Proceedings of the Workshop on Feature Selection for Data Mining:

Interfacing Machine Learning and Statistics

in conjunction with the

2005 SIAM International Conference on Data Mining

April 23, 2005

Newport Beach, CA

Chairs Huan Liu (Arizona State University)
Robert Stine (University of Pennsylvania)
Leonardo Auslender (SAS Institute)

Sponsored By sas

Workshop on Feature Selection for Data Mining:

Interfacing Machine Learning and Statistics

http://enpub.eas.asu.edu/workshop/

2005 SIAM Workshop

April 23, 2005

Message from the Workshop Chairs

Knowledge discovery and data mining (KDD) is a multidisciplinary effort to extract nuggets of knowledge from data. The proliferation of large data sets within many domains poses unprecedented challenges to KDD. Not only are data sets getting larger, but new types of data have also evolved, such as clickstream data on the Web and microarrays in genomics. Research in computer science, engineering, and statistics confront similar issues in feature selection, and we see a pressing need for the interdisciplinary exchange and discussion of ideas. We anticipate that our collaborations will shed new light on research directions and approaches and lead to breakthroughs.

Researchers in data mining and knowledge discovery recognize the value of knowledge imbedded in massive data sets. This knowledge can often be represented as 'patterns' that allow summarization, classification, prediction, and planning. Typically, these useful patterns are derived from data by empirical modeling. Few domains have developed theories that are adequate to organize the large quantities of data now held in repositories that continue to grow larger, with both more dimensions and instances. To arrive at patterns, many techniques simplify the task by reducing number of dimensions by selecting variables and features. This approach has proven to be efficient and effective in dealing with high-dimensional data in various data-mining applications. These applications include pattern recognition, image processing, machine learning, and data mining (including Web, text, and microarrays). The objectives of feature selection include: building simpler and more comprehensible models, improving data mining performance, and helping to prepare, clean, and understand data.

This workshop aims to encourage cross-disciplinary, collaborative research in variable and feature selection. Both computer scientists and statisticians have made important contributions to variable and feature selection and both groups continue to pursue a variety of innovative research programs. Though sharing a common interest, the two groups have not been well-connected. Thus, it is beneficial to computer scientists and statisticians that a bridge of communication be established and maintained. This connection among researchers would enhance learning from one another and help all address the challenges arising from massive data.

These proceedings contain a wide range of research work in feature selection: feature selection methodology research, text categorization and classification, analysis of microarray measures of genetic structure, predictive modeling, multivariate time series analysis, performance improvement, and subspace detection. It has been an enjoyable process for us to work together in achieving the aims of this workshop.

We would like to convey our gratefulness to our PC members and authors who have contributed tremendously to make this workshop a success.

Huan Liu, Robert Stine, and Leonardo Auslender

Workshop Schedule

7:00am – 8:15am Continental Breakfast

8:30am – 10:00am
A Novel Feature Selection Score for Text Categorization (25 minutes)
Susana Eyheramendy, David Madigan
Text Classification by Augmenting the Bag-of-Words Representation with Redundancy-Compensated Bigrams (25 minutes)
Constantinos Boulis, Mari Ostendorf
Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering (25 minutes)
Bin Tang, Michael Shepherd, Evangelos Milios, Malcolm I. Heywood
Near-Optimal Feature Selection (15 minutes)
Jaekyung Yang, Sigurdur Olafsson

10:00am – 10:30am Coffee Break (poster session)

10:30am – 12:00
Boosted Lasso (25 minutes)
Peng Zhao, Bin Yu
Feature Selection with a General Hybrid Algorithm (25 minutes)
Jerffeson Souza, Nathalie Japkowicz, Stan Matwin
Minimum Redundancy and Maximum Relevance Feature Selection and Recent Advances in Cancer Classification (25 minutes)
Hanchuan Peng, Chris Ding
Gene Expression Analysis of HIV-1 Linked p24-specific CD4+ T-Cell Responses for Identifying Genetic Markers (15 minutes)
Sanjeev Raman, Carlotta Domeniconi

12:00 – 1:45pm Lunch

1:45pm – 2:45pm Keynote Talk
Model Building and Feature Selection with Genomic Data
Trevor Hastie

2:45pm – 3:15pm Coffee Break (poster session)

3:15pm – 4:30pm
Feature Filtering with Ensembles Using Artificial Contrasts (15 minutes)
Eugen Tuv, Kari Torkkola
Speeding up Multi-class SVM Evaluation by PCA and Feature Selection (15 minutes)
Hansheng Lei, Venu Govindaraju
Detecting Outlying Subspaces for High-Dimensional Data: A Heuristic Approach (15 minutes)
Ji Zhang
An Optimal Binning Transformation for Use in Predictive Modeling (15 minutes)
Talbot Michael Katz
A Supervised Feature Subset Selection Technique for Multivariate Time Series (15 minutes)
Kiyoung Yang, Hyunjin Yoon, Cyrus Shahabi

4:30pm Workshop Adjourns

Poster Papers
A Hybrid Cluster Tree Algorithm for Variable Selection
Zhiqian Fu, Zhiwei Fu, Isa Sarac
Parallelizing Feature Selection
Jerffeson Souza, Nathalie Japkowicz, Stan Matwin
Optimal Division for Feature Selection and Classification
Mineichi Kudo, Hiroshi Tenmoto

Table of Contents

A novel feature selection score for text categorization

Susana Eyheramendy
Department of Statistics
1 South Parks Road
Oxford University
Oxford, OX1 3TG

David Madigan
Department of Statistics
501 Hill Center
Rutgers University
Piscataway, NJ 08855

Abstract

This paper proposes a new feature selection score for text classification. The value that this score assigns to each feature has an appealing Bayesian interpretation, being the posterior probability of inclusion of the feature in a model. We evaluate the performance of the score, together with five other feature selection scores that have been prominent in the text categorization literature, using four classification algorithms and two benchmark text datasets. The proposed score performs reasonably well. We find that it is among the best two scores,together with χ^2.
Keywords: feature selection, text classification, Bayesian analysis.

1 Introduction

Since many text classification applications involve large numbers of candidate features, feature selection algorithms continue to play an important role. The text classification literature tends to focus on feature selection algorithms that compute a score independently for each candidate feature. This is the so-called *filtering* approach. The scores typically contrast the counts of occurrences of words or other linguistic artifacts in training documents that belong to the target class with the same counts for documents that do not belong to the target class. Given a predefined number of words to be selected, say d, one chooses the d words with the highest score. Several score functions exist (Section 2 provides definitions). Yang and Pedersen (1997) show that Information Gain and χ^2 statistics performed best among five different scores. Forman (2003) provides evidence that these two scores have correlated failures. Hence when choosing optimal pairs of scores these two scores work poorly together. He introduced a new score, the Bi-Normal Separation, that yields the best performance on the greatest number of tasks among twelve feature selection scores. Mladenic and Grobelnik (1999) compare eleven scores under a Naive Bayes classifier and find that the Odds Ratio score performed best in the highest number of tasks.

In regression and classification problems in Statistics, popular feature selection strategies depend on the same algorithm that fits the models. This is the so-called *wrapper* approach. For example, *Best subset regression* finds for each k the best subset of size k based on residual sum of squares. *Leaps and bounds* is an efficient algorithm that finds the best set of features when the number of predictors is no larger than about 40. Miller (2002) provides an extensive discussion.

Barbieri and Berger (2004) in a Bayesian context and under certain assumptions show that for selection among normal linear models, the best model contains those features which have overall posterior probability greater than or equal to 1/2. Motivated by this study we introduce a new feature selection score (PIP) that evaluates the posterior probability of inclusion of a given feature over all possible models, where the models correspond to a set of features. Unlike typical scores used for feature selection via filtering, the PIP score does depend on a specific model. In this sense, the new score straddles the filtering and wrapper approaches.

We present experiments that compare the new feature selection score with five other feature selection scores that have been prominent in the studies mentioned above. The feature selection scores that we consider are evaluated on two widely-used benchmark text classification datasets, Reuters-21578 and 20-Newsgroups, and implemented on four classification algorithms. Following previous studies, we measure the performance of the classification algorithms using the F_1 measure.

We have organized this paper as follows. Section 2 describes the various feature selection scores we consider,

	c_k	$c_{\overline{k}}$	
w	n_{kw}	$n_{\overline{k}w}$	n_w
$\overline{w}$	$n_{k\overline{w}}$	$n_{\overline{k}\overline{w}}$	$n_{\overline{w}}$
	n_k	$n_{\overline{k}}$	n

Table 1: Two-way contingency table of word w and category c_k

both the new score and the various existing competitors. In Section 3 we mention the classification algorithms that we use to compare the feature selection scores. The experimental settings and experimental results are in Section 4. Section 5 has the conclusions.

2 Feature Selection Scores

Feature selection, or word selection in the experiments of this study, uses a score to select the best d words from all words that appear in the training set. Before we list the feature selection scores that we study, we introduce some notation. Table 1 show the basic statistics for a single word and a single category (or class).

n_{kw} : n° of documents in class c_k with word w.
$n_{k\overline{w}}$: n° of documents in class c_k without word w.
$n_{\overline{k}w}$: n° of documents not in class c_k with word w.
$n_{\overline{k}\overline{w}}$: n° of documents not in class c_k without word w.
n_k : total n° of documents in class c_k.
$n_{\overline{k}}$: total n° of documents that are not in class c_k.
n_w : total n° of documents with word w.
$n_{\overline{w}}$: total n° of documents without word w.
n : total n° of documents.

2.1 Posterior Inclusion Probability (PIP) under a Bernoulli distribution

We introduce a new feature selection score which is motivated by the median probability model. We first consider the binary naive Bayes model. Section 2.2 considers a naive Bayes model with Poisson distributions for word frequency. This score for feature or word w and class c_k is defined as

$$PIP(w, c_k) = \frac{l_{0wk}}{l_{0wk} + l_{wk}} \tag{1}$$

where

$$\begin{aligned} l_{0wk} &= \frac{B(n_{kw} + \alpha_{kw}, n_{k\overline{w}}\beta_{kw})}{B(\alpha_{kw}, \beta_{kw})} \\ &\quad \times \frac{B(n_{\overline{k}w} + \alpha_{\overline{k}w}, n_{\overline{k}\overline{w}} + \beta_{\overline{k}w})}{B(\alpha_{\overline{k}w}, \beta_{\overline{k}w})} \end{aligned}$$

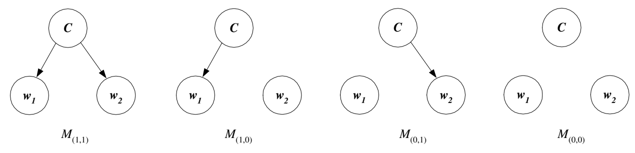


Figure 1: Graphical model representation of the four models with two words, w_1 and w_2.

$$l_{wk} = \frac{B(n_w + \alpha_w, n_{\overline{w}} + \beta_w)}{B(\alpha_w, \beta_w)}$$

$B(a, b)$ is the *Beta* function which is defined as $B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$, and α_{kw}, $\alpha_{k\overline{w}}$, α_w, β_{kw}, $\beta_{k\overline{w}}$, β_w are constants set by the practitioner. In our experiments we set them to be $\alpha_w = 0.2$, $\beta_w = 2/25$ for all words w, $\alpha_{kw} = 0.1$, $\alpha_{k\overline{w}} = 0.1$, $\beta_{kw} = 1/25$ and $\beta_{k\overline{w}} = 1/25$ for all categories k and words w. These settings correspond to rather diffuse priors.

We explicate this score on the context of a two-candidate-word model. In general, with d candidate words, there are 2^d models corresponding to allpossible subsets of the words. For two words, Figure 1 we show a graphical representation of the four possible models. The corresponding likelihoods for each model are given by

$$\begin{aligned} M_{(1,1)} &: \textstyle\prod_i Pr(w_{i1}, w_{i2}, c_i|\theta_{1c}, \theta_{2c}) = \prod_i \mathcal{B}(w_{i1}, \theta_{k1}) \\ &\quad \times\mathcal{B}(w_{i1}, \theta_{\overline{k}1})\mathcal{B}(w_{i2}, \theta_{k2})\mathcal{B}(w_{i2}, \theta_{\overline{k}2})Pr(c_i|\theta_k) \end{aligned}$$

$$\begin{aligned} M_{(1,0)} &: \textstyle\prod_i Pr(w_{i1}, w_{i2}, c_i|\theta_{1c}, \theta_2) = \prod_i \mathcal{B}(w_{i1}, \theta_{k1}) \\ &\quad \times\mathcal{B}(w_{i1}, \theta_{\overline{k}1})\mathcal{B}(w_{i2}, \theta_2)\mathcal{B}(w_{i2}, \theta_2)Pr(c_i|\theta_k) \end{aligned}$$

$$\begin{aligned} M_{(0,1)} &: \textstyle\prod_i Pr(w_{i1}, w_{i2}, c_i|\theta_1, \theta_{2c}) = \prod_i \mathcal{B}(w_{i1}, \theta_1) \\ &\quad \times\mathcal{B}(w_{i1}, \theta_1)\mathcal{B}(w_{i2}, \theta_{k2})\mathcal{B}(w_{i2}, \theta_{\overline{k}2})Pr(c_i|\theta_k) \end{aligned}$$

$$\begin{aligned} M_{(0,0)} &: \textstyle\prod_i Pr(w_{i1}, w_{i2}, c_i|\theta_1, \theta_2) = \prod_i \mathcal{B}(w_{i1}, \theta_1) \\ &\quad \times\mathcal{B}(w_{i1}, \theta_1)\mathcal{B}(w_{i2}, \theta_2)\mathcal{B}(w_{i2}, \theta_2)Pr(c_i|\theta_k) \end{aligned}$$

where w_{ij} takes the value 1 if document i contains word j and 0 otherwise, c_i is 1 if document i is in category k otherwise is 0, $Pr(c_i|\theta_k) = \mathcal{B}(c_i, \theta_k)$ and $\mathcal{B}(w, \theta) = \theta^w(1-\theta)^{1-w}$ denotes a Bernoulli probability distribution.

Therefore, in model $M_{(1,1)}$ the presence or absence of both words in a given docuement depends on the document class. θ_{k1} corresponds to the proportion of documents in category c_k with word w_1 and $\theta_{\overline{k}1}$ to the proportion of documents not in category c_k with word w_1. In model $M_{(1,0)}$ only word w_1 depends on the category of the document and θ_2 correspond to the proportion of documents with word w_2 regardless of the category associated with them. θ_k is the proportion of documents in category c_k and $Pr(c_i|\theta_k)$ is the probability that document d_i is in category c_k.

We assume the following prior probability distributions for the parameters, $\theta_{kw} \sim Beta(\alpha_{kw}, \beta_{kw})$, $\theta_{\overline{k}w} \sim Beta(\alpha_{\overline{k}w}, \beta_{\overline{k}w})$, $\theta_w \sim Beta(\alpha_w, \beta_w)$ and $\theta_k \sim Beta(\alpha_k, \beta_k)$, where $Beta(\alpha, \beta)$ denotes a Beta distribution, i.e. $Pr(\theta|\alpha,\beta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$, $k \in \{1, ..., m\}$ and $w \in \{1, ..., d\}$.

Then the marginal likelihoods for each of the four models above are:

$$Pr(data|M_{(1,1)}) = l_0 \times l_{01k} \times l_{02k}$$

$$Pr(data|M_{(1,0)}) = l_0 \times l_{01k} \times l_{2k}$$

$$Pr(data|M_{(0,1)}) = l_0 \times l_{1k} \times l_{02k}$$

$$Pr(data|M_{(0,0)}) = l_0 \times l_{1k} \times l_{2k}$$

where l_{0wk} and l_{wk} are defined above for $w \in \{1, 2, ..., d\}$ and $l_0 = \int_0^1 \prod_i Pr(c_i|\theta_k) Pr(\theta_k|\alpha_k, \beta_k) d\theta_k$ is the marginal probability for the category of the documents.

The overall posterior probability that a feature is included in a model, its posterior inclusion probability (PIP), is defined as

$$PIP(w, c_k) = \sum_{l:l_j=1} Pr(M_l|data) \qquad (2)$$

where l is a vector of length the number of features and the jth component takes the value 1 if the jth feature is included in model M_l, otherwise it is 0. It is straightforward to show that $PIP(w, c_k)$ in equation (4) is equivalent to $PIP(w, c_k)$ in equation (5), if we assume that the prior probability density for the models is uniform, e.g. $Pr(M_l) \propto 1$.

In the example above, the posterior inclusion probability for word w_1 is given by,

$$\begin{aligned} Pr(w_1|c_k) &= Pr(M_{(1,1)}|data) + Pr(M_{(1,0)}|data) \\ &= \frac{l_{01k}}{l_{01k} + l_{1k}} \end{aligned}$$

To get a single "bag of words" for all categories we compute the weighted average of $PIP(w, c_k)$ over all categories.

$$PIP(w) = \sum_k Pr(c_k) PIP(w, c_k)$$

We note that Dash and Cooper (2002) present similar manipulations of the naive Bayes model but for model averaging purposes rather than finding the median probability model.

2.2 Posterior Inclusion Probability (PIPp) under Poisson distributions

A gernalization of the binary naive Bayes model assumes class-conditional Poisson distributions for the word frequencies in a document. As before, assume that the probability distribution for a word in a document might or might not depend on the category of the document. More precisely, if the distribution for word w depends on the category c_k of the document we have,

$$\begin{aligned} Pr(w|c=1) &= \frac{e^{-\lambda_{kw}} \lambda_{kw}^w}{w!} \\ Pr(w|c=0) &= \frac{e^{-\lambda_{\overline{k}w}} \lambda_{\overline{k}w}^w}{w!} \end{aligned}$$

where w denotes a specific word and the number of times that word appears in the document and λ_{kw} ($\lambda_{\overline{k}w}$) represents the expected number of times that word w appears in documents in category c_k ($c_{\overline{k}}$). If the distribution for word w does not depend on the category of the document then we have,

$$Pr(w) = \frac{e^{-\lambda_w} \lambda_w^w}{w!}$$

where λ_w represents the expected number of times w appears in a document regardless of the category of the document.

Assume the following conjugate prior probability densities for the parameters,

$$\begin{aligned} \lambda_{kw} &\sim Gamma(\alpha_{kw}, \beta_{kw}) \\ \lambda_{\overline{k}w} &\sim Gamma(\alpha_{\overline{k}w}, \beta_{\overline{k}w}) \\ \lambda_w &\sim Gamma(\alpha_w, \beta_w) \end{aligned}$$

where $\alpha_{kw}, \beta_{kw}, \alpha_{\overline{k}w}, \beta_{\overline{k}w}, \alpha_w, \beta_w$ are hyperparameters to be set by the practitioner.

Now, as before, the posterior inclusion probability for poisson distributions (PIPp) is given by

$$PIPp(w, c_k) = \frac{l_{0wk}}{l_{0wk} + l_{wk}} \qquad \text{where,}$$

$$\begin{aligned} l_{0wk} &= \frac{\Gamma(N_{kw} + \alpha_{kw})}{\Gamma(\alpha_{kw})\beta_{kw}^{\alpha_{kw}}} \frac{\Gamma(N_{\overline{k}w} + \alpha_{\overline{k}w})}{\Gamma(\alpha_{\overline{k}w})\beta_{\overline{k}w}^{\alpha_{\overline{k}w}}} \\ &\quad \times (\frac{\beta_{kw}}{n_k \beta_{kw} + 1})^{n_{kw} + \alpha_{kw}} (\frac{\beta_{\overline{k}w}}{n_{\overline{k}} \beta_{\overline{k}w} + 1})^{n_{\overline{k}w} + \alpha_{\overline{k}w}} \\ l_{wk} &= \frac{\Gamma(N_w + \alpha_w)}{\Gamma(\alpha_w)} (\frac{\beta_w}{\beta_w n + 1})^{n_w + \alpha_w} \frac{1}{\beta_w^{\alpha_w}} \end{aligned}$$

This time, $N_{kw}, N_{\overline{k}w}, N_w$ denote:
N_{kw}: n° of times word w appears in documents in class c_k.
$N_{\overline{k}w}$: n° of times word w appears in documents not in class c_k.
N_w: total n° of times that word w appears in all documents.

As before, to get a single "bag of words" for all categories we compute the weighted average of $PIPp(w, c_k)$ over all categories.

$$PIPp(w) = \sum_k Pr(c_k) PIPp(w, c_k)$$

2.3 Information Gain (IG)

Information gain is a popular score for feature selection in the field of machine learning. In particular it is used in the C4.5 decision tree inductive algorithm. Yang and Pedersen (1997) compare five different feature selection scores on 2 datasets and show that Information Gain is among the two most effective ones. The information gain of word w is defined to be:

$$\begin{aligned} IG(w) &= -\sum_{k=1}^{m} Pr(c_k) \log Pr(c_k) \\ &\quad + Pr(w) \sum_{k=1}^{m} Pr(c_k|w) \log Pr(c_k|w) \\ &\quad + Pr(\overline{w}) \sum_{k=1}^{m} Pr(c_k|\overline{w}) \log Pr(c_k|\overline{w}) \end{aligned}$$

where $\{c_k\}_{k=1}^{m}$ denote the set of categories and $\overline{w}$ the abscence of word w. It measures the decrease in entropy when the feature is present versus when the feature is absent.

The probabilities in $IG(w)$ are estimated using the corresponding sample frequencies.

2.4 Bi-Normal Separation (BNS)

Forman (2003) defines Bi-Normal Separation as:

$$BNS(w, c_k) = |\Phi^{-1}(\frac{n_{kw}}{n_k}) - \Phi^{-1}(\frac{n_{\overline{k}w}}{n_{\overline{k}}})|$$

where Φ is the standard normal distribution and Φ^{-1} its corresponding inverse. $\Phi^{-1}(0)$ is set to be equal to 0.0005 to avoid numerical problems following Forman (2003). By averaging over all categories, we get a score that selects a single set of words for all categories.

$$BNS(w) = \sum_{k=1}^{m} Pr(c_k) |\Phi^{-1}(\frac{n_{kw}}{n_k}) - \Phi^{-1}(\frac{n_{\overline{k}w}}{n_{\overline{k}}})|$$

To get an idea for what this score is measuring, assume that the probability that a word w is contained in a document is given by $\Phi(\delta_k)$ if the document belongs to class c_k and otherwise is given by $\Phi(\delta_{\overline{k}})$. A word will discriminate with high accuracy between a document that belongs to a category from one that does not, if the value of δ_k is small and the value of $\delta_{\overline{k}}$ is large , or vice versa, if δ_k is large and $\delta_{\overline{k}}$ is small. Now, if we set $\delta_k = \Phi^{-1}(\frac{n_{kw}}{n_k})$ and $\delta_{\overline{k}} = \Phi^{-1}(\frac{n_{\overline{k}w}}{n-n_k})$, the Bi-Normal Separation score is equivalent to the distance between these two quantities, $|\delta_{\overline{k}} - \delta_k|$.

