

Operational pattern analysis and forecasting for industrial systems has been carefully studied in this paper. A case study from a specific industrial system has been given. To analyze the global pattern, a novel methodology for extracting the underlying trends of signals based on empirical mode decomposition (EMD), is proposed. An adaptive strategy for the selection of the appropriate IMFs to form the final trend, is then developed. Then, the well known Singular Spectrum Analysis (SSA) technique is used for trend forecasting. It is shown, through application to the operational characteristics of a fleet of industrial gas turbines, that the proposed methodologies provide a useful tool for both the customer and unit provider to facilitate timely adaptive maintenance scheduling.

Acknowledgments. The authors would like to acknowledge Siemens Industrial Turbomachinery, Lincoln, UK, for their support of this research.

References

1. Cho, D., Parlar, M.: A survey of maintenance model for multi-unit systems. European Journal of Operational Research 51, 1–23 (1991)
2. Peng, X., Gui, W., Li, Y., Hu, Z., Wang, L.: Operational Pattern Optimization for Copper Flash Smelting Process Based on Pattern Decomposition of Fuzzy Neural Networks. In: IEEE International Conference on Control and Automation, ICCA, pp. 2328–2333 (2007)
3. Huang, N., Shen, Z., Long, S., Wu, M., Shih, H., Zheng, Q., Yen, N., Tung, C., Liu, H.: The empirical mode decomposition and the Hilbert spectrum for nonlinear and nonstationary time series analysis. Proceedings of the Royal Society London A: Mathematical, Physical and Engineering Sciences 454, 903–995 (1998)
4. Huang, N., Shen, Z., Long, S.: A new view of nonlinear water waves: the Hilbert spectrum. Annual Reviews of Fluid Mechanics 31, 417–457 (1999)
5. Wu, Z., Huang, N., Long, S., Peng, C.: On the trend, detrending, and variability of nonlinear and nonstationary time series. Proceedings of National Academy of Sciences 104, 14889–14894 (2007)
6. Kumaresan, R., Tufts, D.: Data-adaptive principal component signal processing. In: IEEE Proceedings Conference on Decision and Control, Albuquerque, p. 949 (1980)
7. Moskvina, V., Zhigljavsky, A.: An algorithm based on singular spectrum analysis for change-point detection. Commun. Statistics-Simul. Comput. 32(2), 319–352 (2003)
8. Baratta, D., Cicioni, G., Masulli, F., Studer, L.: Application of an ensemble technique based on singular spectrum analysis to daily rainfall forecasting. Neural Networks 16, 375–387 (2003)
9. Golyandina, N., Nekrutkin, V., Zhigljavsky, A.: Analysis of Time Series Structure: SSA and related techniques. Chapman and Hall/CRC, New York (2001)
10. Grimaldi, S.: Linear Parametric Models Applied to Daily Hydrological Series. Journal of Hydrologic Engineering 9(5), 383–391 (2004)

Author Index