2.5 Chi-Square

The chi-square feature selection score, $\chi^2(w, c_k)$, measures the dependence between word w and category c_k. If word w and category c_k are independent $\chi^2(w, c_k)$ is equal to zero. When we select a different set of words for each category we utilise the following score,

$$\chi^2(w, c_k) = \frac{n(n_{kw} n_{\overline{k}\overline{w}} - n_{\overline{k}w} n_{k\overline{w}})^2}{n_k n_w n_{\overline{k}} n_{\overline{w}}}.$$

Again, by averaging over all categories we get a score for selecting a single set of words for all categories.

$$\chi^2(w) = \sum_{k=1}^{m} Pr(c_k) \chi^2(w, c_k).$$

2.6 Odds Ratio

The Odds Ratio measures the odds of word w occuring in documents in category c_k divided by the odds of word w not occuring in documents in category c_k. Mladenic and Grobelnik (1999) find this to be the best score among eleven scores for a Naive Bayes classifier. For category c_k and word w the oddsRatio is given by,

$$OddsRatio(w, c_k) = \frac{\frac{n_{kw}+0.1}{n_k+0.1} / \frac{n_{k\overline{w}}+0.1}{n_k+0.1}}{\frac{n_{\overline{k}w}+0.1}{n_{\overline{k}}+0.1} / \frac{n_{\overline{k}\overline{w}}+0.1}{n_{\overline{k}}+0.1}}$$

where we add the constant 0.1 to avoid numerical problems. By averaging over all categories we get,

$$OddsRatio(w) = \sum_k Pr(c_k) OddsRatio(w, c_k).$$

2.7 Word Frequency

This is the simplest of the feature selection scores. In the study of Yang and Pedersen (1997) they show that word frequency is the third best after information gain and χ^2. They also point out that there is strong correlation between these two scores and word frequency. For each category c_k word frequency for word w, is the number of documents in c_k that contain word w, i.e. $WF(w, c_k) = n_{kw}$.

Averaging over all categories we get a score for each w,

$$WF(w) = \sum_k Pr(c_k) WF(w, c_k) = \sum_k Pr(c_k) n_{kw}.$$

3 Classification Algorithms

To determine the performance of the different feature selection scores, the classification algorithms that we consider are the Multinomial, Poisson and Binary Naive Bayes classifiers (McCallum and Nigam (1998) Lewis (1998) Eyheramendy *et al* (2003)) and the hierarchical probit classifier of Genkin et al (2003). We choose these classifiers for our analysis chiefly because of two reasons. The first one is the different nature of the classifiers. The naive Bayes models are generative models while the probit is a discriminative model. Generative classifiers learn a model of the joint probability $Pr(x, y)$, of the input x and the label y, and make their predictions by using Bayes rule to calculate $Pr(y|x)$. In contrast, discriminative classifiers model $Pr(y|x)$ directly. The second reason is the good performance that they achieve. In Eyheramendy et al (2003) the multinomial model, notwithstanding its simplicity, achieved the best performance among four Naive Bayes models. The hierarchical probit classifier of Genkin *et al* (2003) achieves state of the art performance, comparable to the performance of the best classifiers such as SVM (Joachims (1998)). We decide to include the binary and Poisson naive Bayes models (see Eyheramendy et al (2003) for details) because they allow us to incorporate information of the probability model used to fit the categories of the documents into the feature selection score. For instance, in the Binary Naive Bayes classifiers the features that one can select using the PIP score correspond exactly to the features with the highest posterior inclusion probability. We want to examine whether or not that offers an advantage over other feature selection scores.

4 Experimental Settings and Results

Before we start the analysis we remove common noninformative words taken from a standard *stopword* list of 571 words and we remove words that appear less than three times in the training documents, justifying this with the fact that they are unlikely to appear in testing documents. This eliminates 8,752 words in the Reuters dataset (38% of all words in training documents) and

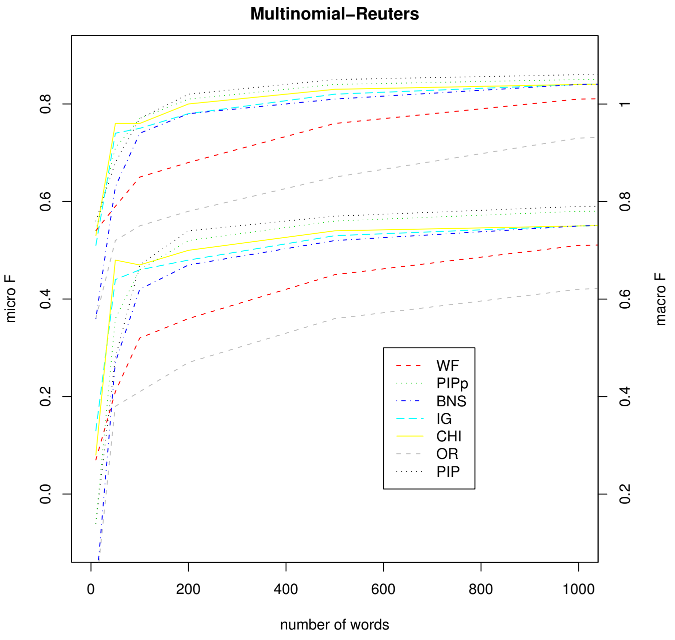


Figure 2: Curves of performance (for the multinomial model) for different number of words measure by macro and micro F (top and bottom sets of curves resp.) for the Reuters dataset.

47,118 words in the Newsgroups dataset (29% of all words in training documents). Words appear on average in 1.41 documents in the Reuters dataset and in 1.55 documents in the Newsgroups dataset.

4.1 Datasets

The 20-Newsgroups dataset contains 19,997 articles divided almost evenly into 20 disjoint categories. The categories topics are related to computers, politics, religion, sport and science. We split the dataset randomly into 75% for training and 25% for testing. We took this version of the dataset from http://www.ai.mit.edu/people/jrennie/20Newsgroups/. Another dataset that we use comes from the Reuters-21578 news story collection. We use a subset of the ModApte version of the Reuters$-21,578$ collection, where each document has assigned at least one topic label (or category) and this topic label belongs to any of the 10 most populous categories - earn, acq, grain, wheat, crude, trade, interest, corn, ship, money-fx. It contains 6,775 documents in the training set and 2,258 in the testing set.

4.2 Experimental Results

In these experiments we compare seven feature selection scores, on two benchmark datasets, Reuters-21578

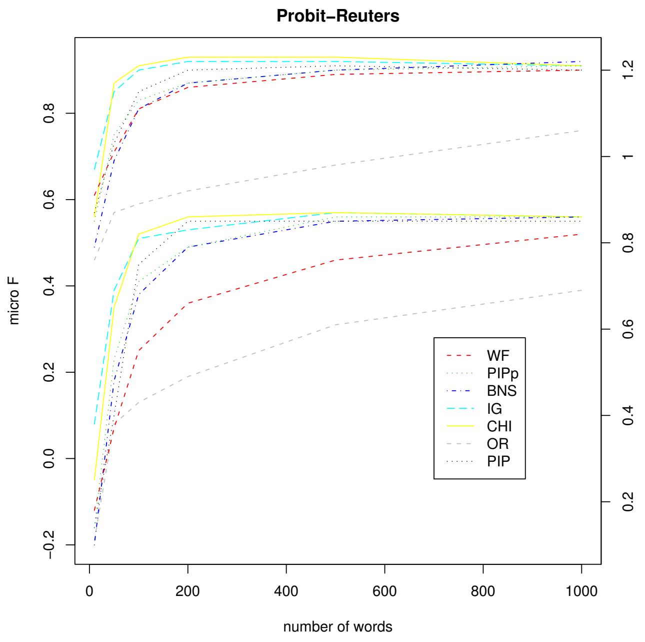


Figure 3: Curves of performance (for the probit model) for different number of words measure by macro and micro F (top and bottom sets of curves resp.) for the Reuters dataset.

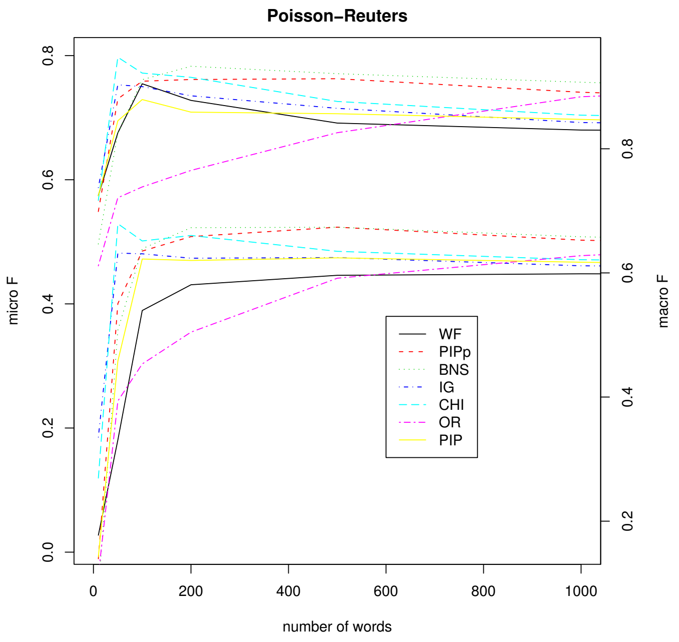


Figure 4: Curves of performance (for the poisson model) for different number of words measure by micro F and macro F (top and bottom sets of curves resp.) for the Reuters dataset.

and Newgroups (see subsection 4.1), under four classification algorithms (see section 3).

We compare the performance of the classifiers for different numbers of words and vary the number of words from 10 to 1000. For larger number of words the classifiers tend to perform somewhat more similarly, and the effect of choosing the words using a different feature selection procedure is less noticeable.

Figure $2-5$ show the micro and macro average F measure for each of the feature selection scores as we vary the number of features to select for the four classification algorithms - multinomial, probit, poisson and binary respectively. In order to have both sets of curves (the curves with the micro F and macro F measures) in the same graph we move them apart. The $y-axes$ for the micro F (macro F) measure correspond to the $y-axes$ on the left (right). The reader will find these figures easier to read in a color rather than black and white rendition.

We noticed that PIP gives, in general, high values to all very frequent words. To avoid that bias we remove words that appear more than 2000 times in the Reuters dataset (that accounts for 15 words) and more than 3000 times in the Newsgroups dataset (that accounts for 36 words).

Reuters. Like the results of Forman (2003), if for scalability reasons one is limited to a small number of features (< 50) the best available metrics are IG and χ^2 as Figures $2-5$ show. For larger number of features (> 50), Figure 2 shows that PIPp and PIP are the best scores for the mutinomial classifier. Figures 4 and 5 shows the performance for the poisson and binary classifiers. PIPp and BNS achive the best performance in the Poisson classifier and PIPp achieves the best performance in the binary classifier. WF performs poorly compare to the other scores in all the classifiers, having the best performance with the poisson.

Newsgroups. χ^2 followed by BNS, IG and PIP are the best performing measures in the probit classifier. χ^2 is also the best one in multinomial model followed by BNS and in the binary classifier with the macro F measure. OR performs best in the poisson classifier. PIPp is best in the binary classifier under the micro F measure. WF performs poorly compare to the other scores in all classifiers. Because of lack of space we do not show graphical display of the performance of the classifiers in the Newsgroups dataset.

In Table $2-3$ we summarize an overall performance of the feature selection scores considered by integrating the curves depicted in Figures $2-5$. Each column corresponds to a feature selection. For instance the number 812 under the header "Multinomial model Reuters-21578" and in the row "micro F" corresponds to the

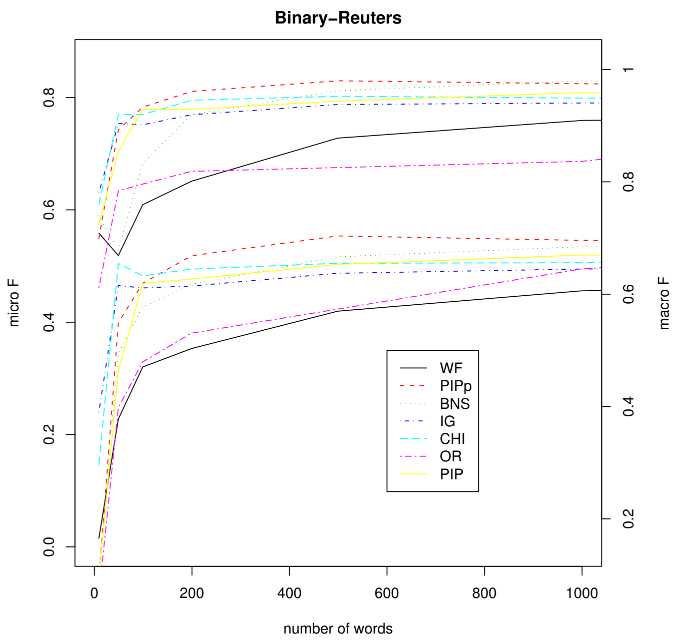


Figure 5: Curves of performance (for the binary naive Bayes model) for different number of words measure by micro F and macro F (top and bottom sets of curves resp.) for the Reuters dataset.

	IG	χ^2	OR	BNS	WF	PIP	PIPp
	Poisson model Reuters-21578 dataset						
micro F_1	708	719	670	763	684	699	755
macro F_1	618	628	586	667	590	618	667
	Poisson model 20-Newsgroups dataset						
micro F_1	753	808	928	812	684	777	854
macro F_1	799	841	936	841	773	813	880
	Berboulli model Reuters-21578 dataset						
micro F_1	779	794	669	804	721	786	822
macro F_1	680	698	618	709	614	696	746
	Bernoulli model 20-Newsgroups dataset						
micro F_1	531	566	508	556	436	534	650
macro F_1	628	673	498	652	505	627	650

Table 2: This table summarizes an overall performance of the feature selection scores considered by integrating the curves depicted in Figures $2-5$. In red are the best performing score and in blue are the second best.

area under the IG top curve in Figure 2. In seven out of sixteen instances χ^2 is the best performing score and in three is the second best. PIPp in four out of sixteen is the best score and in six is the second best. BNS is the best in two and second best in six. In red are the best performing score and in blue are the second best.

5 Conclusion

In this study we introduced a new feature selection score, PIP. The value that this score assigns to each word has an appealing Bayesian interpretation, being the posterior probability of inclusion of the word in a model. Such models assume a probability distribution on the words of the documents. We consider two probability distributions, Bernoulli and Poisson. The former takes into account the presence or absence of words in the documents, and the latter, the number of times each word appears in the documents. Future research could consider alternative PIP scores corresponding to different probabilistic models.

The so-called wrapper approach for feature selection provide an advantage over the filtering approach. The wrapper approach attempts to identify the best feature subset to use with a particular algorithm and dataset, whereas the filtering approach attempts to assess the merits of features from the data alone. The feature selection PIP offers that advantage over feature selection scores that follow the filtering approach, for some classifiers. Specifically, for some naive Bayes models like the Binary naive model or Poisson naive model, the score computed by PIP Bernoulli and PIP Poisson respectively depends on the classification algorithm. Our empirical results do not corroborate in the benefit of using the same model in the feature selection score and in the classification algorithm.

χ^2, PIPp, and BNS are the best performing scores. Still, feature selection scores and classification algorithms seem to be highly data- and model-dependent. The feature selection literature reports similarly mixed findings. For instance, Yang and Pedersen (1997) find that IG and χ^2 are the strongest feature selection scores. They perform their experiments on two datasets, Reuters-22173 and OHSUMED, and under two classifiers, kNN and a linear least square fit. Mladenic and Grobelnik (1999) find that OR is the strongest feature selection score. They perform their experiments on a Naive Bayes model and use the Yahoo dataset. Forman (2003) favors bi-normal separation.

Our results regarding the performance of the different scores are consistent with Yang and Pedersen (1997) in that χ^2 and IG seem to be strong scores for feature selection in discriminative models, but disagree in that WF appears to be a weak score in most instances. Note that we do not use exactly the same WF score. Ours is a weighted average by the category proportion.

	IG	χ^2	OR	BNS	WF	PIP	PIPp
	Multinomial model Reuters-21578 dataset						
micro F_1	812	822	644	802	753	842	832
macro F_1	723	733	555	713	644	762	753
	Multinomial model 20-Newsgroups dataset						
micro F_1	535	614	575	584	456	564	575
macro F_1	594	644	565	634	486	604	585
	Probit model Reuters-21578 dataset						
micro F_1	911	921	674	891	881	901	891
macro F_1	861	861	605	842	753	842	851
	Probit model 20-Newsgroups dataset						
micro F_1	703	723	575	713	565	693	644
macro F_1	693	723	565	703	565	683	624

Table 3: This table summarizes an overall performance of the feature selection scores considered by integrating the curves depicted in Figures 2 – 5. In red are the best performing score and in blue are the second best.

Acknowledgements

We are grateful to David D. Lewis for helpful discussions.

References

Barbieri, M.M. and Berger, J.O. (2004). Optimal predictive model selection. *Annals of Statistics*, **32**, 870–897.

Bernardo, J. M. and Smith, A. F. M. (1994). *Bayesian Theory*. New York: Wiley.

Dash, D. and Cooper, G.F. (2002). Exact model averaging with naive Bayesian classifiers. In: *Proceedings of the Nineteenth International Conference on Machine Learning*, 91-98.

Eyheramendy, S., Lewis, D.D. and Madigan, D. (2003). On the naive Bayes classifiers for text categorization. In *Proceedings of the ninth international workshop on Artificial Intelligence and Statistics*, eds, C.M. Bishop and B.J. Frey.

Forman, G. (2003). An extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*

Genkin, A., Lewis, D.D., Eyheramendy, S., Ju, W.H. and Madigan, D. (2003). Sparse Bayesian Classifiers for Text Categorization, submitted to JICRD.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of ECML-98, 137–142.

Lewis, D.D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of ECML-98, 4–15.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive Bayes text classification. In *AAAI/ICML Workshop on Learning for Text Categorization*, pages 41 – 48.

Miller, A.J. (2002) *Subset selection in regression (second edition)*. Chapman and Hall.

Mladenic, D. and Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive Bayes. Proceedings ICML-99, pages 258-267.

Silvey, S. D. (1975). Statistical Inference. Chapman & Hall. London.

Yang, Y. and Pedersen, J.O. (1997). A comparative study on feature selection in text categorization. Proceedings ICML-97, 412-420.

Text Classification by Augmenting the Bag-of-Words Representation with Redundancy-Compensated Bigrams [*]

Constantinos Boulis[†] Mari Ostendorf[‡]

Abstract

The most prevalent representation for text classification is the bag-of-words vector. A number of approaches have sought to replace or augment the bag-of-words representation with more complex features, such as bigrams or part-of-speech tags, but the results have been mixed at best. We hypothesize that a reason why integrating bigrams did not appear to help text classification is that the new features were not adequately examined for redundancy, i.e. the new feature can be relevant by itself but irrelevant when considered jointly with other features. Searching for optimal feature subsets in the combined space of unigrams and bigrams is prohibitively expensive given that the vocabulary size is in the order of tens of thousands. In this work we propose a measure that evaluates the redundancy of a bigram based only on its unigrams. This approach although suboptimal, since it does not consider interactions between different bigrams or different unigrams, is very fast and targets a main source of bigram redundancy. We apply our feature augmentation measure in three text corpora; the Fisher corpus, a collection of telephone conversations; the 20Newsgroups corpus, a collection of postings to electronic forums; and the WebKB corpus, a collection of web pages. We use Naive Bayes and Support Vector Machines as the learning methods and show consistent gains.

Keywords: Text categorization, Bigrams, 20Newsgroups, WebKB, Fisher

1 Introduction

Text classification is an important instance of the classification problem with unique challenges and requirements. The objective is to classify a segment of text, e.g. a document or a news article, to one (or more) of C possible classes. A number of D tuples $(\vec{x}_d, y_d)$ are presented for training where $\vec{x}_d$ is the vector representation of the d-th document and y_d is a scalar (or set) that indicates the class(es) of the d-th document.

A major challenge of the text classification problem is the representation of a document. The simplest and almost universally used approach is the bag-of-words representation, where the document is represented with a vector of the word counts that appear in it. Depending on the classification method, the bag-of-words vector can be normalized to unity and scaled so that common words are less important than rare words, such as in the tf·idf representation.

Despite the simplicity of such a representation, classification methods that use the bag-of-words feature space often achieve high performance. Over the past, a number of attempts have been made to augment or substitute the bag-of-words representation with richer features. In [12, 4] linguistic phrases, proper names and complex nominals are used and in [20, 16] bigrams are added to the feature space. In [15] character n-grams are used for text classification. A recent comprehensive study [14] surveys the different approaches that have been taken thus far and evaluates them in standard text classification resources. The conclusion is that more complex features do not offer any gain when combined with state-of-the-art learning methods, such as Support Vector Machines (SVM).

We argue that a reason past approaches have failed to show improvements is that they have looked only at the *relevance* of the new features and not *redundancy.* The issues of relevance and redundancy are both central to the choice of optimum feature subset selection [9, 21]. Relevance is the degree to which a feature is useful for classification by itself, and redundancy is the degree to which a feature is correlated with other features. If a feature has high relevance but is also strongly correlated with other equally or more relevant features, adding it to the feature subset can actually hurt classification performance in the typical situation when training is limited. When constructing more complex representations, the number of potential features can increase exponentially. For example, using bigrams increases the vector dimension from V to V^2, where V is the vocabulary size. With so many features, care must be taken to include not simply those that are relevant by themselves but only those that are jointly relevant with the rest of the features.

[*]This work has been supported by NSF grant IIS-0121396.

[†]Dept. of Electrical Engineering, University of Washington, Seattle, USA.

[‡]Dept. of Electrical Engineering, University of Washington, Seattle, USA.

A major problem with determining redundancy is the amount of computations needed. Algorithms such as [9, 11] are of order $O(T^2)$ where T is the original number of features. Adding bigrams as potential features makes such an approach impractical, since $T = V + V^2$ and V is usually on the order of tens of thousands. Even approaches such as [21] with less than quadratic requirements can pose overwhelming computational burdens. In this work, we propose a filter approach to feature selection that determines the redundancy of a bigram based on its unigrams. Although this approach is not optimum, meaning that only a portion of possible feature combinations are examined for redundancy, it is shown that it can offer gains in challenging text classification tasks and that it scales efficiently with vocabulary size and order of word sequences. Performance is not the only reason bigrams are a suitable target for augmenting the feature space. Another important reason is interpretation. A common way to interpret and describe the topics present is to output the top-N discriminative features. Adding bigrams to the list can offer a more natural interpretation, although we have no formal way of measuring this.

2 Adding relevant and non-redundant bigrams

There are two main approaches to the problem of feature selection for supervised learning. The filter approach [7] and the wrapper approach [8]. The filter approach scores features independently of the classifier, while the wrapper approach jointly computes the classifier and the subset of features. A third approach, often called embedded [5], combines the two approaches into one by embedding a filter feature selection method into the process of classifier training, rather than treating the classifier as a black box. While the wrapper approach is arguably the optimum approach, for applications such as text classification where the number of features ranges from dozens to hundreds of thousands it can be prohibitively expensive.

We followed a filter approach to feature selection, and we implemented information gain (IG) since it has been shown before [3] that is one of the best performing methods. The IG measure is given by:

$$IG_w = -H(\mathbf{C}) + p(w)H(\mathbf{C}|w) + p(\bar{w})H(\mathbf{C}|\bar{w}) \tag{2.1}$$

where $H(\mathbf{C}) = \sum_{c=1}^{C} p(c) \log p(c)$ denotes the entropy of the discrete topic category random variable $\mathbf{C}$. Each document is represented with the Bernoulli model, i.e. a vector of 1 or 0 depending if the word appears or not in the document.

We have also implemented another filter feature selection mechanism, the KL-divergence, which is given by:

$$KL_w = D[p(c|w)||p(c)] = \sum_{c=1}^{C} p(c|w) \log \frac{p(c|w)}{p(c)} \tag{2.2}$$

In the KL-divergence we have used the multinomial model, i.e. each document is represented as a vector of word counts. We smoothed the word-topic distributions by assuming that every word in the vocabulary is observed at least 10 times for each topic. All words in the vocabulary are ranked according to KL, the higher the KL score the more topic-specific the word is. KL outperformed IG, in all three corpora used and thus experiments reported here are carried out with KL only.[1]

A problem with measures such as IG and KL is that they do not consider the interactions of features, rather they evaluate each feature independently. Therefore, they have no way of dealing with redundancy. To compensate for that we define the new measure Redundancy-Compensated KL (RCKL) as:

$$RCKL_{w_i w_{i+1}} = KL_{w_i w_{i+1}} - KLw_i - KLw_{i+1} \tag{2.3}$$

Therefore, if a bigram is highly relevant, i.e. $KLw_i w_{i+1}$ is high, but its unigrams are also highly relevant it will be less likely to get added. In words, equation (2.3) can be described as *How much more topic information can $w_i w_{i+1}$ give us compared to its unigrams?* To illustrate the basic idea consider some examples from one of our data sets. For the topic *trials*, the words *commit* and *perjury* are deemed to be important for classification. The bigram *commit perjury*, although being by itself very much relevant, does not add further information than the words *commit* and *perjury*. As another example, the bigram *a holiday* is redundant given that the word *holiday* is already included in the feature subset. Examples of relevant and non-redundant bigrams would be *big brother* for the topic *reality shows*, or *second hand* for the topic *smoking*.

3 Experiments

3.1 Description of corpora used We conducted experiments on three large corpora. The first is the Fisher corpus [1] a collection of 5-minute telephone conversations on a predetermined topic. The topic was selected from a list of 40 before the start of the conversation. After eliminating conversations where at least one of the speakers was non-native or the participants

[1] A measure similar to (2.2) has been suggested in [17]. Although we have not seen an exact mention of (2.2) in the literature, we view this as being variation on a theme and not the main contribution of this paper.

did not follow closely the topic, we were left with 10127 conversations or 20254 conversation sides. There were about 15M words in the collection and conversation sides were unequally divided among the 40 topics. The median number of sides per topic was 478 with a standard deviation of 202 (max 1018, min 198). Only words with 5 or more occurrences were kept, leading to a vocabulary of 23236 words. The Fisher corpus was created to facilitate speech recognition research and, to the best of our knowledge, it has not been used before for text classification. The Fisher corpus brings interesting new challenges to the problem of text classification. It bears the same core characteristics of text classification, such as a very high dimensional space, but unlike other corpora such as Reuters-21578 or 20Newsgroups it consists of transcripts of spoken language. The language is less structured and more spontaneous than written text, including disfluencies such as repetitions, restarts and deletions both at the word and above-word level. An additional difficulty stems from the fact that 14% of words in spoken language text are pronouns vs. 2% in written text [18]. Since pronouns substitute for nouns or noun phrases that are generally considered to convey semantic information, they may have a negative impact on clustering or classification performance. On the other hand, the vocabulary is about half the size of a comparable corpus of written text. Also, conversation classification involves first converting speech into text, which is a procedure that generates errors (state-of-the-art systems achieve a word error rate of about 15%-20% [19]). In this paper we have not dealt with the issue of errorful transcriptions, i.e. the input to the classification algorithms is the human-transcribed conversations. Classifying conversations by topic can be important in a number of scenarios, such as summarizing business meetings or analyzing customer service call-centers.

The second corpus is 20Newsgroups [10], a collection of 18827 postings to electronic discussion forums or newsgroups. There are 20 different classes in 20Newsgroups and the corpus is almost perfectly balanced, i.e. equal number of postings per newsgroup. Preprocessing consisted of converting all numbers to a single token and removing the *From:* field. Words with 5 or more occurrences were kept, resulting in a vocabulary of 34658 words.

The third corpus is a common subset of WebKB [2]. WebKB is a collection of html pages from different categories. In this work we selected 4 classes (faculty, student, project, course) of 4199 pages in total. This is a subset that has been used before [11]. Standard preprocessing was followed, such as keeping only the text of each web page and ignoring hyperlinks and headers and converting numbers to special tokens. The vocabulary of words with 2 or more occurrences consisted of 26087 words.

All three of the corpora are examples of single-label collections, i.e. each document is associated with a single class. A more general setting is a multi-label corpus where a document is associated with a set of classes, not necessarily of fixed length. Examples of multi-label corpora are Reuters-21758 and OHSUMED. Training multi-label classifiers was not investigated in this work.

3.2 Learning methods and evaluation measures

Two learning methods were used throughout our experiments: Naive Bayes [13] and Support Vector Machines (SVM) [6]. The two methods are the most common used for text classification, with Naive Bayes representing a standard baseline and SVM being the state-of-the-art method in text classification. Since our feature augmentation method is a filter approach, we would like to investigate how it performs for more than one classifier. For Naive Bayes we used the *Rainbow* toolkit (http://www-2.cs.cmu.edu/mccallum/bow/rainbow/). For SVM we used the *SVMLight* toolkit (http://svmlight.joachims.org/). Since SVM are inherently binary classifiers and *SVMLight* does not have implemented multi-class approaches to classification, we used the one-vs-one approach. In the one-vs-one approach, given a C-category classification problem, $C*(C-1)/2$ binary classifiers are constructed for every pair of classes. For each pair $\{i, j\}$ a function $H_{ij}(\vec{d})$ is estimated, where $\vec{d}$ is the vector representation of document d. During testing, if $H_{ij}(\vec{d}) > 0$ then $votes(i) = votes(i) + 1$ else $votes(j) = votes(j) + 1$. Document d is assigned to the class with the maximum number of votes $\hat{i} = argmax_i votes(i)$. SVM require much larger computational resources than Naive Bayes, although both can be run in parallel on multiple machines. For Naive Bayes, the feature counts were used as input, while for SVM the tf·idf measure was used. Applying tf·idf or other normalization schemes does not apply in Naive Bayes, since the model assumes a discrete generation mechanism.

Since we operate in a single-label setting, the class with the highest likelihood (for Naive Bayes) or number of votes (for SVM) was selected as output. Classification accuracy was used as the evaluation measure. Micro-F, which is a common evaluation measure in text classification, does not apply in this case since classification accuracy and micro-F are identical for the single-label case.

3.3 Results In all our experiments we used 10 random 80/20 train/test splits and averaged the classifi-

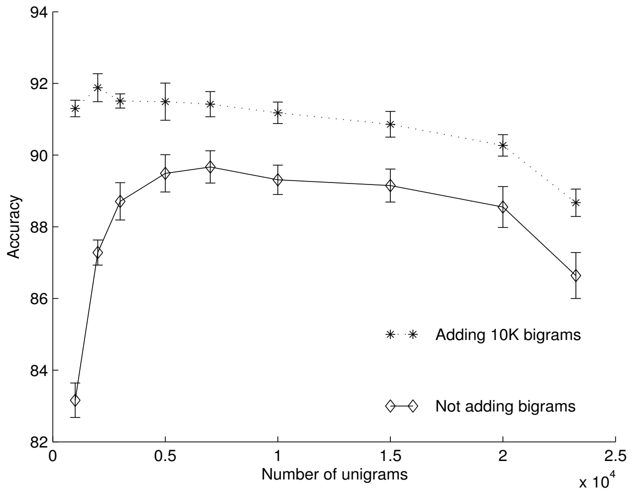


Figure 1: Naive Bayes performance with and without adding bigrams on the Fisher corpus.

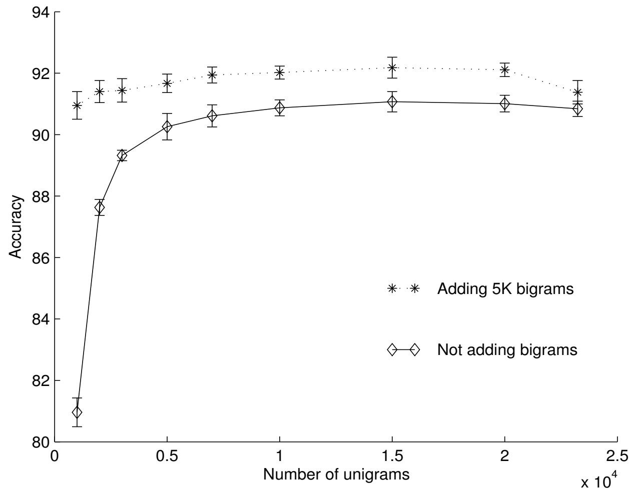


Figure 2: SVM performance with and without adding bigrams on the Fisher corpus.

cation accuracies over all splits. In Table 1 we see the performance of both learning methods, Naive Bayes and SVM, for a varying number of unigrams selected according to (2.2) and bigrams selected according to (2.3). We avoided making a decision on the number of unigrams and bigrams because we wanted to observe the performance of the feature augmentation method for a range of possible features. In addition, it is not always clear what criterion we should use to select the optimum number of features. One choice could be the highest classification accuracy on a held-out set. Another choice could be the ratio of classification accuracy and number of features, so that we prefer classifiers with low numbers of features. From Table 1 we see a clear gain from adding bigrams for both Naive Bayes and SVM. Table 1 also reveals a smooth accuracy variation for different number of bigrams, therefore having an automatic method for determining the number of bigrams should not be radically different from the optimum case. In Figures 1, 2 we plot four columns of Table 1 with the associated standard deviations to show the difference between unigrams-only and mix of unigrams and bigrams. In Table 2 we see the performance of using bigrams-only. We observe that it is the combination of unigrams and bigrams that achieves the highest accuracy rather than unigrams-only or bigrams-only representations. In addition, from Table 1 we can see that by using 1K unigrams and 1K bigrams we achieve the same performance as 7K unigrams or 5K bigrams with Naive Bayes. This can be important when we want the most compact model for the fastest calculation and the smallest memory or disk footprint.

In Table 3 we see the performance of the feature augmentation method on the 20Newsgroups corpus. This corpus is qualitatively different than Fisher. Some of the documents are very small (42 with 5 or less words and 93 with 10 or less words) and the vocabulary is much bigger than Fisher's (34658 vs. 23286). Applying feature selection on unigrams resulted in a slight increase of classification accuracy for up to 30K features and then a constant degradation of performance. The degradation was even worse if IG was used as the feature selection method. In such a task where feature selection does not appear to be important, Naive Bayes did not benefit from augmenting its feature space with bigrams. Performance did not degrade either, which shows that the added features are relevant, given the sensitivity that Naive Bayes has to high-dimensional spaces. SVM gets a small boost of performance by integrating bigrams in the feature space. Using bigrams only did not provide a superior alternative either, as it is shown in Table 4.

In Table 5 we see the performance of the feature augmentation method on the WebKB corpus. Here feature selection appears to be more important than in 20Newsgroups for both Naive Bayes and SVM, even if the vocabulary is much smaller. Adding bigrams offers gains for both Naive Bayes and SVM. In Table 6 we see the performance using bigrams only. Naive Bayes achieves better results than using unigrams only but SVM performance is about the same. Overall, the best text classification accuracy for WebKB is obtained by augmenting the bag-of-words space with bigrams, from 91.62 to 93.02 with standard deviation being for both

Table 1: 10-fold cross validation mean accuracies using a mix of unigrams and bigrams on the Fisher corpus. Bigrams are selected according to (2.3). Standard deviations are in 0.2-0.4 range. Horizontal axis is bigrams, vertical unigrams.

		0	**0.5K**	**1K**	**3K**	**5K**	**10K**	**20K**	**90K**
23286	**NB**	86.64	87.91	87.97	88.21	88.41	88.67	88.61	84.02
	SVM	90.84	91.33	91.28	91.87	91.38	91.22	91.53	90.61
20K	**NB**	88.55	89.25	89.31	89.95	90.25	90.27	90.12	84.62
	SVM	91.01	91.54	91.25	91.53	92.11	91.86	91.85	90.83
15K	**NB**	89.15	90.00	90.11	90.52	90.70	90.86	90.75	85.07
	SVM	91.07	91.19	91.76	91.83	92.18	91.76	91.48	90.39
10K	**NB**	89.31	90.09	90.46	90.53	91.07	91.18	91.38	85.08
	SVM	90.87	91.52	91.40	91.72	92.02	91.61	91.48	90.81
7K	**NB**	89.67	90.38	90.67	90.91	91.14	91.42	91.30	85.07
	SVM	90.61	91.33	91.35	91.43	91.94	91.76	91.73	90.73
5K	**NB**	89.49	90.57	90.70	91.10	91.34	91.49	91.46	85.15
	SVM	90.26	90.86	91.24	91.39	91.67	91.72	91.60	90.30
3K	**NB**	88.71	90.34	90.75	90.97	91.26	91.51	91.45	84.55
	SVM	89.32	90.50	91.11	91.49	91.44	91.65	91.52	90.21
2K	**NB**	87.28	90.16	90.46	90.97	91.38	91.88	91.64	84.29
	SVM	87.63	90.17	90.23	90.93	91.40	91.58	91.48	90.00
1K	**NB**	83.16	88.94	89.87	90.62	91.02	91.30	91.47	83.58
	SVM	80.96	88.90	89.44	90.57	90.95	90.78	90.11	89.88

Table 2: 10-fold cross validation mean accuracies using only bigrams on the Fisher corpus. Bigrams are ranked according to KLw_iw_{i+1}. Standard deviations are in the range 0.2-0.4

	1K	**5K**	**10K**	**20K**	**50K**	**100K**	**150K**	**230K**
NB	85.69	89.00	89.91	90.63	90.71	89.61	87.35	73.60
SVM	80.01	88.25	89.75	90.42	91.02	90.19	90.11	90.23

0.81.

In Table 7 a summary of the results is shown. The highest classification accuracies using each one of the three feature construction methods are shown. It should be noted that in practice a scheme to automatically estimate the number of features should be applied. Table 7 shows that 5 out of 6 times the augmented space is better than the bag-of-words space and 5 out of 6 times better than the bigrams-only space. In no occasion was the augmented space worse than either of the representations on all three corpora and learning methods and for the SVM method (which gave the best results) the augmented space is always better than either individual space.

4 Discussion

In this work, we have shown that incorporating selected bigrams offers improvements over the bag-of-words representation, across a variety of corpora and learning methods. Key to the new representation is that the added bigrams are compensated for redundancy. A bigram is added according to how much more information it brings compared to its unigrams. Therefore, bigrams such as *a holiday*, *the holiday* will not be preferred given that *holiday* is already in the feature set. This work may help dismiss the myth that more complex representations do not help text classification. The implicit assumption was that the bag-of-words representation captures enough of topic information and more complex representations are hard to model, since they considerably increase the dimensionality of the feature space. Moreover, previous attempts to use more complex features were not successful. As a result of this fallacy, research in text classification has mostly focused on learning methods and not on vector representations. The suggested method, although suboptimal since it does not check for redundancy for all pairs of bigrams and unigrams, offers some evidence that design of feature

Table 3: 10-fold cross validation mean accuracies using a mix of unigrams and bigrams on the 20Newsgroups corpus. Bigrams are selected according to (2.3). Standard deviations are in 0.2-0.4 range. Horizontal axis is bigrams, vertical unigrams.

		0	**0.5K**	**1K**	**5K**	**10K**	**20K**	**50K**
34658	**NB**	89.16	89.20	89.14	89.31	89.52	89.41	89.52
	SVM	90.13	90.84	90.93	90.86	91.02	91.13	91.08
30K	**NB**	89.72	88.98	89.36	89.70	89.70	89.34	89.52
	SVM	90.73	90.81	91.14	91.05	91.24	91.27	90.84
25K	**NB**	89.34	89.40	89.47	89.41	89.67	89.42	89.39
	SVM	91.04	90.93	91.08	91.05	91.50	91.26	91.21
20K	**NB**	89.02	88.85	89.08	89.38	89.92	89.67	89.50
	SVM	90.49	91.02	91.02	91.20	91.51	91.38	90.95
15K	**NB**	88.66	88.25	88.41	89.06	89.54	89.30	89.05
	SVM	90.35	90.37	90.73	90.63	91.42	90.87	90.81
10K	**NB**	87.73	87.44	88.01	88.45	89.15	88.86	89.11
	SVM	89.23	89.96	90.13	90.40	90.66	90.55	90.34
5K	**NB**	85.67	85.96	85.98	87.04	87.72	87.58	88.11
	SVM	82.30	83.05	86.77	89.13	89.79	89.81	89.77

Table 4: 10-fold cross validation mean accuracies using only bigrams on the 20Newsgroups corpus. Bigrams are ranked according to KLw_iw_{i+1}. Standard deviations are in the range 0.2-0.4.

	5K	**10K**	**15K**	**20K**	**30K**	**50K**	**100K**	**135K**
NB	80.14	82.08	83.39	84.23	85.42	86.64	87.14	86.14
SVM	N/A	N/A	75.60	81.17	85.03	86.66	87.30	86.75

spaces can be more important than previously considered.

It would be interesting to connect the suggested criterion with the model selection literature. In our work we used an ad-hoc way for identifying non-redundant bigrams. Is there an “optimal” compensation term that could be added when considering the redundancy of a bigram, as in the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC)? This formulation may help extend this criterion in a natural way to higher order n-grams.

References

[1] C. Cieri, D. Miller, and K. Walker. The Fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of the 4th Language Resources and Evaluation Conference (LREC)*, pages 69–71, 2004.

[2] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th meeting of the American Association for Artificial Intelligence (AAAI-98)*, 1998.

[3] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Machine Learning Research*, 3:1289–1305, 2003.

[4] J. Frunkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the WWW. In *Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization*, 1998.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning Research*, 3:1157–1182, 2003.

[6] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. PhD thesis, University of Dortmund, 2002.

[7] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 121–129, 1994.

[8] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[9] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of 16th International Conference on Machine Learning (ICML)*, pages 284–292, 1996.

[10] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

Table 5: 10-fold cross validation mean accuracies using a mix of unigrams and bigrams on the WebKB corpus. Bigrams are selected according to (2.3). Standard deviations are in the 0.6-1.2 range. Horizontal axis is bigrams, vertical unigrams.

		0	**0.5K**	**1K**	**2K**	**5K**	**10K**	**20K**	**50K**
26087	**NB**	85.44	86.02	86.50	87.37	88.01	87.53	87.97	87.70
	SVM	90.12	91.51	91.33	91.10	90.89	91.03	91.26	90.60
20K	**NB**	85.21	86.90	87.47	87.88	87.52	87.95	88.09	87.44
	SVM	90.51	92.00	91.37	90.79	90.75	91.25	90.82	90.58
15K	**NB**	85.61	86.70	86.64	87.47	88.10	87.69	88.53	88.00
	SVM	90.45	91.75	91.31	91.42	91.52	91.18	91.17	91.24
10K	**NB**	84.98	86.57	87.70	87.66	88.12	87.90	88.37	87.72
	SVM	90.91	91.56	91.49	91.61	91.51	92.08	91.74	91.00
5K	**NB**	86.78	89.22	88.65	89.17	88.52	88.59	88.40	88.08
	SVM	91.35	91.71	91.26	91.86	91.68	91.85	91.37	91.21
2K	**NB**	87.25	89.16	89.64	89.47	89.67	89.28	88.64	89.21
	SVM	91.41	91.91	92.08	92.07	92.47	92.28	92.59	91.77
1K	**NB**	87.01	89.61	90.28	90.05	89.77	89.59	89.35	88.67
	SVM	89.79	92.23	92.61	92.84	93.02	93.00	92.06	91.75
0.5K	**NB**	81.75	88.33	89.36	90.10	89.78	89.26	88.69	88.84
	SVM	N/A	N/A	90.95	91.25	91.78	92.17	91.74	91.11

Table 6: 10-fold cross validation mean accuracies using only bigrams on the WebKB corpus. Bigrams are ranked according to KLw_iw_{i+1}. Standard deviations are in the range 0.6-1.2

	1K	**2K**	**3K**	**5K**	**10K**	**20K**	**50K**	**70K**	**110K**
NB	89.22	89.96	90.39	89.95	90.06	90.12	89.51	89.40	88.31
SVM	33.73	65.27	90.70	91.51	91.62	91.41	91.11	91.38	89.14

[11] C. Lee and G.G. Lee. MMR-based feature selection for text categorization. In *Proceedings of the Human Language Technologies/North American Chapter of the Association for Computational Linguistics(HLT/NAACL):short papers*, pages 5–8, 2004.

[12] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

[13] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[14] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR)*, 2004.

[15] F. Peng, D. Schuurmans, and S. Wang. Language and task independent text categorization with simple language models. In *Proceedings of the Human Language Technologies/North American Chapter of the Association for Computational Linguistics conference(HLT/NAACL)*, 2003.

[16] B. Raskutti, H. Ferra, and A. Kowalczyk. Second order features for maximizing text classification performance. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, 2001.

[17] K.M. Schneider. A new feature selection score for multinomial naive bayes text classification based on KL-divergence. In *Proceedings of the 42nd Meeting of the Association of Computational Linguistics (ACL)*, pages 186–189, 2004.

[18] S. Schwarm, I. Bulyko, and M. Ostendorf. Adaptive language modeling with varied sources to cover new vocabulary items. *IEEE Trans. on Speech and Audio Processing*, 12:334–342, May 2004.

[19] A. Stolcke. Speech-to-text research at SRI-ICSI-UW. In *Proceedings of the NIST Rich Transcription Workshop*, 2004.

[20] C.-M. Tan, Y.-F. Wang, and C.-D. Lee. The use of bigrams to enhance text categorization. *Information Processing and Management*, 38:529–546, 2002.

[21] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Machine Learning Research*, 5:1205–1224, 2004.

Table 7: Summary results from all corpora. The best accuracies for each feature construction method are shown. Student's t-test is performed to assess the significance of difference. The last two symbols show if the performance of the augmented representation is statistically different than the unigrams-only and bigrams-only representation respectively at the confidence level of 0.95. A (+) symbol means that the augmented representation is better and a (=) symbol means that the difference is not significant.

		Only 1-grams	**Only 2-grams**	**Mix of 1-grams, 2-grams**		
Fisher	**NB**	89.67	90.71	91.88	(+)	(+)
	SVM	91.07	91.02	92.18	(+)	(+)
20Newsgroups	**NB**	89.72	87.14	89.92	(=)	(+)
	SVM	91.04	87.30	91.51	(+)	(+)
WebKB	**NB**	87.25	90.39	90.28	(+)	(=)
	SVM	91.42	91.62	93.02	(+)	(+)

Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering

Bin Tang[*], Michael Shepherd, Evangelos Milios, Malcolm I. Heywood
{btang, shepherd, eem, mheywood}@cs.dal.ca
Faculty of Computer Science, Dalhousie University, Halifax, Canada, B3H 1W5

Abstract
A great challenge of text mining arises from the increasingly large text datasets and the high dimensionality associated with natural language. In this research, a systematic study is conducted of six Dimension Reduction Techniques (DRT) in the context of the text clustering problem using three standard benchmark datasets. The methods considered include three feature transformation techiques, Independent Component Analysis (ICA), Latent Semantic Indexing (LSI), Random Projection (RP), and three feature selection techniques based on Document Frequency (DF), mean TfIdf (TI) and Term Frequency Variance (TfV). Experiments with the k-means clustering algorithm show that ICA and LSI are clearly superior to RP on all three datasets. Furthermore, it is shown that TI and TfV outperform DF for text clustering. Finally, experiments where a selection technique is followed by a transformation technique show that this combination can help substantially reduce the computational cost associated with the best transformation methods (ICA and LSI) while preserving clustering performance.

Keywords: dimension reduction techniques, ICA, LSI, term frequency variance, mean TfIdf

1 Introduction

Document clustering is the fundamental enabling tool for efficient document organization, summarization, navigation and retrieval for very large datasets. The most critical problem for text clustering is the high dimensionality of the natural language text. The focus of this research is to investigate the relative effectiveness of various dimension reduction techniques (DRT) for text clustering.

There are two major types of DRTs, feature transformation and feature selection [17]. In feature transformation, the original high dimensional space is projected onto a lower dimensional space, in which each dimension in the lower dimensional space is some linear or non-linear combination of the original high dimensional space. Widely used examples include, Principal Components Analysis (PCA), Factor Analysis, Projection Pursuit, Latent Semantic Indexing (LSI), Independent Component Analysis (ICA), and Random Projection (RP) [8]. Feature selection methods only select a subset of "meaningful or useful" dimensions (specific for the application) from the original set of dimensions. For text applications, some feature selection methods for text applications include, Document Frequency (DF), mean TFIDF (TI), Term Frequency Variance (TfV).

Although many research projects are actively engaged in furthering DRTs as a whole, so far, there is a lack of experimental work comparing them in a systematic manner especially for text clustering task. In our previous work [18], we compared four of the abovementioned methods (including ICA, LSI, RP, DF) on five benchmark datasets. Considering both the effectiveness and robustness of all the methods, in general, we can rank the four DRTs in the order of ICA > LSI > DF > RP. ICA demonstrates good performance and superior stability compared to LSI. Both ICA and LSI can effectively reduce the dimensionality from a few thousands to the range of 100 to 200 or even less. Though providing superior performance, the computation cost of ICA is much higher compared to DF. In [18], we pointed out the need to find proper feature selection methods to pre-screen dimensions before the ICA computation to reduce the computational cost of ICA without sacrificing performance.

In this work, we investigate the relative effectiveness and robustness of six dimension reduction techniques when used for text clustering using three benchmark datasets. The DRTs are Document Frequency (DF), mean TFIDF (TI), Term Frequency Variance (TfV), Latent Semantic Indexing (LSI), Random Projection (RP) and Independent Component Analysis (ICA). We also demonstrate the effectiveness of combining TI or TfV with ICA as a computationally cheaper alternative to the default ICA with full dimensions.

This paper is organized as follows. Section 2 provides more details for the DRTs used in this research. Section 3 describes our experimental procedure, evaluation methods and dataset issues. Section 4 presents our experimental results and appropriate discussion notes. Finally, conclusions are drawn and future research di-

[*]corresponding author

rections identified in Section 5.

2 Dimension Reduction Techniques for Text Clustering

In the discussion, we will use the following notation. A document collection is represented by its term-document matrix $X$of dimension m by n, where m is the number of terms and n the number of documents.

2.1 Feature Selection Methods Feature Selection methods sort terms on the basis of a numerical measure computed from the document collection to be clustered, and select a subset of the terms by thresholding that measure. In this section, we will describe the mathematic details of three feature selection methods, including Document Frequency (DF) in Section 2.1.1, Mean TFIDF (TI) in Section 2.1.2 and Term Frequency Variance (TfV) in Section 2.1.3.

2.1.1 Document Frequency (DF) Document Frequency (DF) may itself be used as the basis for feature selection. That is, only those dimensions with high DF values appear in the feature vector. DF can be formally defined as follows. For a document collection X of m terms by n documents, the DF value of term t, DF_t, is defined as the number of documents in which t occurs at least once among the n documents. To reduce the dimensionality of X from m to k ($k < m$), we choose to use the k dimensions (terms) with the top k DF values. It is obvious that the DF takes $O(mn)$ to evaluate. In spite of its simplicity, it has been demonstrated to be as effective as more advanced techniques in text categorization [19].

2.1.2 Mean TFIDF (TI) In information retrieval (IR), we value a term with high term frequency but low document frequency as a good indexing term. In IR, we generate a vector representation for each document d_j, where the weight for each term t in document d_j is its $tfidf$ value, defined as:

$$tfidf_j = tf_j \log \frac{|T_r|}{DF_t}$$

where

$$tf_j = \begin{cases} 1 + \log t_j & \text{if } t_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

and T_r is the total number of documents in collection X, DF_t is the document frequency of term t, t_j is the frequency of term t in document d_j. In this work, we propose to use the mean value of $tfidf$ over all the documents (hereafter referred to as TI) for each term as a measure of the quality of the term. The higher the TI value, the better the term to be ranked.

2.1.3 Term Frequency Variance (TfV) The TfV method for ranking term quality was demonstrated to successfully reduce the dimension to only 15% of the original dimension [6, 13]. The basic idea is to rank the quality of a term based on the variance of its term frequency. This is similar in spirit to the intuition of TI method. The term frequency of term t in document d_j, tf_j, is defined the same way as in Section 2.1.2. The quality of term t is calculated by

$$\sum_j^n tf_j^2 - \frac{1}{n}\left[\sum_j^n tf_j\right]^2$$

where n is the total number of documents.

2.2 Feature Transformation Methods Feature transformation methods perform a transformation of the vector space representation of the document collection into a lower dimensional subspace, where the new dimensions can be viewed as linear combinations of the original dimensions. In this section, we will introduce some mathematic details of the three feature transformation methods, i.e., Latent Semantic Indexing (LSI) in Section 2.2.1, Random Projection (RP) in Section 2.2.2 and Independent Component Analysis (ICA) in Section 2.2.3.

2.2.1 Latent Semantic Indexing (LSI) LSI, as one of the standard dimension reduction techniques in information retrieval, has enjoyed long-lasting attention [2, 5, 7, 10, 15, 16]. By detecting the high-order semantic structure (term-document relationship), it aims to address the ambiguity problem of natural language, i.e., the use of synonymous, and polysemous words, therefore, a potentially excellent tool for automatic indexing and retrieval.

LSI uses Singular Value Decomposition (SVD) to embed the original high dimensional space into a lower dimensional space with minimal distance distortion, in which the dimensions in this space are orthogonal (statistically uncorrelated). During the SVD process, the newly generated dimensions are ordered by their "importance". Using the full rank SVD, the term-document matrix X is decomposed as $X = USV^T$, where S is the diagonal matrix containing singular values of X. U and V are orthogonal matrices containing left and right singular values of X, often referred to as term projection matrix and document projection matrix respectively. Using truncated SVD, the best rank-k approximation (in least-squares sense) of X is $X_k \cong U_k S_k V_k^T$, in which X is projected from m dimensional space to k dimensional space ($m > k$). In the new k-dimension, each original document d can be re-

represented as $\tilde{d} = U_k S_k d^T$. The truncated SVD not only captures the most important associations between terms and documents, but also effectively removes noise and redundancy and word ambiguity within the dataset [5]. One major drawback of LSI is its high computational cost. For a data matrix, X, of dimension $m \times n$, the time complexity to compute LSI using the most commonly used *svd* packages is in the order of $O(m^2 n)$ [15]. For a sparse matrix, the computation can be reduced to the order of $O(cmn)$, where c is the average number of terms in each document [16].

2.2.2 Random Projection (RP) As a computationally cheaper alternative to LSI for dimension reduction with bounded distance distortion error, the method of Random Projection (RP) has recently received attention from the machine learning and information retrieval communities [1, 4, 9, 12, 15]. Unlike LSI, the new dimensions in RP are generated randomly (random linear combinations of original terms) with no ordering of "importance". The new dimensions are only approximately orthogonal. However, researchers don't seem to agree on the effectiveness and computational efficiency of RP as a good alternative for LSI-like techniques [4, 9, 12, 15]. So far, the effectiveness of RP is still not clear, especially in the context of text clustering.

Similar to LSI, RP projects the columns of term-document matrix X from the original high dimensional space (with m dimensions) onto a lower k-dimensional space using a randomly generated projection matrix R_k of shape $k \times m$, where the columns of R are unit length vectors following a Gaussian distribution. Under the new k dimension space, X is approximated as $X_k \cong R_k X$.

2.2.3 Independent component analysis (ICA) A recent method of feature transformation called Independent Component Analysis (ICA) has gained widespread attention in signal processing [11]. It is a general-purpose statistical technique, which tries to linearly transform the original data into components that are maximally independent from each other in a statistical sense. Unlike LSI, the independent components are not necessarily orthogonal to each other, but are statistically independent. This is a stronger condition than statistical uncorrelateness, as used in PCA or LSI [11]. In most of applications of ICA, PCA is used as a preprocessing step, in which the newly generated dimensions are ordered by their importance. Based on the PCA transformed data matrix, ICA further transforms the data into independent components. Therefore, using PCA as a preprocessing step, ICA can be used as a dimension reduction technique. Until very recently, there were only a few experimental works in which ICA is applied to text data [3, 14].

ICA assumes each observed data item (a document) x to have been generated by a mixing process of statistically independent components (latent variables s_i). Formally, for the term-document matrix $X_{m \times n}$, the noise-free mixing model can be written as $X_{m \times n} = A_{m \times k} S_{k \times n}$, where A is referred to as the mixing matrix and $S_{k \times n}$ is the matrix of independent components. The inverse of A, A^{-1}, is referred as the unmixing matrix, W. The independent components can be expressed as $S_{k \times n} = W_{k \times m} X_{m \times n}$. Here, W is functionally similar to the projection matrix R in RP that project X from the m dimensional space to a lower k dimensional space.

In this research, we used the most commonly used $FastICA$ implementation [11]. $FastICA$ is known to be robust and efficient in detecting the underlying independent components in the data for a wide range of underlying distributions [8]. The mathematical details of $FastICA$ can be found in [11].

In practical applications of $FastICA$, there are two pre-processing steps. The first is *centering*, i.e., making x into zero-mean variables. The second is *whitening*, which means that we linearly transform the observed vector x into x^{new}, such that its components are uncorrelated and their variance equals unity. Whitening is done through PCA. In practice, the most time consuming part of $FastICA$ is the whitening, which can be computed by the *svds* MATLAB^{TM} function.

3 Evaluation

In this section, we present the evaluation methods and experimental setups in Section 3.1, followed by the description of the datasets used in Section 3.2, and ended with the description of the preprocessing procedure in Section 3.3.

3.1 Evaluation Methods and Experimental Setup The judgment of the relative effectiveness of the DRTs for text clustering is based on the final clustering results after different DRTs are applied. The final ranking of DRTs depends on both the absolute clustering results and the robustness of the DRT. Here, good robustness implies that when using a certain DRT, reasonably good clustering results remain relatively stable across a relatively wide range of reduced dimensions.

The quality of text clustering is measured by micro-average of *classification accuracy* (hereafter referred to as CA) over all the clusters, a similar measure to *Purity* as introduced in [20]. To avoid the bias from the training set, CA is only computed based on the test data in the following fashion. The clustering process is only based on the training set. After clustering, each cluster

i is assigned a class label T_i based on the majority vote from its members' classes using only training data. Then, assign each point in test set to its closest cluster. The CA_i for cluster i is defined as the proportion of points assigned as members of cluster i in the test set whose class labels agree with T_i. The total CA is micro-averaged over all the clusters. The comparison between two methods is usually based on student t-test.

Since k-means or its variants are the most commonly used clustering algorithms used in text clustering, we choose to use k-means with our modification to do text clustering. A well-known problem for k-means is that poor choices of initialization often lead to poor convergence to sub optimal solutions. To ameliorate the negative impact of poor initialization, we devised a simple procedure, $InitKMeans$, to pre-select "good" seeds for k-means clustering. It has been proved very effective in our previous work [18]. Our experiments for all the DRTs follow the same general procedure. A sketch of our procedure is as follows, details of our experimental procedure including $InitKMeans$ can be found elsewhere [18].

```
1.  Each dataset is split randomly into training and
    testing set of ratio 3:1 proportionally to their
    category distribution.
2.  For each DRT, run a series of reduced dimensions
    For each desired dimension $k$,
        Apply DRT only to the training data,
        producing proper projection matrix $PR$
        (in feature transformation), or, subset of
        selected dimensions $SD$ (feature selection);
        Apply $PR/SD$ to both training and test set;
        Clustering on the reduced training set;
        Assign $T_i$ to each cluster in reduced
        training set;
        Compute $CA$ using reduced test set;
    End For
```

3.2 Dataset Characteristics In our experiments, we used a variety of datasets of different genres, which include WWW-pages (WebKB[1]), newswire stories (Reuters-21578[2]), and technical reports (CSTR[3]). These datasets are widely used in information retrieval and text mining research. The number of classes ranges from 4 to 50 and the number of documents ranges between 4 and 3807 per class. Table 1 summarizes the characteristics of the datasets.

Reuters-2, a subset of Reuters-21578 dataset, is a collection of documents each document with a single topic label. The version of Reuters-2 that we used eliminates categories with less than 4 documents, leaving only 50 categories. WebKB4 is a subset of WebKB dataset, which is limited to the four most common categories: student, faculty, course, and project. The CSTR dataset contains 505 abstracts of technical reports, divided into four research areas: AI, Robotics and Vision, Systems, and Theory.

3.3 Preprocessing The pre-processing of the datasets follows the standard procedures, including removal of the tags and non-textual data, stop word removal[4], and stemming[5]. Then we further remove the words with low document frequency. For example, for the Reuters-2 dataset we only selected words that occurred in at least 4 documents. The word-weighting scheme we used is the *ltc* variant of the *tfidf* function, defined in Section 2.1.2.

4 Experimental Results

For each given dataset, we applied six DRTs for a complete comparative study. First, we concentrate on comparing the feature selection methods. The results are described in detail in Section 4.1. The comparison results of feature transformation methods are mainly extracted from our previous work [18], which will be summarized in Section 4.2. Based on the results from both DRT method groups, we choose to use TI and TfV as thresholding methods to pre-select subset of dimensions to be further processed by ICA. We focus on comparing the results of ICA with TI/TfV thresholding at different threshold levels against the default version of ICA without TI/TfV thresholding. Here, the threshold levels are defined as the top $x\%$ of selected dimensions using TI or TfV. In this set of experiments, we use TI (or TfV) to pre-select the top $x\%$ of dimensions and pass on the dataset with reduced dimensions to the ICA computation. The results are described in detail in Section 4.2. For completeness, we compile all the comparison results in one figure 1-3 for each dataset. In each figure, there are four subfigures, describing the results of feature transformation methods, results of feature selection methods, results of ICA with TI thresholding, and results of ICA with TfV thresholding respectively.

The comparison of any two methods is based on Student paired t-test comparing the performance of the

[1]http://www2.cs.cum.edu/afs/cs/project/theo-11/www/wwkb

[2]http://www.cs.cmu.edu/TextLearning/datasets.html

[3]http://www.cs.rochester.edu/trs

[4]http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

[5]http://www.tartarus.org/~martin/PorterStemmer/

Datasets	Dataset size $\lvert terms\rvert \times \lvert docs\rvert$	#classes	Class Size range	Type
Reuters 2	7315 x 8771	50	[4, 3807]	News
WebKB4	9870 x 4199	4	[504, 1641]	University Web pages
CSTR	2335 x 505	4	[76, 191]	Technical Reports

Table 1: Summary of the datasets

two methods over a dimension range. The dimension range, denoted as $[k1, k2]$, is usually hand-picked, such that, within such a range, the two methods cannot be clearly differentiated visually, and beyond this range, the performance of the two comparing methods are too poor to be of interest.

4.1 Comparing Feature Selection Methods We performed mutual comparison among DF, TI and TfV for all the three datasets using paired Student t-test. The p values are reported in In Table 2. For the paired Student t-test, the null hypothesis, H_0, assumes $\mu_{X-Y} = 0$. Here X represents the methods listed in rows in Table 2, while Y represent methods listed in columns in Table 2. The alternative hypothesis, H_a, assumes $\mu_{X-Y} > 0$. For Reuters-2, the comparisons are performed over the the dimension range of $[70, 1095]$. Based on the p values of the paired t-test, the null hypothesis, $\mu_{DF-TI} = 0$ is weakly rejected, and $\mu_{DF-TfV} = 0$ is strongly rejected and $\mu_{TI-TfV} = 0$ holds. Therefore, for Reuters-2, we can say that DF systematically performs worse than TI and TfV, and there is no statistical difference between TI and TfV. For WebKB4, the comparisons are performed over the dimension range of $[80, 1980]$. The resulting p values indicate that there is no significant different among DF, TI and TfV, even though TI and TfV provide better CA results than DF. For CSTR, the comparisons are performed over the range of dimensions $[115, 989]$. The resulting p values indicate that there is no significant difference between DF and TfV and between TI and TfV, while DF is worse than TI with slight significance.

Considering all the comparison results, TI and TfV are better feature selection methods than DF for text applications. Therefore, we choose to use TI and TfV as pre-screening methods for ICA in subsequent experiments.

4.2 Results of Feature Transformation Methods and Thresholded ICA In the following, we will describe the results by the order of dataset. For each dataset, we will remark on the comparison results for feature transformation methods based on our previous work [18] for completeness. We will focus on the comparisons between the performance of default ICA and ICA preceded by TI/TfV thresholding. The comparison results are reported based on the p values in separate Tables 3,4,5.

Reuters-2 Results Based on the results of our previous work [18], comparing ICA, LSI and RP, we observed that both ICA and LSI achieve superior results with low dimensionalities ([30,93]) comparing to RP. Within the dimension range of [30,93], ICA not only shows a superior performance over LSI in terms of classification accuracy but also demonstrates better stability than LSI.

The results of comparing the plain ICA (with no pre-selection of dimensions) with that of ICA with pre-selection of dimensions by TI/TfV are reported in Table 3. The null hypothesis, H_0, assumes $\mu_{X-Y} = 0$. Here, X refers to plain ICA, while Y represents ICA with different TI/TfV thresholding levels. The alternative hypothesis, H_a, assumes $\mu_{X-Y} > 0$. Another alternative hypothesis, H_b, assumes $\mu_{X-Y} < 0$. [6] The comparisons are performed over that dimension range of $[10, 153]$. In Table 3, the p values clearly indicate that the plain ICA performs significantly better than ICA with TI-thresholding levels of 5-15%. But there are no significant differences between the plain ICA and ICA with TI-thresholding levels of 20-25% . Similarly, the plain ICA performs significantly better than ICA with TfV-thresholding levels of 5-20%. Interestingly, p value indicates that the ICA with TfV-thresholding level of 25% performs significantly better than the basic ICA.

WebKB4 Results Based on our previous work, we observe that the best performance of ICA is slightly worse than that of LSI [18]. But ICA shows much stable performance over longer range of dimensions than LSI. Both LSI and ICA are better than RP.

In Table 4, we reported the results of combining ICA with TI/TfV thresholding. The comparisons between the plain ICA and those ICAs with TI/TfV thresholding are performed over the range of $[7, 90]$. The p values indicate clearly that the plain ICA is significantly better than ICAs with TI-thresholding levels of 5% and 20%. But there is no significant

[6] We used the same hypothesis tests for Table 4, 5,therefore, not stated explicitly later.

Reuters-2				WebKB4				CSTR			
	DF	TI	TfV		DF	TI	TfV		DF	TI	TfV
DF	N/A	0.07	0.01	DF	N/A	0.16	0.16	DF	N/A	0.04	0.13
TI	0.93	N/A	0.32	TI	0.84	N/A	N/A	TI	0.96	N/A	0.31
TfV	0.99	0.68	N/A	TfV	0.84	N/A	N/A	TfV	0.87	0.69	N/A

Table 2: P Values of Student Paired t-test for Comparing Feature Selection Methods

.

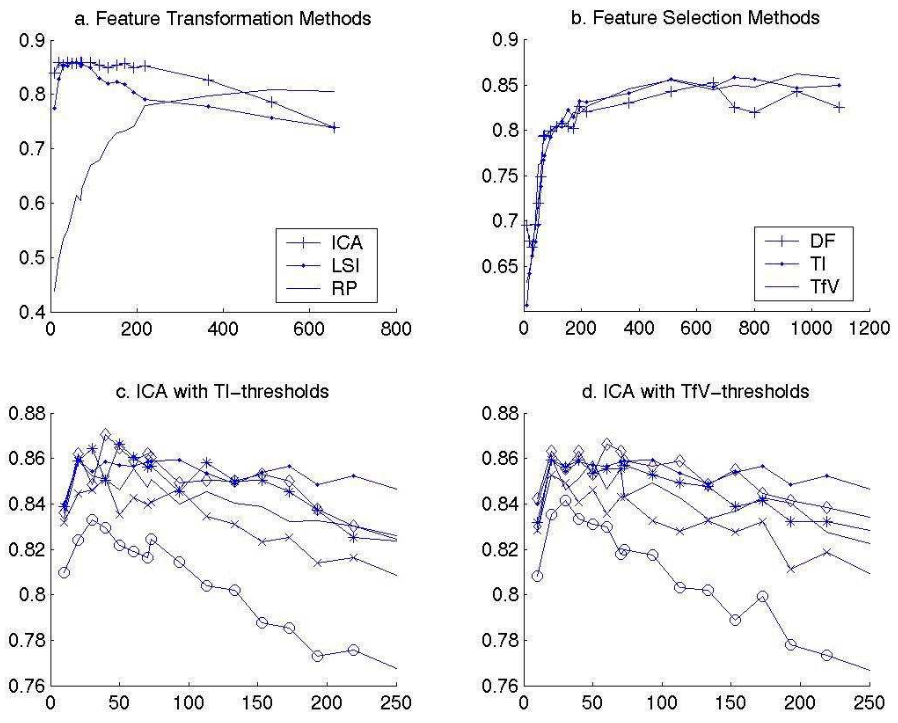


Figure 1: Comparison results of Reuters-2. In all the sub-figures, the x-axis denotes the dimensionality, and the y-axis represents classification accuracy (CA). (a) results of feature transformation method. '+' denotes ICA, '.' denotes LSI, '-' denotes RP. (b) results of feature selection methods. '+' denotes DF, '.' denotes TI, '-' denotes TfV. (c) results of ICA with different level of TI thresholding. 'o' denote thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%,'◇' 25%, and with '.' for plain ICA with full dimensions. (d) results of ICA with different levels of TfV thresholding, 'o' denotes thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%, '◇' 25%, and with '.' for basic ICA

		ICA with TI thresholding					ICA with TfV thresholding				
		5%	10%	15%	20%	25%	5%	10%	15%	20%	25%
H_a	p-value	0.00	0.00	0.00	0.44	0.63	0.00	0.00	0.00	0.01	0.96
H_b	p-value	1.00	1.00	1.00	0.56	0.37	1.00	1.00	1.00	0.99	0.04

Table 3: P-values of the results of ICA combined with TI/TfV thresholding (Reuters-2)

difference between the plain ICA and ICAs with TI-thresholding level of 10%, 15% and 25%. For TfV thresholding, the plain ICA is better than ICA with TfV thresholding level of 5 % with significance and better than 10% with slight significance. But there is no significant difference between the plain ICA and ICAs with TfV-thresholding level of 15-25%.

CSTR Results From our previous work, we observed no significant difference between ICA and LSI for the dimension range of $[5, 33]$. ICA and LSI are better than RP [18].

The results of combining ICA with TI/TfV thresholding are reported in Table 5. We compared the performance of the plain ICA with those of ICAs with TI/TfV thresholdings over the dimension range of $[5, 43]$. Based on the p values, we conclude that the plain ICA is significantly better than ICAs with TI thresholding levels of 5-15% , and there is no significant difference between the plain ICA and ICAs with TI thresholding levels of 20-25%. For TfV thresholding, the plain ICA is better than ICAs with TfV thresholding levels of 5-15%, and there is no significant difference between the plain ICA and ICAs with TfV thresholding levels of 20-25% .

5 Conclusion and Future Work

In this research, we compared the performance of six DRT methods when applied to text clustering problem using three benchmark datasets of distinct genres. Based on all the results, we have observed the following. For feature transformation methods, we can rank $ICA > LSI > RP$ considering classification accuracy and stability. Both ICA and LSI reach their best performance with very low dimensionality, often less than 100 and occasionally lower than 10. ICA and LSI maintain their best performances over a wide range dimensions. ICA appears more stable than LSI. For feature selection methods, DF is inferior comparing to TI and TfV. The best results of TI and TfV can match those of ICA and LSI but at much higher dimensions. The results of combining ICA with TI or TfV thresholding are most interesting. For most of the cases, it is safe to say that ICA with TI or TfV thresholding level 20% performs at least the same as the basic ICA if not better occasionally. This is interesting, since the bottleneck of computing ICA is its preprocessing PCA step (takes $O(m^2n)$ to compute, where m is the dimensionality, and n is the number of points). With pre-screening the dimensions by TI or TfV methods, theoretically, we reduce the computational cost of PCA to 1/25 of the original cost without sacrificing performance.

From our previous and current research, we identify the "ideal" dimension reduction technique for text clustering to be ICA. Though we have achieved moderate success in reducing the computational cost of ICA, we believe that further research should be focused on this issue. Different sampling techniques should be able to provide even more fruitful success in reducing the computational cost of ICA without sacrificing its performance.

References

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of PODS*, pages 274–281, 2001.

[2] M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[3] E. Bingham, A. Kabán, and M. Girolami. Topic identification in dynamical text by complexity pursuit. *Neural Processing Letters*, 17(1):69–83, 2003.

[4] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proc. SIGKDD*, pages 245–250, 2001.

[5] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[6] I. S. Dhillon, J. Kogan, , and M. Nicholas. Feature selection and document clustering. In M.W. Berry, editor, *A Comprehensive Survey of Text mining.* Springer-Verlag, 2003.

[7] C. H. Ding. A probabilistic model for dimensionality reduction in information retrieval and filtering. In *Proc. of 1st SIAM Computational Information Retrieval Workshop*, 2000.

[8] I.K. Fodor. A survey of dimension reduction techniques. Technical report UCRL-ID-148494, LLNL, 2002.

[9] D. Fradkin and D. Madigan. Experiments with random projection for machine learning. In *Proc. SIGKDD*, pages 517–522, 2003.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. SIGIR*, pages 50–57, 1999.

[11] A. Hyvarinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000. FastICA package: http://www.cis.hut.fi/∼xaapo/.

[12] S. Kaski. Dimensionality reduction by random mapping. In *Proc. Int. Joint Conf. on Neural Networks*, volume 1, pages 413–418, 1998.

[13] J. Kogan, C. Nicholas, and V. Volkovich. Text mining with information-theoretical clustering. *Computing in Science and Engineering*, accepted May 2003.

[14] T. Kolenda, L. K. Hansen, and S. Sigurdsson. Independent components in text. In *Advances in Independent Component Analysis*, pages 229–250. Springer-Verlag, 2000.

[15] J. Lin and D. Gunopulos. Dimensionality reduction by

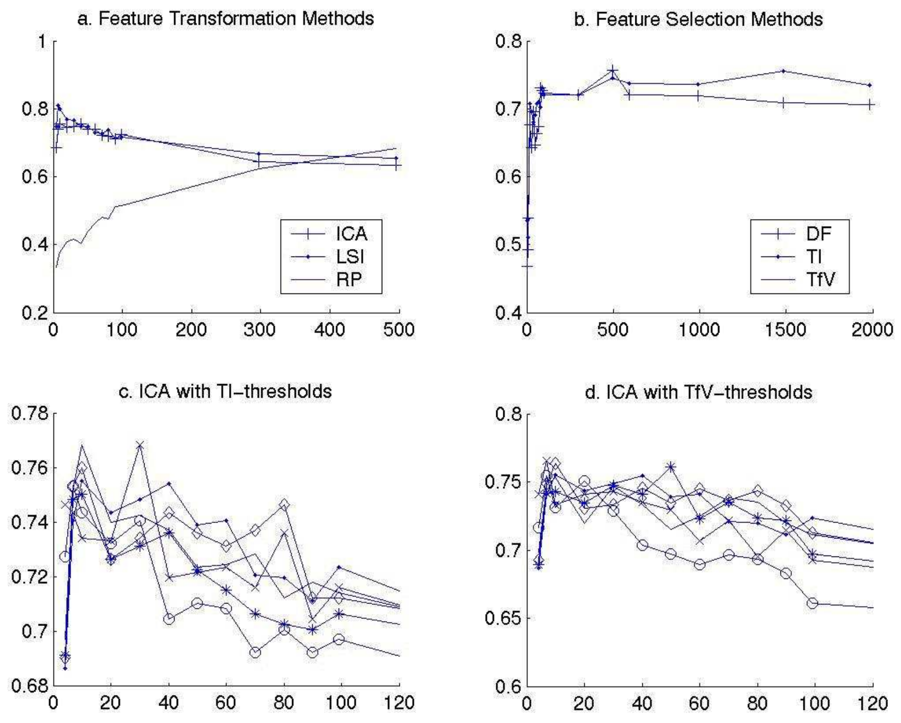


Figure 2: Comparison results of WebKB4. In all the sub-figures, the x-axis denotes the dimensionality, and the y-axis represents classification accuracy (CA). (a) results of feature transformation method. '+' denotes ICA, '.' denotes LSI, '-' denotes RP. (b) results of feature selection methods. '+' denotes DF, '.' denotes TI, '-' denotes TfV. (c) results of ICA with different level of TI thresholding. 'o' denote thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%,'⋄' 25%, and with '.' for plain ICA with full dimensions. (d) results of ICA with different levels of TfV thresholding, 'o' denotes thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%, '⋄' 25%, and with '.' for basic ICA

		ICA with TI thresholding					ICA with TfV thresholding				
		5%	10%	15%	20%	25%	5%	10%	15%	20%	25%
H_a	p-value	0.00	0.10	0.24	0.00	0.56	0.00	0.06	0.22	0.47	0.80
H_b	p-value	0.99	0.90	0.76	1.00	0.44	1.00	0.94	0.78	0.53	0.2

Table 4: P-values of the results of ICA combined with TI/TfV thresholding (WebKB4)

		ICA with TI thresholding					ICA with TfV thresholding				
		5%	10%	15%	20%	25%	5%	10%	15%	20%	25%
H_a	p-value	0.00	0.03	0.00	0.08	0.80	0.00	0.01	0.05	0.06	0.93
H_b	p-value	1.00	0.97	1.00	0.92	0.20	1.00	1.00	0.95	0.94	0.07

Table 5: P-values of the results of ICA combined with TI/TfV thresholding (CSTR)

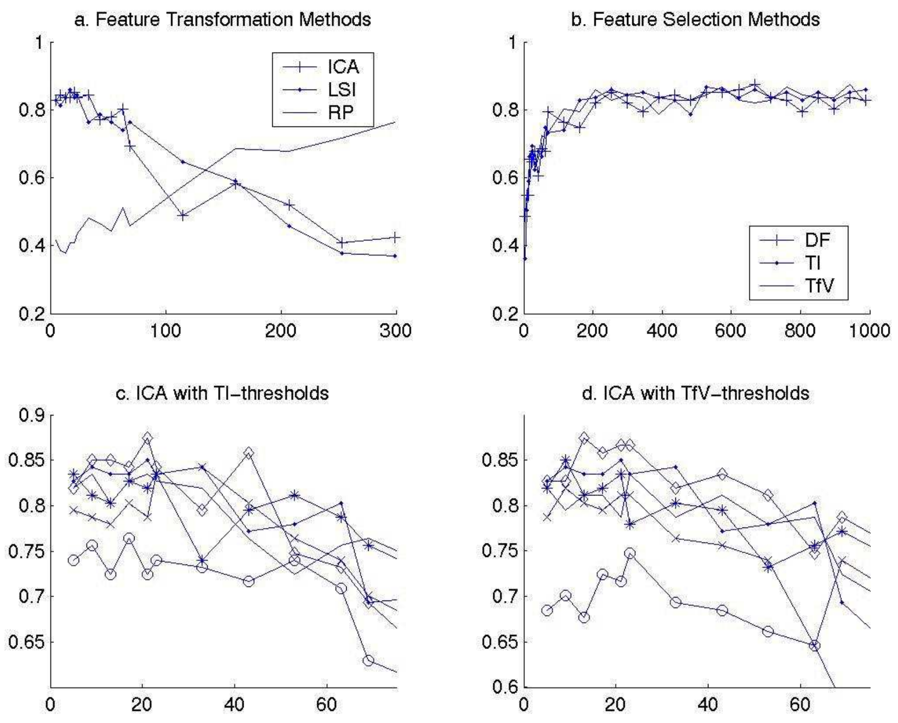


Figure 3: Comparison results of CSTR dataset. In all the sub-figures, the x-axis denotes the dimensionality, and the y-axis represents classification accuracy (CA). (a) results of feature transformation method. '+' denotes ICA, '.' denotes LSI, '-' denotes RP. (b) results of feature selection methods. '+' denotes DF, '.' denotes TI, '-' denotes TfV. (c) results of ICA with different level of TI thresholding. 'o' denote thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%,'◇' 25%, and with '.' for plain ICA with full dimensions. (d) results of ICA with different levels of TfV thresholding, 'o' denotes thresholding level 5%, 'x' 10%, '-' 15%, '*' 20%, '◇' 25%, and with '.' for basic ICA

random projection and latent semantic indexing. In *Proc. SDM'03 Conf., Text Mining Workshop*, 2003.

[16] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. ACM SIGPODS*, pages 159–168, 1998.

[17] L. Parsons, E. Hague, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter, Special issue on learning from imbalanced datasets*, 6(1):90–105, 2004.

[18] B. Tang, X. Luo, M.I. Heywood, and M. Shepherd. A comparative study of dimension reduction techniques for document clustering. Technical Report CS-2004-14, Faculty of Computer Science, Dalhousie University, 2004. http://www.cs.dal.ca/research/techreports/2004/CS-2004-14.shtml.

[19] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. ICML*, pages 412–420, 1997.

[20] Y Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, Department of Computer Science, University of Minnesota, 2001. http://cs.umn.edu/karypis/publications.

Near-Optimal Feature Selection

Jaekyung Yang
IT Services Research Division
Electronics and Telecommunications Research Institute
Daejon, 305-350 Korea

Sigurdur Olafsson
Department of Industrial and Manufacturing Systems Engineering
Iowa State University
Ames, IA 50011

Abstract

We analyze a new optimization-based approach for feature selection that uses the nested partitions method for combinatorial optimization as a heuristic search procedure to identify near-optimal feature subsets. In particular, we show how to improve the performance of the nested partitions method using random sampling of instances. The new approach uses a two-stage sampling scheme that determines the required sample size to guarantee convergence to a near-optimal solution. This approach therefore has attractive theoretical characteristics. In particular, when the algorithm terminates in finite time, rigorous statements can be made concerning the quality of the final feature subset. Numerical results are reported to illustrate the key results, and show that the new approach is considerably faster than the original nested partitions method.

Key words: Feature selection, combinatorial optimization, metaheuristics

1 INTRODUCTION

Feature selection can be used to improve the simplicity of a data mining system, while maintaining acceptable accuracy for the learning algorithm to be used. It is also known that feature selection can improve the scalability of a data mining system as the learning process is usually faster with fewer features. In this paper, we are interested in improving the scalability of the feature selection process itself with respect to large number of instances. Our approach is based on an optimization-based feature selection method that uses the nested partitions (NP) metaheuristic [8], which has been shown to perform well when compared with other feature selection methods [5]. The NP method uses random search to explore the entire space of possible feature subsets, and is thus similar to methods such as genetic algorithms [12] and evolutionary search [7]. However, the search strategies themselves are quite different.

We show that using random sampling of instances can considerably reduce the computational time of the NP based feature selection algorithm. Since the random sampling may add noise to the evaluation of each feature subset, we propose using a two-stage variant of the algorithm that can be used to control this noise and is guaranteed to converge to a near-optimal feature subset in finite time. Using sampling of instances to improve scalability has been investigated intensely in the literature, and perhaps the most important, but yet difficult issue, is determining the appropriate sample size to maintain an acceptable accuracy. Some of the related research includes determining the sufficient sample sizes for finding association rules [9], progressive sampling methods [6], finding best sample sizes using a tuple relational calculus [3], and investigating the effect of class distribution on scalable learning [10].

2 NP-BASED FEATURE SELECTION

The feature selection problem involves identifying a subset A of the set $A^{(ALL)}$ of all n features that performs well given the training set T of m instances. The performance is measured according to some measure f, and the objective is to find the optimal subset $A^* \subseteq A^{(ALL)}$, where

$$f^* = f\left(A^*\right) = \min_{A \subseteq A^{(All)}} f(A).$$

The NP method uses *partitioning* to divide the space of all possible feature subsets into regions that can be analyzed individually and then aggregates the results from each region to determine how to continue the search, that is, how to concentrate the computational effort. In other words, the NP method adaptively takes *random samples of feature subsets* from the entire space of possible feature subsets and concentrates the sampling effort by systematic partitioning of this space. A key component in formulating the feature selection problem is selecting a performance measure. Depending on how this is done, feature selection methods may be divided into two categories: wrappers and filters. Wrappers use the accuracy of the resulting classification. Thus, to evaluate a subset of features, a predictive model is induced based on these features. Filters, on the other hand, select features before any other learning algorithm is applied. A different performance measure must therefore be specified. When choosing a wrapper or filter, the general consideration is that wrappers will give better performance when used with a supervised learning method, whereas filters are usually much faster. The NP optimization method can be implemented as either a wrapper or filter for feature selection [5]. Here, we focus on a filter employing the following correlation based measure [2]:

$$f_{correlation}(A) = \frac{k\overline{\rho}_{ca}}{\sqrt{k + k(k-1)\overline{\rho}_{aa}}}, \tag{1}$$

where k is the number of features in the set A, $\overline{\rho}_{ca}$ is the average correlation between the features in this set and the classification feature, and $\overline{\rho}_{aa}$ is the average correlation between features in the set A.

The NP method searches through the space of feature subsets by evaluating the entire subsets. On the other hand, it also incorporates methods that evaluate individual features into the partitioning to impose a structure that speeds the search. When it is done in such a way that good solution as clustered together in the same subsets, then those subsets are selected by the algorithm with relatively little effort. We now discuss an intelligent partitioning strategy when solving feature selection problems [5]. Given a current set $\mathsf{A}(k)$ of potential feature subsets, partition the set into two disjoint subsets

$$\mathsf{A}_1(k) = \{A \in \mathsf{A}(k) : a \in A\}, \tag{2}$$

$$\mathsf{A}_2(k) = \{A \in \mathsf{A}(k) : a \notin A\}. \tag{3}$$

The surrounding region is simply $\mathsf{A}_3(k) = \mathsf{A} \backslash \mathsf{A}(k)$. Each of these three regions is then sampled and based on these samples the next most promising region is selected. The selected region is partitioned into smaller subregions in the next iteration. If the surrounding region contains the best solution this is taken as an indication that the last move might not have been the best move, so the algorithm backtracks to what was the most promising region in the previous iteration. In theory, the features can be selected in an arbitrary order, but an intelligent partitioning where features are ordered according to their information gain performs significantly better, and this partitioning is used in all of the numerical experiments below.

This partitioning creates a tree of subsets that we refer to as the partitioning tree. The distance of the current promising region from the top of the tree, which corresponds to the minimum number of iterations it takes to get to this region, we refer to as the depth of the region. Once a maximum depth region is reached, that is a region that will not be partitioned further, the algorithm terminates. In the context of feature selection problem, this maximum depth will be equal to the number of features that are considered for either inclusion or exclusion from the selected set.

The key to the convergence of the NP method is the probability by which a region is selected correctly in each iteration. A sufficient condition for asymptotic convergence is that this probability of correct selection is bigger than one half, and to guarantee that a minimum probability is obtained, Olafsson [4] proposed using a two-stage sampling procedure that determines how much random sampling effort $N(\psi, \delta)$ is needed from each region to guarantee correct selection with probability ψ within an indifference zone $\delta > 0$. If this sampling effort is used, the probability of having found sufficiently good solution the first time maximum depth is reached, that is, when the search space has been reduced to a single feature subset, is bounded as follows:

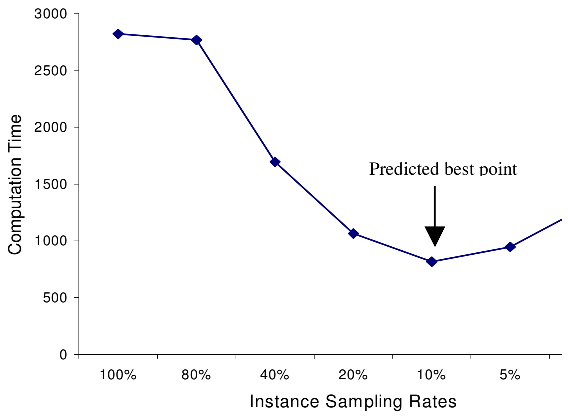
$$\Pr\left[\,| f(\mathsf{A}(k)) - f^* | \leq \delta\right] \geq \Psi, \tag{4}$$

where

$$\Psi = \frac{\psi^n}{(1-\psi)^n + \psi^n}. \tag{5}$$

Here ψ is the user selected minimum probability by which a correct selection is made in each iteration, and n is as before the total number of features.

We call the NP method applied to feature selection using the filter evaluation the NP-Filter, and if it also uses the two-stage sampling approach the Two-Stage NP-Filter (TSNP-Filter). A pseudo-code for the TSNP-Filter is shown in Appendix A, and used the following notation. We let N_j denote the number of sample sets in $\mathsf{A}_j(k)$, j = 1, 2, 3., j-th subregion in the k-th iteration and $X_{ij} = f(A_i^j)$, where A_i^j and $f(\cdot)$ are defined as the sample performance of i-th set in the j-th region. The two-stage ranking-and-



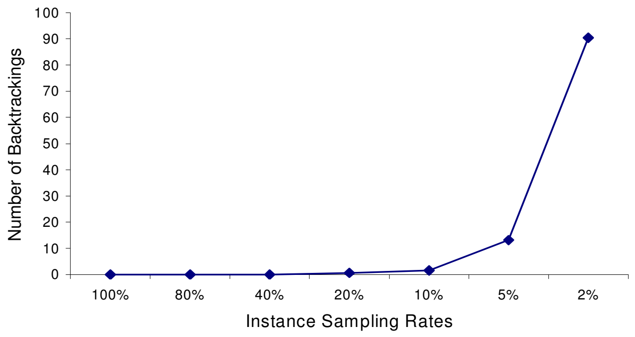


Figure 1. Computational time and number of backtrackings for sampling rates (data set 'vote').

selection procedure takes n_0 samples in the first stage, and then determines the total number N_j samples required from the j-th region using based on the sample variance of the performance estimates.

3 INSTANCE SAMLING IN THE NP-FILTER

In this section, we consider improving the scalability of the feature selection method in terms of its ability to handle increasing number of instances. The NP method was originally conceived for simulation-based optimization and is therefore naturally consistent with using performance estimates that are noisy due to sampling. Indeed, in the NP-Filter, a new set $A(k)$ of instances is sampled in each iteration in such a way that this set is independent of the previous sets: $A(0)$, $A(1)$, …, $A(k$ - 1). Thus, if the new instances indicate an erroneous decision has been made, the backtracking feature of the NP method enables the algorithm to make corrections, thus correcting the potential bias. The question still remains as of how large of a portion of the database is needed by the NP method. In particular, as the proportion is decreased and more backtracking is required. Then at some point the computational inefficiencies of backtracking will outweigh the savings obtained by using fewer instances. To evaluate these questions empirically, we apply the NP-Filter. We used four small data sets from the UCI repository of machine learning databases [1]. The characteristics of these data sets are shown in Table 1. As the NP-Filter is randomized algorithms, we run five replications for each experiment and report the average.

Table 1. Characteristics of the test datasets

Data Set	Instances	Features
Cancer	148	18
Vote	435	16
Audiology	226	69
kr-vs-kp	3196	36

Figure 1 illustrates the computation time needed by the NP-Filter for different sampling rates used. We note from the left hand graph of that figure that at first the computational decreases, but if the sampling rate becomes less than approximately 10% of the instances, the computational time actually increases. The intuitive explanation of this is shown in the right hand graph. It is clear that number of backtrackings abruptly increases when less than 10% of the instances are used. This means that even though each iteration may take less computation time, the number of iterations until maximum depth is reached increases dramatically, hence increasing the overall computation time. There is therefore some optimal sample rate R^*, where the NP-Filter would perform best.

To find this optimal rate, we must consider the cause of backtracking. The NP-Filter backtracks when it discovers that the surrounding region is actually more promising than the current most promising region. It thus corrects mistakes made due to noisy performance estimates by backtracking when the error is discovered, so we would expect to see more backtracking when fewer instances are used. In particular, when too few instances are used, the noise is excessive and backtracking must hence increase dramatically to compensate for the noise.

Table 2. Performance variances for sampling rates of instances.

Sample Rates	100%	80%	60%	40%	20%	10%	5%	2%
vote1	0.0	1.4	4.0	5.0	8.1	17.3	27.5	N/A
vote2	0.0	6.6	9.1	16.4	28.0	38.3	41.9	N/A
audiology1	0.0	1.5	4.2	6.1	16.9	33.4	48.8	94.3
audiology2	0.0	1.2	1.9	5.3	14.9	25.6	58.7	91.1
audiology3	0.0	0.9	2.4	4.9	10.8	28.7	58.2	185.4
cancer1	0.0	0.7	2.1	4.3	19.2	49.1	109.7	N/A
cancer2	0.0	0.5	1.8	4.5	14.7	52.7	104.7	N/A
cancer3	0.0	0.6	1.4	3.0	13.9	53.4	150.8	N/A
Kr-vs-kp1	0.0	0.2	0.4	0.9	2.8	5.9	8.9	14.9
Kr-vs-kp2	0.0	0.1	0.1	0.2	0.5	1.2	2.7	6.3
Kr-vs-kp3	0.0	0.1	0.1	0.2	0.5	1.2	2.7	9.5

In order to get a better feel for how the noise in the performance increases as a function of decreasing sample rate, we consider the datasets in Table 1. To calculate the true amount of noise, we must calculate the sample variance given all the feature subsets in a particular region, for all possible levels of the tree. Since the total number of non-empty feature subsets is 2^n-1, where n is the number of features, this quickly becomes infeasible, so instead of working with the datasets directly, we work with subsets of the dataset, each containing 7 randomly selected features. Even for such small datasets, 127 feature subsets must be evaluated for each experiment. The results are shown in Table 2.

From Table 2 it is clear that the performance variance increases rapidly as the instance sampling rate decreases. Indeed, all of the test datasets exhibit exponential pattern. We also note that although all of the datasets illustrate exponential growth, the rate is different for each dataset. We infer that the sample variance increases exponentially as the sample rate decreases, but that the rate of increase the exponential increase is application dependent and must be estimated from the data.

4 DETERMINING THE SAMPLING RATE FOR THE TSNP-FILTER

As for other methods that employ instance sampling to improve performance, finding the optimal sampling rate R^* is the biggest challenge when using the NP-Filter. As seen in Section 3, this sampling rate is related to the backtracking, which is in turn related to the variance of the performance. Intuitively, decreasing the sampling rate will always decrease the computation time for each iteration, but very small samples may cause a large variance of performances and cause excessive backtracking. This will in turn increase the number of iterations and eventually the overall computation time.

The trade-off is therefore between the computation time within each iteration, and the number of iterations needed until convergence is achieved. In the TSNP-Filter, the expected computation time within an iteration is a function of the performance variance, and the expected number of iterations is a function of the probability of selecting the correct region. Thus, analytical expression can be obtained for both these quantities, and the optimal trade-off achieved.

4.1 Formulation

In this section, we formulate the trade-off between minimizing the computation time within an iteration and minimizing the number of iterations for the TSNP-Filter as an optimization problem and find the optimal instance sampling rate R^* by solving this problem. We use the following notation. The total computation time is $T = T_1 + T_2 + \ldots + T_K$, where T_j is the computation time in the jth iteration and K is the total number of iterations. As before $N = N_j(k)$ denotes the number of sample feature sets at each iteration k, and I is the number of sample instances. The sampling rate is therefore given by $R = I / m$.

We are interested in minimizing the expected time of iteration k that can be affected by the performance variability and sampling rate of instance. We may therefore find a solution using the trade-off between $E[N_k \mid K]$ and $E[T_k \mid N_k]$. Since from the previous section it is known that the variability can be represented as the number of instances and $E[N_k \mid K]$ would increase as the number of instances decreases while $E[T_k \mid N_k]$ would decrease. Therefore, we formulate the trade-off as the following optimization problem.

$$\min \lambda \cdot E[N_k \mid K] + (1-\lambda) \cdot E[T_k \mid N_k] \qquad (6)$$

It is clear that as functions of the sampling rate R, $E[N_k \mid K]$ is decreasing and $E[T_k \mid N_k]$ is increasing. Furthermore, their scale can be different, so we weight them together using an weight λ that should be determined by the experimenter. From equation (10) in Appendix A, we know that $E[N_k \mid K]$ can be written in terms of the expected performance variance. However, the expected performance variance depends both on the application and the manner in which partitioning is done, so an analytical form cannot be obtain. The same is true for $E[T_k \mid N_k]$, but our empirical results strongly indicate certain patterns for these expected values, and we state those as assumptions.

Assumption 1. The expected calculation time of each feature sample is directly proportional to the number of instances.

Assumption 2. The relationship between performance variance and instance sampling rate is exponentially distributed. $E[S^2(k)] = c_1 e^{-c_2 \cdot R}$ for $c_1 > 0$, $c_2 > 0$.

As noted before, these is no theoretical justification for Assumptions 1-2, but they are both intuitively appealing and supported by our empirical results. Now, given those assumption, the optimal instances sampling rate is found in the following key theorem.

Theorem 1. Let Assumptions 1-2 hold. By using uniform sampling rate of instances and selecting the initial number of sample feature subsets sufficiently small so that it is smaller than then required number of samples, the optimal instance sampling rate is given by

$$R^* = -\frac{1}{c_2} \cdot \ln\left[\frac{(1-\lambda)\cdot c_0 \cdot \delta^2 \cdot m}{\lambda \cdot h^2 \cdot c_1 \cdot c_2}\right] \tag{7}$$

Proof. By Assumption 1, the expected computation time in each iteration, given the number of samples is directly proportional to the number of instances, that is,

$$E[T_k \mid N_k] = c_0 \cdot E[I] = c_0 \cdot I$$

From the fact that if the number of instance samples decreases, the performance variability of feature sample sets increases exponentially as stated in the previous section. The expected number of feature sample sets in each iteration can be stated as follows:

$$E[N_k \mid K] = \frac{h^2}{\delta^2} E\left[S^2(k)\right]$$

Based on equation (6) and the assumptions, the restated problem is as follows.

$$\min_R \quad \lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda)\cdot c_0 \cdot m \cdot R$$

$$\text{subject to} \quad 0 < \lambda < 1$$
$$0 < R \le 1.$$

This problem can be solved by taking the derivative of the objective function and identifying the minimum point that satisfies the constraints. In particular, since

$$\frac{d^2 Cost}{dR^2} = \frac{\lambda \cdot h^2}{\delta^2} \cdot c_1 \cdot {c_2}^2 \cdot e^{-c_2 \cdot R} > 0,$$

any solution to

$$0 = \frac{d}{dR}\left[\lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda)\cdot c_0 \cdot m \cdot R\right]$$
$$= -Rc\lambda \cdot \frac{h^2}{\delta^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda)\cdot c_0 \cdot m$$

is a minimum. It follows that R^* is given by equation (7). □

The value of λ can be chosen by users according to their preference. The indifference zone, δ and selection probability, P^* that determines the value of h should be also determined by user preferences. If δ is small and P^* is large, the sampling rate would be large and vice versa.

4.2 Numerical Results

The NP method guarantees a correct selection with probability ψ within an indifference zone $\delta > 0$. However, since the TSNP-Filter incorporates a heuristic approach, the robustness of this must be evaluated empirically. The constants c_0, c_1, and c_2 are calculated empirically for each of the datasets in Table 1, and R^* calculated according to equation (12). To evaluate if the TSNP-Filter solution is within the indifference zone, the true optimum must be known. Since this is computationally intractable except for small datasets, we again use modified data sets that now contain 8 randomly selected features in addition to the class feature. First we find the optimal solution for each dataset using an enumerative approach and

Table 3. Accuracy of the TSNP-Filter on the reduced datasets.

		ψ					
		0.75		0.85		0.95	
Dataset	δ	Sampling Rate (%)	Accuracy	Sampling Rate (%)	Accuracy	Sampling Rate (%)	Accuracy
Vote	5	16	95.57±0.27	21	95.57±0.26	23	95.58±0.25
	1	32	95.59±0.25	37	95.58±0.26	39	95.61±0.11
Audiology	5	27	42.62±0.03	30	43.61±0.02	35	43.63±0.01
	1	44	43.63±0.02	47	43.63±0.02	51	43.64±0.02
Cancer	5	24	70.32±0.24	28	70.34±0.11	31	70.35±0.06
	1	40	70.35±0.17	45	70.34±0.09	47	70.36±0.03
Kr-vs-kp	5	8	65.55±0.48	13	65.57±0.32	15	65.55±0.41
	1	25	65.51±0.21	29	65.59±0.15	32	65.61±0.07

then calculate how many solutions of the TSNP - Filter out of 100 replications are within the indifference zone δ . For the other parameters, we set $\lambda = 0.5$, δ = 1 and 5 percentage points, and ψ = 0.75, 0.85 and 0.95. The results are reported in Table 3.

From Table 3, we note that as expected the sample rate is smaller for δ = 5 than δ = 1, and smaller when ψ is smaller. Thus, by adjusting the instance sampling rate appropriately, the quality of the solutions found by the TSNP-Filter remains constant. This is supported by the results in Table 3, which shows that for each problem there is no significant difference in the accuracy obtained. The percentage of time that this accuracy is within the indifference zone is reported in Table 4. We note that for indifference zone of δ = 5, the estimated probability of being within the indifference zone is actually significantly higher that the minimum probability ψ of correct selection. The intuitive explanation for this is that when the indifference zone is selected this large then it is relatively easy to find feature subsets with accuracy within the indifference zone, and hence this will happen most of the time, even if ψ = 0.75 is selected. When the indifference zone is smaller, δ = 5, then the estimated probabilities closely follow the prescribed minimum ψ , but except for the 'vote' dataset, the minimum is not met exactly.

Table 4. Probabilities that a solution is within the indifference zone.

Dataset	δ	ψ		
		0.75	***0.85***	***0.95***
Vote	5	0.96	0.98	0.98
	1	0.78	0.88	0.96
Audiology	5	0.98	0.98	1.00
	1	0.72	0.83	0.89
Cancer	5	0.83	0.88	0.97
	1	0.65	0.72	0.81
Kr-vs-kp	5	0.90	0.94	0.95
	1	0.63	0.74	0.87

The results reported above provide some insights into how the TSNP-Filter works. However, we are primarily interested in how the new two-stage sampling approach improves the performance of the NP-Filter. We thus make a three-fold comparison between the TSNP-Filter, the original NP-Filter, and the NP-Filter with a constant sampling rate found by experiments with sampling rates of R = {100,80,60,40,20,10,5,2} and selecting the best rate. The results are reported in Table 5.

Table 5. Comparison of three different scalability methods.

Dataset	***Approach***	***Sample Rate***	***Accuracy***	***Speed***	***Backtracks***
Vote	TSNP-Filter	16	93.2±1.3	786±113	0.2±0.4
	NP-Filter w/sampling	10	92.4±1.0	816±167	1.6±2.2
	NP-Filter	100	93.5±0.4	2820±93	0.0±0.0
Audilogy	TSNP-Filter	27	70.2±1.6	27722±6804	128.8±24.8
	NP-Filter w/sampling	10	69.2±2.4	35839±14563	371.0±182.0
	NP-Filter	100	69.7±1.9	41105±3255	0.0±0.0
Cancer	TSNP-Filter	24	73.5±0.5	418±10	2.4±2.8
	NP-Filter w/sampling	10	72.6±1.2	486±89	7.4±3.4
	NP-Filter	100	73.2±0.6	795±83	0.0±0.0
Kr-vs-kp	TSNP-Filter	3	89.0±0.4	5189±492	0.0±0.0
	NP-Filter w/sampling	5	89.0±1.2	7246±809	1.8±3.0
	NP-Filter	100	87.9±5.7	107467±8287	1.8±3.0

As indicated by Table 5, the TSNP-Filter generally provides the better performance in terms of computation time for four data sets without sacrificing accuracy. On the other hand, original NP-Filter shows the worse performance. A more interesting result is that the TSNP-Filter even performs better than the NP-Filter with sampling where the sampling rate is determined experimentally as the best sample rate. The intuitive reason for this is that this approach uses the same sample rate in every iteration without consideration of the size of the regions begin compared in that iteration. This means that it tends to oversample in certain situations when the decision is relatively easy. The TSNP-Filter, on the other hand, automatically determines the best sampling rate and does this very effectively.

5 CONCLUSION

The NP method for combinatorial optimization has previously been shown to be an effective approach for feature selection, and compare favorably to other methods [5]. In this paper, we have shown that by using random sampling of instances, the speed of the NP-based feature selection method can be improved significantly. The key issue in using sampling is to determine the sample size. For the NP-based approach, using too small of a sample rate causes too much noise in the performance evaluation that causes the algorithm to make incorrect moves that must be corrected through backtracking. Hence, the number of iterations increases and the overall computation time does as well. The optimal sampling rate will depend on both the size and structure of the particular dataset, so it cannot be easily determined a priori. However, we proposed a two-stage sampling approach that determines the necessary sampling effort based on the estimated variance. The numerical results reported show that sampling works well in general, and that the two-stage approach finds very good sample rates in an automated manner.

REFERENCES

[1] Blake, C.L. and Merz, C.J., 1998, *UCI Repository of machine learning databases* <http://www.ics.uci.edu/mlearn/MLRepository.html>, University of California, Irvine, CA (Date Accessed: October 31, 2003).

[2] Hall, M.A., 1998, "Correlation-based feature selection for discrete and numeric class machine learning", in *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford University, CA. Morgan Kaufmann.

[3] Kivinen, J. and Mannila, H., 1994, "The power of sampling in knowledge discovery", in *ACM Symposium on Principles of Database Theory*, 77-85.

[4] Olafsson, S., 2004, "Two-stage nested partitions method for stochastic optimization," *Methodology and Computing in Applied Probability*, 6, 5-27.

[5] Olafsson, S. and Yang, J., 2005, "Intelligent partitioning for feature selection", *INFORMS Journal on Computing*, in print.

[6] Provost, F., Jensen, D. and Oates, T., 1999, "Efficient progressive sampling", in *Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining*, 23-32.

[7] Shi, L. and Olafsson, S., 2000, "Nested partitions method for global optimization", *Operations Research*, 48, 390-407.

[8] Toivonen, H., 1996, "Sampling large databases for association rules", in *Proceedings of the 22nd International Conference on Very Large Databases*, 134-145.

[9] Weiss, G. M. and Provost, F., 2001, "The effect of class distribution on classifier learning: an empirical study", Technical Report ML-TR-44, Department of Computer Science, Rutgers University August 2, 2001.

[10] Yang, J. and Honavar, V., 1998, "Feature subset selection using a genetic algorithm" In H. Motada and H. Liu (eds), *Feature Selection, Construction, and Subset Selection: A Data Mining Perspective*, Kluwer, New York.

APPENDIX A: TSNP-FILTER

```
Given $K > 1$, $n_0$, $d_{stop}(n)$, $\delta$, $\Psi$ and an order $a_{[1]}$, $a_{[2]}$, …, $a_{[n]}$ of features
Initialize $\mathrm{A}(0) \longleftarrow A$, $k \longleftarrow 0$, $A^* = \{\}$ and $f^* = \infty$
loop
    $A_1(k) \longleftarrow \{A \in \mathrm{A}(k) : a_{d(k)} \in A\}$,
    $A_2(k) \longleftarrow \{A \in \mathrm{A}(k) : a_{d(k)} \notin A\}$,
    $A_3(k) \longleftarrow A \setminus A(k)$,
    for every set $A_j(k)$
        $A^j_{best}(k) \longleftarrow \{\}$,
        $f^j_{best}(k) \longleftarrow \infty$,
        $i \leftarrow 1$
        Obtain $n_0$ sample sets
        Calculate the first-stage sample means and variance for $j$=1,2,3:
        $\bar{X}^{(1)}_j(k) \leftarrow \frac{1}{n_0}\sum_{i=1}^{n_0} X_{ij}(k)$,      (8)
        $S^2_j(k) \leftarrow \frac{\sum_{i=1}^{n_0}\left[X_{ij}(k) - X^{(1)}_j(k)\right]^2}{n_0 - 1}$      (9)
        Compute the total sample size
        $N_j(k) \leftarrow \max\left\{n_0 + 1, \left\lceil \frac{h^2 S^2_j(k)}{\delta^2} \right\rceil\right\}$      (10)
        where $\delta$ is the indifference zone and $h$ is a constant that is determined by $n_0$ and the minimum selection probability $P^*$ of correct selection [7].
        Obtain $N_j(k) - n_0$ more samples in each region
        loop
          $A_{ji}(k) \longleftarrow$ Randomly select a feature subset
          if $f_{ji}(k) < f^j_{best}(k)$ then
              $f^j_{best}(k) \longleftarrow f_{ji}(k)$,
              $A^j_{best}(k) \longleftarrow A_{ji}(k)$
          $i \leftarrow i + 1$
        until enough feature subset samples
        $j^* \leftarrow \arg\min_j f^j_{best}(k)$
        if $j^* = 3$ then $\mathrm{A}(k+1) \longleftarrow \mathrm{A}(k-1)$
        else $\mathrm{A}(k+1) \longleftarrow \mathrm{A}_{j^*}(k)$
        $k \leftarrow k + 1$
        end
until $d(\mathrm{A}(k)) = d_{stop}(n)$
```

Boosted Lasso [*]

Peng Zhao[†] Bin Yu[‡]

Abstract

In this paper, we propose a Boosted Lasso (BLasso) algorithm which is able to produce the complete regularization path for general Lasso problems. BLasso works in a similar fashion like Boosting and Forward Stagewise Fitting with an additional "backward" step which works by shrinking the model complexity of an ensemble learner. Both theoretical and experimental results are shown for the BLasso algorithm. In addition, we generalize BLasso to deal with problems with general convex loss with general convex penalty.

Keywords: Regularization Path; Boosting; Lasso; Backward Step; Steepest Descent

1 Introduction

An important idea that recently comes from the statistics community is the Lasso [16]. Lasso is a shrinkage method that regularizes fitted models using a L_1 penalty. Its popularity can be explained in several ways. Since nonparametric models that fit training data well often have low bias but large variances, prediction accuracy can sometimes be improved by shrinking a model or making a model more sparse. The regularization resulting from the L_1 penalty leads to sparse solutions where there are few basis functions with nonzero weights (among all possible choices). This Statement is proved rigorously in recent works [4] in the specialized setting of over-complete representation and large underdetermined systems of linear equations. Furthermore, the sparse models induced by Lasso are more interpretable and often preferred in areas such as Biostatistic and Social Sciences.

Another vastly popular idea, Boosting, is a method for iteratively building an additive model. Since its inception in 1990 [6] [7] [15], it has become one of the most successful machine learning ideas.

While it is a natural idea to combine boosting and Lasso to have a regularized boosting procedure, it is also intriguing that boosting, without any additional regularization, has its own resistance to overfitting. For specific cases, e.g. L_2Boost [9], this resistance is understood to some extent [3]. However, it is not until later when Forward Stagewise Fitting (FSF) was introduced and connected with a boosting procedure with much more cautious steps that a shocking similarity between FSF and Lasso was observed [11] [5].

This link between Lasso and FSF is formally described in linear regression case through LARS (Least Angle Regression, [5]). It is also known that for special cases (e.g. orthogonal designs) FSF can approximate Lasso path infinitely close, but in general, they are different from each other despite of their similarity. However, FSF is still used as an approximation to Lasso for different regularization parameters because it is computationally prohibitive to solve Lasso for many regularization parameters.

In this paper, we propose a new algorithm **Boosted Lasso** (**BLasso**) that approximates the Lasso path in all cases. The motivation comes from a critical observation that both Forward Stagewise Fitting and Boosting work in a forward fashion (so is Forward Stagewise Fitting named). The model complexity, measured by the L_1 norm of model parameters, holds a dominating upward trend for both methods. This is often proven too greedy – the algorithms are not able to correct mistakes made in early stages. We introduce an innovative "backward" step which utilizes the same minimization rule as the forward step to define each fitting stage but utilizes an additional rule to force the model complexity to decrease. As a combination of backward and forward step, Boosted Lasso is able to go back and forth and tracks the Lasso path correctly.

BLasso has the same order of computational complexity as FSF. But unlike FSF, BLasso can be proven to converge to the Lasso solutions as step size of the algorithm goes to zero. The fact that BLasso can also be generalized to give regularized path for other penalized loss functions with general convex penalties also comes as a pleasant surprise.

After a brief overview of Boosting, Forward Stagewise Fitting in Section 2.1 and the Lasso in Section 2.2, Section 3 introduces BLasso and its properties. Section 4 discusses the backward step which gives the intuition behind BLasso and explains how FSF fails to give the

[*]This research is partially supported by NSF grants CCR-0106656, FD01-12731 and ARO grant DAAD19-01-1-0643. Yu was also partially supported by a Miller Ressearch Professorship from Miller Institue at UC Berkeley in spring, 2004.

[†]University of California, Berkeley

[‡]University of California, Berkeley

Lasso path. Section 5 covers the least square problem in details as an example for BLasso. In section 6, we support the theory and algorithms by experiments using simulated and real data sets which demonstrate the attractiveness of Boosted Lasso. Finally, Section 7 contains a discussion on choice of step sizes and application of BLasso in online learning with a summary of the paper.

2 Boosting, Forward Stagewise Fitting and the Lasso

Boosting utilizes an iterative fitting procedure that builds up model stage by stage. Forward Stagewise Fitting uses more fitting stages by limiting the step size at each stage to a small fixed constant and produces solutions that are strikingly similar to the Lasso. We first give a brief overview of these two algorithms followed by an overview of the Lasso.

2.1 Boosting and Forward Stagewise Fitting

The boosting algorithms can be seen as functional gradient descent techniques. The task is to estimate the function $F : R^d \to R$ that minimizes an expected loss

$$E[C(Y, F(X))], \quad C(\cdot,\cdot) : R \times R \to R^+ \tag{2.1}$$

based on data $Z_i = (Y_i, X_i)(i = 1, ..., n)$. The univariate Y can be continuous (regression problem) or discrete (classification problem). The most prominent examples for the loss function $C(\cdot,\cdot)$ include Classification Margin, Logit Loss and L_2 Loss functions.

The family of $F(\cdot)$ being considered is the set of ensembles of "base learners"

$$D = \{F : F(x) = \sum_{j=1}^{m} \beta_j h_j(x), x \in R^d, \beta_j \in R\}. \tag{2.2}$$

Let $\beta = (\beta_1, ...\beta_m)^T$, we can reparametrize the problem using

$$L(Z, \beta) := C(Y, F(X)), \tag{2.3}$$

where the specification of F is hidden by L which makes our notation simpler.

The parameter estimate $\hat{\beta}$ can be found by minimizing the empirical loss

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} L(Z_i; \beta). \tag{2.4}$$

Despite the fact that the empirical loss function is often convex in β, this is usually a formidable optimization problem for a moderately rich function family, and we often settle for approximating suboptimal solutions by a progressive procedure that iteratively builds up the solution:

$$(\hat{j}, \hat{g}) = \arg\min_{j,g} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + g1_j) \tag{2.5}$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{g}1_{\hat{j}} \tag{2.6}$$

where 1_j is the jth standard basis for R^m and $g \in R$ is a stepsize parameter, i.e. the vector with all 0s except for a 1 in the jth coordinate.

FSF is a similar method for approximating the minimization problem described by (2.5) with additional regularization. It disregards the stepsize g in (2.6) and instead update $\hat{\beta}^t$ by a fixed stepsize ϵ:

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \epsilon \cdot \text{sign}(g)1_{\hat{j}}$$

When FSF was introduced [11] [5], it was only described for the L_2 regression setting. For general loss functions, it can be defined by removing the minimization over g in (2.5):

$$(\hat{j}, \hat{s}) = \arg\min_{j,s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \tag{2.7}$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}1_{\hat{j}}, \tag{2.8}$$

Notice that this is only a change of form, underlying mechanic of the algorithm remains unchanged in the L_2 regression setting from [5] as can be seen later in Section 5. Initially all coefficients are zero. At each successive step, a coefficient is selected that best fits the empirical loss. Its corresponding coefficient $\beta_{\hat{j}}$ is then incremented or decremented by a small amount, while all other coefficients β_j, $j \neq \hat{j}$ are left unchanged.

By taking small steps, Forward Stagewise Fitting imposes some implicit regularization. After applying it with $T < \infty$ iterations, many of the coefficients will be zero, namely those that have yet to be incremented. The others will tend to have absolute values smaller than the unregularized solutions. This shrinkage and sparsity property is observed in the striking similarity between the solutions given by Forward Stagewise Fitting and the Lasso which we give a brief overview next.

2.2 Lasso Let $T(\beta)$ denote the L_1 penalty of $\beta = (\beta_1, ..., \beta_m)^T$,

$$T(\beta) = \|\beta\|_1 = \sum_{i=1}^{m} |\beta_i|$$

and let $\Gamma(\beta; \lambda)$ denote the Lasso loss function

$$\Gamma(\beta; \lambda) = \sum_{i=1}^{n} L(Z_i; \beta) + \lambda T(\beta). \tag{2.9}$$

The Lasso estimate $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_m)^T$ is defined by

$$\hat{\beta} = \min_{\beta} \Gamma(\beta; \lambda)$$

The parameter $\lambda \geq 0$ controls the amount of regularization applied to the estimate. Setting $\lambda = 0$ reverses the Lasso problem to minimizing unregularized empirical loss. On the other hand, a very large λ will completely shrink $\hat{\beta}$ to 0 thus leads to an empty model. In general, moderate values of λ will cause shrinkage of the solutions towards 0, and some coefficients may be exactly equal to 0. This sparsity in Lasso solutions has been studied extensively, e.g. [4].

Computation of the solution of the Lasso problem for a fixed λ has been studied for special cases. Specifically, for least square regression, it is a quadratic programming problem with linear inequality constraints; for 1-norm SVM, it can be transformed into a linear programming problem. But to get a good fitted model that performs well on future data, we need to select an appropriate value for the tuning parameter λ. Practical algorithms have been proposed for square loss function (LARS, [5]) and SVM (1-norm SVM, [17]) to give the entire regularization path.

But how to give the entire regularization path of the Lasso problem for general convex loss function remained open. Next, we propose a Boosted Lasso (BLasso) algorithm which works in a computationally efficient fashion as FSF and is able to approximate the Lasso path infinitely close.

3 Boosted Lasso

We first describe the algorithm.

Boosted Lasso (BLasso)

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a small stepsize constant $\epsilon > 0$, take an initial forward step

$$\begin{aligned} (\hat{j}, \hat{s}_{\hat{j}}) &= \arg \min_{j, s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; s1_j), \\ \hat{\beta}^0 &= \hat{s}_{\hat{j}} 1_{\hat{j}}, \end{aligned}$$

Then calculate the initial regularization parameter

$$\lambda^0 = \frac{1}{\epsilon}\left(\sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^0)\right)$$

.

Set the active index set $I_A^0 = \{\hat{j}\}$. Set $t = 0$.

Step 2 (Backward and Forward steps). Find the "backward" step that leads to the minimal empirical loss

$$\hat{j} = \arg \min_{j \in I_A^t} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s_j 1_j) \quad \text{where } s_j = -\text{sign}(\hat{\beta}_j^t)\epsilon.$$

Take the step if it leads to a decrease in the Lasso loss, otherwise force a forward step and relax λ if necessary:

If $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) < \Gamma(\hat{\beta}^t, \lambda^t)$, then

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \ \lambda^{t+1} = \lambda^t.$$

Otherwise,

$$\begin{aligned} (\hat{j}, \hat{s}) &= \arg \min_{j, s=\pm\epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \hat{s} 1_{\hat{j}}, \\ \lambda^{t+1} &= \min[\lambda^t, \frac{1}{\epsilon}(\sum_{i=1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i=1}^{n} L(Z_i; \hat{\beta}^{t+1}))], \\ I_A^{t+1} &= I_A^t \cup \{\hat{j}\}. \end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \leq 0$.

We defer formal definition of forward and backward steps till the next section. Immediately BLasso has the following properties:

Lemma 3.1. *The following statements hold:*

1. *For $\forall\lambda$ s.t. $\exists j, |s| = \epsilon$ $\Gamma(s1_j; \lambda) \leq \Gamma(0; \lambda)$, we have $\lambda^0 \geq \lambda$.*
2. *For $\forall t$ where $\lambda^{t+1} = \lambda^t$, we have $\Gamma(\hat{\beta}^{t+1}; \lambda^t) \leq \Gamma(\hat{\beta}^t; \lambda^t)$.*
3. *For $\forall t$ where $\lambda^{t+1} < \lambda^t$, we have $\Gamma(\hat{\beta}^t; \lambda^t) < \Gamma(\hat{\beta}^t \pm \epsilon 1_j; \lambda^t)$, $\forall j$ and $\|\hat{\beta}^{t+1}\|_1 = \|\hat{\beta}^t\|_1 + \epsilon$.*

According to the Lemma, Boosted Lasso starts with an initial λ_0 which is the largest λ that would allow an ϵ step away from 0. For each value of λ, BLasso performs coordinate descent until there is no descent step. Then the value of λ is reduced and a forward step is forced. Since the minimizers correspond to adjacent λs are usually close, this procedure proceeds from one solution to the next within a few steps and effectively approximates the Lasso path. In general, we have the following result:

Theorem 3.1. *If $L(Z; \beta)$ is strictly convex and continuously differentiable in β, then as $\epsilon \to 0$, the BLasso path converges to the Lasso path.*

Many popular loss functions, e.g. L^2, logistic and likelihood functions of exponential family, are convex and continuously differentiable. Other functions like the hinge loss (SVM) is continuous and convex but not strictly convex or differentiable. It is theoretically possible that BLasso's coordinate descent strategy gets stuck at nonstationary points for these functions. However, as we illustrate in the second experiment, BLasso works well for 1-norm SVM problem empirically.

4 the Backward Boosting Step

We now explain the motivation and working mechanic of BLasso. One observation is that FSF uses only "forward" steps. It only takes steps that lead to direct reduction of the empirical loss. Comparing to classical model selection methods like Forward Selection and Backward Elimination, Growing and Pruning of a classification tree, a "backward" counterpart is missing. Without the backward step, FSF can be too greedy and does not reproduce the Lasso path in general. For a given $\beta \neq 0$ and $\lambda > 0$, consider the impact of a small $\epsilon > 0$ change of β_j to the Lasso loss $\Gamma(\beta;\lambda)$. For an $|s| = \epsilon$,

$$\begin{aligned} \Delta_j \Gamma &= (\sum_{i=1}^n L(Z_i;\beta + s1_j) - \sum_{i=1}^n L(Z_i;\beta)) \\ &+ \lambda(T(\beta + s1_j) - T(\beta)) \\ &:= \Delta_j(\sum_{i=1}^n L(Z_i;\beta)) + \lambda \Delta_j T(\beta). \end{aligned} \tag{4.10}$$

Since $T(\beta)$ is simply the L_1 norm of β, $\Delta T(\beta)$ reduces to a simple form:

$$\begin{aligned} \Delta_j T(\beta) &= \|\beta + s1_j\|_1 - \|\beta\|_1 \\ &= |\beta_j + s| - |\beta_j| \\ &= \text{sign}^+(\beta_j, s) \cdot \epsilon \end{aligned} \tag{4.11}$$

where $\text{sign}^+(\beta_j, s) = 1$ if $s\beta_j > 0$ or $\beta_j = 0$, $\text{sign}^+(\beta_j, s) = -1$ if $s\beta_j < 0$ and $\text{sign}^+(\beta_j, s) = 0$ if $s = 0$.

Equation (4.11) shows that an ϵ step's impact on penalty is a fixed ϵ for different j. Only the sign of the impact may vary. Suppose given a β, the "forward" steps for different j have impacts on the penalty of the same sign, then $\Delta_j T$ is a constant in (4.10) for all j. Thus, minimizing the Lasso loss using fixed-size steps is equivalent to minimizing the empirical loss directly. At the early stages of Forward Stagewise Fitting, all forward steps are parting from zero, therefore all the signs of the "forward" steps' impact on penalty are positive. As the algorithm proceeds into later stages, some of the signs may change into negative and minimizing the empirical loss is no longer equivalent to minimizing the Lasso loss. Thus, in the beginning, Forward Stagewise Fitting carries out a steepest descent algorithm that minimizes the Lasso loss and follows Lasso's regularization path, but as it goes into later stages, the equivalence is broken and they part ways.

In fact, except for special cases like orthogonal designed covariates, the signs of the forward steps' impacts on penalty can change from positive to negative. These steps then reduce the empirical loss and penalty simultaneously therefore they should be preferred over other forward steps. Moreover, there can also be occasions where a step goes "backward" to reduce the penalty with a small sacrifice in empirical loss. In general, to minimize the Lasso loss, one need to go "back and forth" to trade-off the penalty with empirical loss basing on different regularization parameters. We call a direction that leads to reduction of the penalty a "backward" direction and define a backward step as the following:

For a given $\hat{\beta}$, a **backward step** is such that:

$$\Delta\hat{\beta} = s_j 1_j,$$

for some j, subject to $\hat{\beta}_j \neq 0$, $\text{sign}(s) = -\text{sign}(\hat{\beta}_j)$ and $|s| = \epsilon$. Making such a step will reduce the penalty by a fixed amount $\lambda \cdot \epsilon$, but its impact on the empirical loss may vary, therefore we also want:

$$\hat{j} = \arg\min_j \sum_{i=1}^n L(Z_i; \hat{\beta} + s_j 1_j)$$

$$\text{subject to } \hat{\beta}_j \neq 0 \text{ and } s_j = -\text{sign}(\hat{\beta}_j)\epsilon,$$

i.e. $\hat{j}$ is picked such that the empirical loss after making the step is as small as possible.

While forward steps try to reduce the Lasso loss through minimizing the empirical loss, the backward steps try to reduce the Lasso loss through minimizing the Lasso penalty. During a fitting process, although rarely happen, it is possible to have a step reduce both the empirical loss and the Lasso penalty thus it is both forward and backward. We do not distinguish such steps as they do not create any confusions.

By identifying the backward steps, we are able to work with the penalized Lasso loss directly and take backward steps to correct previous steps that are seen too greedy in later stages. This new concept both motivated the Boosted Lasso algorithm and is the underlying mechanic that BLasso utilizes to follow the Lasso path.

5 Least Square Problem

For the most common special case – least square regression, the forward steps, backward steps and BLasso all become simpler and more intuitive. To see this, we write out the empirical loss function $L(Z_i;\beta)$ in its L_2 form,

$$\sum_{i=1}^n L(Z_i;\beta) = \sum_{i=1}^n (Y_i - X_i\beta)^2 = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^n \eta_i^2.$$

where $\hat{Y} = (\hat{Y}_1, ..., \hat{Y}_n)^T$ are the "fitted values" and $\eta = (\eta_1, ..., \eta_n)^T$ are the "residuals".

Recall that in a penalized regression setup $X_i = (X_{i1}, ..., X_{im})$ where every covariates $X^j = (X_{1j}, ..., X_{nj})^T$ is normalized, i.e. $\|X^j\|^2 = \sum_{i=1}^n X_{ij}^2 = 1$ and $\sum_{i=1}^n X_{ij} = 0$. For a given $\beta = (\beta_1, ...\beta_m)^T$, the impact of a step s of size $|s| = \epsilon$ along β_j on the empirical loss function can be written as:

$$\begin{aligned} &\Delta(\sum_{i=1}^n L(Z_i;\beta)) \\ = &\sum_{i=1}^n [(Y_i - X_i(\beta + s1_j))^2 - (Y_i - X_i\beta)^2] \\ = &\sum_{i=1}^n [(\eta_i - sX_i 1_j)^2 - \eta_i^2] \\ = &\sum_{i=1}^n (-2s\eta_i X_{ij} + s^2 X_{ij}^2) \\ = &-2s(\eta \cdot X^j) + s^2. \end{aligned}$$

The last line of these equations delivers a strong message – in least square regression, given the step size, the impact on the empirical loss function is solely determined by the correlation between the fitted residuals and the coordinate. Specifically, it is proportional to the negative correlation between the fitted residuals and the covariate plus the step size squared. Therefore, steepest descent with a fixed step size on the empirical loss function is equivalent to finding the covariate that has the maximum size of correlation with the fitted residuals, then proceed along the same direction. This is in principle same as Forward Stagewise Fitting.

Translate this for the forward step where originally

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^n L(Z_i; \beta + s1_j),$$

we get

$$\hat{j} = \arg\max_j |\eta \cdot X^j| \quad \text{and} \quad \hat{s} = \text{sign}(\eta \cdot X^{\hat{j}})\epsilon,$$

which coincides exactly with the stagewise procedure described in [5] and is in general the same principle as L_2 Boosting, i.e. recursively refitting the regression residuals along the most correlated direction except the difference in step size choice [9] [3]. Also, under this simplification, a backward step becomes

$$\hat{j} = \arg\min_j(-s(\eta \cdot X^j))$$

subject to $\hat{\beta}_j \neq 0$ and $s_j = -\text{sign}(\hat{\beta}_j)\epsilon$.

Ultimately, since both forward and back steps are based solely on the correlations between fitted residuals and the covariates, therefore in the L_2 case, BLasso reduces to finding the best directions in both forward and backward directions by examining the correlations, then decide whether to go forward or backward based on the regularization parameter.

6 Generalized Boosted Lasso

As stated earlier, BLasso not only works for general convex loss functions, it can also be generalized for convex penalties other than L_1 penalty. For the Lasso problem, BLasso algorithm does a fixed step size coordinate descent to minimize the penalized loss. Since the penalty has the special L_1 norm and (4.11) holds, therefore the coordinate descent takes form of "backward" and "forward" steps. For general convex penalties, this nice feature is lost but the algorithm still works.

Assume $T(\beta)$: $R^m \to R$ is a penalty function and is convex in β, now we describe the Generalized Boosted Lasso algorithm:

Generalized Boosted Lasso

Step 1 (initialization). Given data $Z_i = (Y_i, X_i)$, $i = 1, ..., n$ and a small stepsize constant $\epsilon > 0$, take an initial forward step

$$\begin{aligned} (\hat{j}, \hat{s}_{\hat{j}}) &= \arg\min_{j, s=\pm\epsilon} \sum_{i=1}^n L(Z_i; s1_j), \\ \hat{\beta}^0 &= \hat{s}_{\hat{j}} 1_{\hat{j}}. \end{aligned}$$

Then calculate the corresponding regularization parameter

$$\lambda^0 = \frac{\sum_{i=1}^n L(Z_i; 0) - \sum_{i=1}^n L(Z_i; \hat{\beta}^0)}{T(\hat{\beta}^0) - T(0)}.$$

Set $t = 0$.

Step 2 (steepest descent on Lasso loss). Find the steepest coordinate descent direction on Lasso loss

$$(\hat{j}, \hat{s}_{\hat{j}}) = \arg\min_{j, s=\pm\epsilon} \Gamma(\hat{\beta}^t + s1_j; \lambda^t).$$

Update $\hat{\beta}$ if it reduces Lasso loss; otherwise force $\hat{\beta}$ to minimize the empirical loss and recalculate the regularization parameter :

If $\Gamma(\hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}; \lambda^t) < \Gamma(\hat{\beta}^t, \lambda^t)$, then

$$\hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}_{\hat{j}} 1_{\hat{j}}, \; \lambda^{t+1} = \lambda^t.$$

Otherwise,

$$\begin{aligned}\hat{j} &= \arg\min_j \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + \text{sign}(\hat{\beta}^t_j)\cdot \epsilon 1_j),\\ \hat{\beta}^{t+1} &= \hat{\beta}^t + \text{sign}(\hat{\beta}^t_{\hat{j}}) 1_{\hat{j}},\\ \lambda^{t+1} &= \min[\lambda^t, \frac{\sum_{i=1}^n L(Z_i;\hat{\beta}^t) - \sum_{i=1}^n L(Z_i;\hat{\beta}^{t+1})}{T(\hat{\beta}^{t+1}) - T(\hat{\beta}^t)}].\end{aligned}$$

Step 3 (iteration). Increase t by one and repeat Step 2 and 3. Stop when $\lambda^t \le 0$.

In the Generalized Boosted Lasso algorithm, explicit "forward" or "backward" steps are no longer seen. However, the mechanic remains the same – minimize the penalized loss function for each λ, relax the regularization by reducing λ when the minimal is reached.

Another algorithm of a similar fashion is developed independently in [14]. There, starting from $\lambda = 0$, a solution is generated by taking a small Newton-Raphson step for each λ, then λ is increased by a fixed amount. The algorithm assumes twice-differentiability of both loss function and penalty function and involves calculation of the Hessian matrix. A step size parameter is used for increasing the λ.

In comparison, BLasso only assumes convexity of the functions and uses much simpler and computationally less intensive operations for each λ. The stepsize is defined in the original parameter space which makes the solutions evenly spreaded in parameter space rather than in λ. In fact, since λ is approximately the reciprocal of size of the penalty, as fitted model grows larger and penalty becomes bigger, changing λ by a fixed amount makes the algorithm in [14] stepping too fast in the parameter space. On the other hand, when the model is close to empty and the penalty function is very small, λ is very large, but the algorithm still uses same small steps thus computation time is wasted to generate solutions that are too close from each other. And since $\lambda \to \infty$ as the model shrinks to empty, to stop the algorithm, a $\lambda^{\max}$ needs to be selected in advance. BLasso suffers none of these problems. Also, for situations like boosting trees where the number of basic learners is huge and at each step the minimization of empirical loss can only be done through approximation tricks, BLasso can be easily adapted by replacing exact minimization with approximate minimization.

7 Experiment

Tow experiments are carried out to illustrate BLasso with both simulated and real datasets. We first run BLasso on a diabetes dataset [5] under the classical Lasso setting, i.e. L_2 regression with an L_1 penalty. Then, switching from regression to classification, we use simulated data to illustrate BLasso solving regularized classification problem under the 1-norm SVM setting.

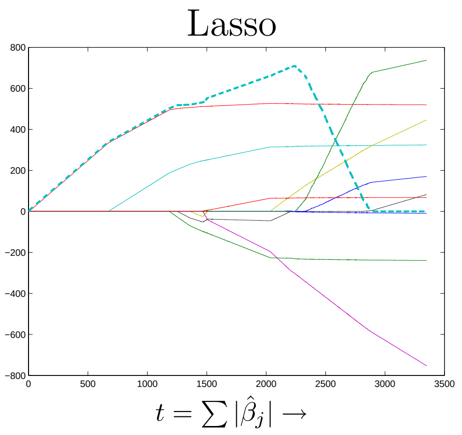


Figure 1: Lasso estimates of regression coefficients as a function of $t = \|\beta\|_1$.

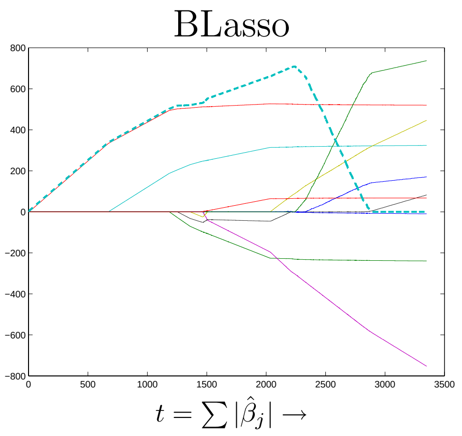


Figure 2: BLasso solutions, which can be seen identical to the Lasso solutions.

7.1 L_2 Regression with L_1 Penalty (Classical Lasso)

The dataset used in this and the following experiment is from a Diabetes study where diabetes patients were measured on 10 baseline variables. A prediction model was desired for the response variable, a quantitative measure of disease progression one year after baseline. One additional variable, $X_{11} = -X_7 + X_8 + 5X_9$, is added to make the difference between FSF and Lasso solutions more visible.

The classical Lasso – L_2 regression with L_1 penalty is used for this purpose. Let $X^1, X^2, ..., X^m$ be $n-$vectors representing the covariates and Y the vector of responses for the n cases, $m = 11$ and $n = 442$ in this study. Location and scale transformations are done

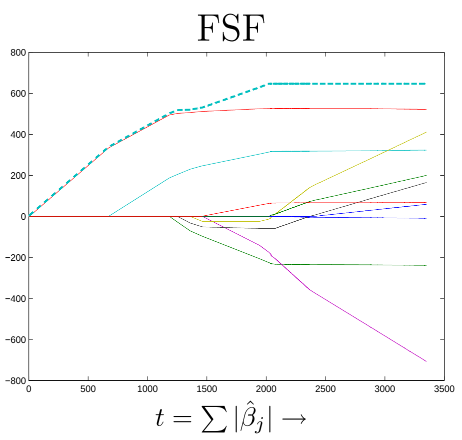


Figure 3: Forward Stagewise Fitting solutions, which are different from Lasso solutions.

so that all covariates are standardized to have mean 0 and unit length, and that the response has mean zero.

The penalized loss function has the form:

$$\Gamma(\beta; \lambda) = \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_1 \tag{7.12}$$

Figure 2 shows the coefficient plot for BLasso applied to the diabetes data. Figure 1 (Lasso) and 2 (BLasso) are indistinguishable from each other. Both FSF and BLasso pick up X_{11} (the dashed line) in the earlier stages, but due to the greedy nature of FSF, it is not not able to correct the mistake and remove X_{11} in the later stages thus every parameter estimate is affected which leads to significantly different solutions from Lasso.

The BLasso solutions were built up in 8700 steps (making the step size $\epsilon = 0.5$ small enough so that the plots are smooth enough) which consist 840 backward steps. In comparison, Forward Stagewise Fitting took 7300 pure forward steps. BLasso's backward steps mainly concentrate around the spots where Forward Stagewise Fitting and BLasso tend to differ.

7.2 Classification with 1-norm SVM (Hinge Loss) In addition to the regression experiment in the previous section, we also look at binary classification. We generate 50 training data in each of two classes. The first class has two standard normal independent inputs X^1 and X^2 and class label $Y = -1$. The second class also has two standard normal independent inputs, but conditioned on $4.5 \leq (X^1)^2 + (X^2)^2 \leq 8$ and has class label $Y = 1$. We wish to find a classification rule from the training data. so that when given a new input, we can assign a class Y from $\{1, -1\}$ to it.

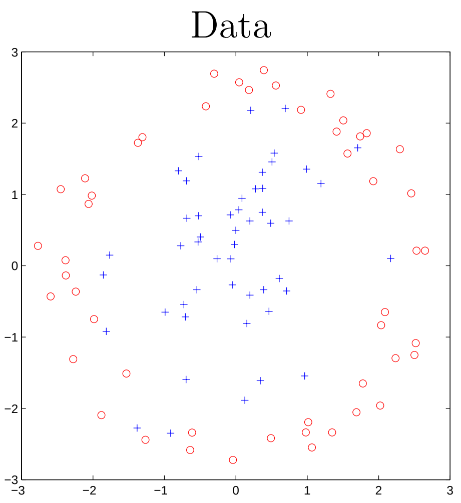


Figure 4: Scatterplot of the data points with labels: '+' for $y = -1$; 'o' for $y = 1$.

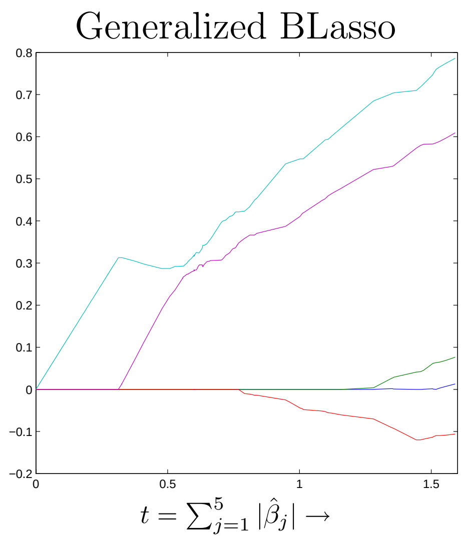


Figure 5: Estimates of 1-norm SVM coefficients $\hat{\beta}_j$, j=1,2,...5, for the simulated two-class classification data. BLasso solutions are plotted as functions of $t = \sum_{j=1}^{5} |\hat{\beta}_j|$.

To handle this problem, 1-norm SVM [17] is considered:

$$(\hat{\beta}_0, \hat{\beta}) = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n}(1-Y_i(\beta_0+\sum_{j=1}^{m}\beta_j h_j(X_i)))^{+}+\lambda\|\beta\|_1 \tag{7.13}$$

where h_i are basis functions and λ is the regularization parameter. The dictionary of basis functions considered here is $D = \{\sqrt{2}X^1, \sqrt{2}X^2, \sqrt{2}X^1X^2, (X^1)^2, (X^2)^2\}$. The fitted model is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^{m}\hat{\beta}_j h_j(x).$$

The classification rule is given by $\text{sign}(\hat{f}(x))$.

Since neither the hinge loss function nor the penalty function is differentiable, Theorom 3.1 does not hold. However Generalized BLasso ran without a problem. It takes Generalized BLasso 490 iterations to generate the solutions. The covariates enter the regression equation sequentially as t increase, in the following order: the two quadratic terms first, followed by the interaction term then the two linear terms.

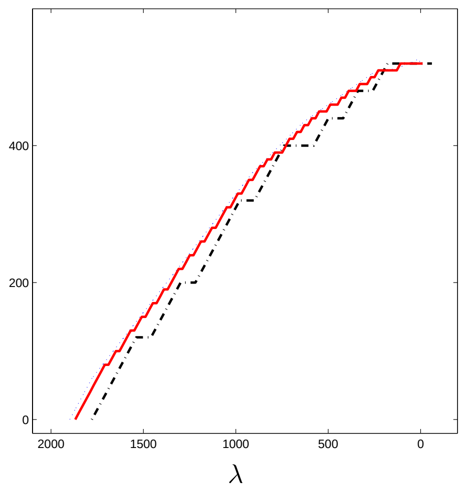


Figure 6: Estimates of regression coefficients $\hat{\beta}_3$ for the diabetes data. Solutions are plotted as functions of λ. *Dotted Line* – Estimates using step size $\epsilon = 0.05$. *Solid Line* – Estimates using step size $\epsilon = 10$. *Dash-dot Line* – Estimates using step size $\epsilon = 50$.

8 Discussion and Concluding Remarks

As seen from the experiments, BLasso is effective for solving the Lasso problem and general convex penalized loss minimization problems. One practical issue left undiscussed is the choice of stepsize. In general, BLasso take $O(1/\epsilon)$ steps to produce the whole path. For simple L_2 regression with m covariates, each step uses $O(m \cdot n)$ basic operations. Depend on the actual loss function, base learners and minimization trick used in each step, the actual computation complexity varies. Although choice of smaller step size gives smoother solution path and more accurate estimates, we observe that the the actual coefficient estimates are pretty accurate even for relatively large step sizes.

As can be seen from Figure 6, for small step size $\epsilon = 0.05$, the solution path can not be distinguished from the exact regularization path. However, even when the step size is as large as $\epsilon = 10$ and $\epsilon = 50$, the solutions are still good approximations.

BLasso has only one step size parameter. This parameter controls both how close BLasso approximates the minimization coefficients for each λ and how close two adjacent λ on the regularization path are placed. As can be seen from Figure 6, a smaller stepsize leads to a closer approximation to the solutions and also finer grids for λ. We argue that, if λ is sampled on a coarse grid there is no point of wasting computational power on finding a much more accurate approximation of the coefficients for each λ. Instead, the available computational power spent on these two coupled tasks should be balanced. BLasso's 1-parameter setup automatically balances these two aspects of the approximation which is graphically expressed by the staircase shape of the solution paths.

One of our current research topics is to apply BLasso in an online setting. Since BLasso has both forward and backward steps, it should be able to transform into an adaptive online learning algorithm where it goes back and forth to track the best regularization parameter and the corresponding model.

In this paper, we introduced the Boosted Lasso algorithms is able to produce the complete regularization path for general convex loss function with convex penalty. To summarize, we showed that

1. As a combination of both forward and backward steps, a Boosted Lasso (BLasso) algorithm can be constructed to efficiently produce the Lasso solutions for general loss functions which can be proven rigorously under the assumption that the loss function is convex and continuously differentiable.

2. Backward steps are critical for producing the Lasso path. Without them, the Forward Stagewise Fitting algorithm can be too greedy and in general does not produce with Lasso solutions.

3. When the loss function is square loss, BLasso takes a simpler and more intuitive form.

4. BLasso can be generalized to deal with general convex loss function with convex penalty.

A Appendix: Proofs

First, we offer a proof for Lemma 3.1.

Proof. (Lemma 3.1)

1. Suppose $\exists\lambda, j, |s| = \epsilon$ s.t. $\Gamma(\epsilon 1_j;\lambda) \le \Gamma(0;\lambda)$. We have
$$\sum_{i=1}^n L(Z_i;0) - \sum_{i=1}^n L(Z_i;s1_j) \ge \lambda T(s1_j) - \lambda T(0),$$
therefore
$$\begin{aligned}\lambda &\le \frac{1}{\epsilon}\{\sum_{i=1}^n L(Z_i;0) - \sum_{i=1}^n L(Z_i;s1_j)\}\\ &\le \frac{1}{\epsilon}\{\sum_{i=1}^n L(Z_i;0) - \min_{j,|s|=\epsilon}\sum_{i=1}^n L(Z_i;s1_j)\}\\ &= \frac{1}{\epsilon}\{\sum_{i=1}^n L(Z_i;0) - \sum_{i=1}^n L(Z_i;\hat\beta^0)\}\\ &= \lambda^0.\end{aligned}$$

2. Since a backward step is only taken when $\Gamma(\hat\beta^{t+1};\lambda^t) < \Gamma(\hat\beta^t;\lambda^t)$, so we only need to consider the forward step. When a forward step is forced, if $\Gamma(\hat\beta^{t+1};\lambda^t) > \Gamma(\hat\beta^t;\lambda^t)$, then
$$\sum_{i=1}^n L(Z_i;\hat\beta^t) - \sum_{i=1}^n L(Z_i;\hat\beta^{t+1}) < \lambda^t T(\hat\beta^{t+1}) - \lambda^t T(\hat\beta^t),$$
therefore
$$\lambda^{t+1} = \frac{1}{\epsilon}\{\sum_{i=1}^n L(Z_i;\hat\beta^t) - \sum_{i=1}^n L(Z_i;\hat\beta^{t+1})\} < \lambda^t$$
which contradicts the assumption.

3. Since $\lambda^{t+1} < \lambda^t$ and λ can not be relaxed by a backward step, we immediately have $\|\hat\beta^{t+1}\|_1 = \|\hat\beta^t\|_1 + \epsilon$. Then from
$$\lambda^{t+1} = \frac{1}{\epsilon}\{\sum_{i=1}^n L(Z_i;\hat\beta^t) - \sum_{i=1}^n L(Z_i;\hat\beta^{t+1})\}$$
we get
$$\Gamma(\hat\beta^t;\lambda^{t+1}) = \Gamma(\hat\beta^{t+1};\lambda^{t+1}).$$
Plus both sides by $\lambda^t - \lambda^{t+1}$ times the penalty term, and recall $T(\hat\beta^{t+1}) = \|\hat\beta^{t+1}\|_1 > |\hat\beta^t\|_1 = T(\hat\beta^t)$, we get
$$\begin{aligned}\Gamma(\hat\beta^t;\lambda^t) &< \Gamma(\hat\beta^{t+1};\lambda^t)\\ &= \min_{j,|s|=\epsilon}\Gamma(\hat\beta^t + s1_j;\lambda^t)\\ &\le \Gamma(\hat\beta^t \pm \epsilon 1_j;\lambda^t)\end{aligned}$$
for $\forall j$. This completes the proof.

Theorem 3.1 claims "the BLasso path converges to the Lasso path", by which we mean:

1. As $\epsilon \to 0$, for $\forall t$ s.t. $\lambda^{t+1} < \lambda^t$, $\hat\beta^t \to \beta^*(\lambda^t)$ where $\beta^*(\lambda^t)$ is the Lasso solution for $\lambda = \lambda^t$.
2. For each $\epsilon > 0$, it takes finite steps to run BLasso.

Proof. (Theorem 3.1)

1. Since $L(Z;\beta)$ is strictly convex in β, so $\Gamma(\beta;\lambda)$ is strictly convex in β for $\forall\lambda$. So $\Gamma(\beta;\lambda)$ has unique minimum and has no stationary point except at the minimum. Lemma 3.1 says BLasso does steepest coordinate descent with fixed step size ϵ. So we only need to check if BLasso can get stuck around nonstationary points.

   Consider at $\hat\beta^t$, we look for the steepest descent on the surface of a polytope defined by $\beta = \hat\beta^t + \Delta\beta$ where $\|\Delta\beta\|_1 = \epsilon$, i.e.
$$(1.14)\qquad \min_{\Delta\beta}\Delta\Gamma(\hat\beta^t + \Delta\beta;\lambda) \quad \text{subject to } \|\Delta\beta\|_1 = \epsilon.$$
   Here
$$\Delta\Gamma = \Delta L + \lambda\Delta T.$$
   Since L is continuously differentiable w.r.t. β, we have
$$(1.15)\qquad \Delta L = \sum_j \frac{\partial L}{\partial\beta_j}\Delta\beta_j + o(\epsilon).$$
   And since $T(\beta) = \sum_j |\beta_j|$, we have
$$(1.16)\qquad \Delta T = \sum_j \text{sign}^+(\hat\beta^t, \Delta\beta_j)\|\Delta\beta_j\|.$$
   Therefore as $\epsilon \to 0$, (1.14) becomes a linear programming problem, for which the solution is always achieved on the edges where $\Delta\beta = s1_j$ for some j and $|s| = \epsilon$. Thus, to do steepest descent on $\|\Delta\beta\|_1$, one only need to look on the coordinates.

   This indicates that, if $\Gamma(\hat\beta^t;\lambda^t) < \Gamma(\hat\beta^t + s1_j;\lambda^t)$, $\forall j, s$ where $|s| = \epsilon$, then $\Gamma(\hat\beta^t;\lambda^t) < \Gamma(\hat\beta^t + \Delta\beta;\lambda^t)$, $\forall\Delta\beta$ where $\|\Delta\beta\|_1 = \epsilon$. Now since Γ is strictly convex with unique minimum $\beta^*(\lambda)$, this must means the minimum is inside the polytope:
$$(1.17)\qquad \|\beta^*(\lambda) - \hat\beta^t\| < \epsilon,$$
   which gives the proof.

2. First, suppose we have $\lambda^{t+1} < \lambda^t$, $\lambda^{t'+1} < \lambda^{t'}$ and $t < t'$. Immediately, we have $\lambda^t > \lambda^{t'}$, then
$$\Gamma(\beta^{t'};\lambda^{t'}) < \Gamma(\beta^t;\lambda^{t'}) < \Gamma(\beta^t;\lambda^t) < \Gamma(\beta^{t'};\lambda^t).$$

Therefore

$$\Gamma(\beta^{t'};\lambda^t) - \Gamma(\beta^{t'};\lambda^{t'}) > \Gamma(\beta^t;\lambda^t) - \Gamma(\beta^t;\lambda^{t'}),$$

from which we get

$$T(\beta^{t'}) > T(\beta^t).$$

So the BLasso solution before each time λ gets relaxed strictly increases in L_1 norm. Then since the L_1 norm can only change on an ϵ-grid, so λ can only be relaxed finite times till BLasso reaches the unregularized solution.

Now for each value of λ, since BLasso is always strictly descending, the BLasso solutions never repeat. By the same ϵ-grid argument, BLasso can only take finite steps before λ has to be relaxed.

Combining the two arguments, we conclude that for each $\epsilon > 0$ it can only take finite steps to run BLasso.

References

[1] Breiman, L. (1998). "Arcing Classifiers", *Ann. Statist. 26, 801-824.*

[2] Breiman, L. (1999). "Prediction Games and Arcing Algorithms", *Neural Computation 11, 1493-1517.*

[3] Buhlmann, P. and Yu, B. (2001). "Boosting with the L2 Loss: Regression and Classification", *J. Am. Statist. Ass. 98, 324-340.*

[4] Donoho, D. and Elad, M. (2004). "Optimally sparse representation in general(non-orthogonal) dictionaries vy l_1 minimization", *Technical reports, Statistics Department, Stanford University.*

[5] Efron, B., Hastie,T., Johnstone, I. and Tibshirani, R. (2002). "Least Angle Regression", *Ann. Statist. 32 (2004), no. 2, 407-499.*

[6] Freund, Y. (1995). "Boosting a weak learning algorithm by majority", *Information and Computation 121, 256-285.*

[7] Freund, Y. and Schapire, R.E. (1996). "Experiments with a new boosting algorithm", *Machine Learning: Proc. Thirteenth International Conference, pp. 148-156. Morgan Kauffman, San Francisco.*

[8] Friedman, J.H., Hastie, T. and Tibshirani, R. (2000). "Additive Logistic Regression: a Statistical View of Boosting", *Ann. Statist. 28, 337-407.*

[9] Friedman, J.H. (2001). "Greedy Function Approximation: a Gradient Boosting Machine", *Ann. Statist. 29, 1189-1232.*

[10] Hansen, M. and Yu, B. (2001). "Model Selection and the Principle of Minimum Description Length", *J. Am. Statist. Ass. Vol. 96, 746-774.*

[11] Hastie, T., Tibshirani, R. and Friedman, J.H. (2001).*The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer Verlag, New York

[12] Li, S. and Zhang, Z. (2004). "FloatBoost Learning and Statistical Face Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence Vol 26, 1112-1123*

[13] Mason, L., Baxter, J., Bartlett, P. and Frean, M. (1999). "Functional Gradient Techniques for Combining Hypotheses", In *Advance in Large Margin Classifiers.* MIT Press.

[14] Rosset, S. (2004). "Tracking Curved Regularized Optimization Solution Paths", *NIPS 2004, to appear.*

[15] Schapire, R.E. (1990). "The Strength of Weak Learnability". *Machine Learning 5(2), 1997-227.*

[16] Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso", *J. R. Statist. Soc. B, Vol. 58, No. 1., pp. 267-288.*

[17] Zhu, J. Rosset, S., Hastie, T. and Tibshirani, R.(2003) "1-norm Support Vector Machines", *Advances in Neural Information Processing Systems 16.* MIT Press.

Feature Selection with a General Hybrid Algorithm

Jerffeson Souza[*] Nathalie Japkowicz[†] Stan Matwin[‡]

Abstract
Feature subset selection algorithms can be classified into three broad categories: filters, wrappers and hybrid algorithms. In this paper, we develop a framework to help us classify and study hybrid solutions for the feature selection problem. In addition, we propose a new general hybrid solution named FortalFS. This algorithm uses results from another feature selection system as a starting point in the search through subsets of features that are evaluated by a machine learning algorithm. The search is performed in a stochastically guided fashion. FortalFS is empirically shown to outperform several well-known filter and wrapper feature selection algorithms.

Keywords: FortalFS, Hybrid Feature Selection, Machine Learning.

1 Introduction

Classification is a key problem in machine learning. Algorithms for classification have the ability to predict the outcome of a new situation after having been trained on data representing past experience. A number of factors influence the performance of classification algorithms, including the number and quality of features provided to describe the data, the training and testing data distribution, and others. The factor we focus on in this paper is the number and quality of features present in the sample data. The *Feature Selection* problem involves discovering a subset of features such that a classifier built only with this subset would have better predictive accuracy than a classifier built from the entire set of features. Other benefits of feature selection include a reduction in the amount of training data needed to induce an accurate classifier, that is consequently simpler and easier to understand, and a reduced execution time. In practice, feature selection algorithms will discover and select features of the data that are *relevant* to the task to be learned.

Feature subset selection algorithms can be classified into three broad categories based on whether or not feature selection is done independently of the learning algorithm used to construct the classifier. If feature selection is performed independently of the learning algorithm, the technique is said to follow a *filter* approach. Otherwise, it is said to follow a *wrapper* approach. While the filter approach is generally computationally more efficient than the wrapper approach, its major drawback is that an optimal selection of features may not be independent of the inductive and representational biases of the learning algorithm that is used to construct the classifier. The wrapper approach on the other hand, involves the computational overhead of evaluating candidate feature subsets by executing a selected learning algorithm on the dataset represented using each feature subset under consideration. A combination of these two approaches, that is, the use of two evaluation methods (a filter-type evaluation function and a classifier) creates a *hybrid* solution. Hybrid solutions attempt to combine the good characteristics of both filters and wrappers. For a more detailed overview of previous work in feature selection research on filters and wrappers please see [4].

The remainder of this paper is composed of five sections. Section 2 presents a framework for hybrid feature selection algorithms. Section 3 presents our general hybrid approach for feature selection, FortalFS. In Section 4, we discuss how FortalFS relates to the framework. Section 5 presents an empirical evaluation of our method. Finally, Section 6 concludes the paper by summarizing its contributions.

2 A Framework for Hybrid Feature Selection

2.1 Introduction Only a few hybrid solutions for feature selection have been proposed thus far. By taking into consideration both the type of filter evaluation measure and the classifier used by hybrid feature selection algorithms, we are able to introduce a framework to help us organize and study hybrid methods. The filter evaluation methods used in our framework are based on Distance, Information Gain, Dependency and Consistency. These classes of evaluation functions have been described previously in [6]. The classifiers are Decision Tree, k-Nearest Neighbour, Gaussian classifier and “Others”. The “Others” class is used to indicate that the algorithms may employ any other learning algorithm.

[*]Computer Science Department, Federal University of Ceará, Fortaleza, Ceará, 60.455-760, Brazil.

[†]School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada.

[‡]School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada.

2.2 The Framework Table 1 presents the framework for hybrid feature selection algorithms. In the table, a plus sign ($^{+}$) next to a particular method in a certain category indicates that such a method can be adapted to fall under this category, even though no practical attempt has been made to do so.

In [12], the authors start by proposing a new criterion to estimate the relevance of features called *Relative Certainty Gain* (RCG)[1]. The RCG evaluation method assumes that the leaner's ability to correctly classify label instances depends on the existence of wide geometrical structures (characterized using a Minimum Spanning Tree built on the learning data) of identical label points. Next, a new filter for feature selection is proposed, where subsets are generated by a greedy forward selection algorithm and evaluated according to the RCG measure. The authors then address the similarities between the Minimum Spanning Tree (MST) and the 1-Nearest Neighbour (1-NN) graph, which allows for the replacement of the MST by the 1-NN graph. The 1-NN graph besides being less expensive to compute also helps shifting the behaviour of the proposed feature selection algorithm toward wrapper approaches, even though classification accuracy is not used.

The filter component of Xing, Jordan and Karp's hybrid algorithm [15] is build in three phases. Initially, *unconditional univariate mixture modeling* is used mainly to discretize the measurements for a given feature. Next, the algorithm ranks all features according to an *information gain* measure which is used as initial filter. The remaining features are then passed to the more computationally expensive third phase. In this final filter step, the authors propose the use of *markov blanket filtering* to select subsets of features for each subset cardinality. Finally, each subset is evaluated via cross validation and the best one is returned.

Bala and others [2] propose a feature selection hybrid strategy that integrates genetic algorithms and decision tree learning. In their algorithm, a genetic system drives the search for subsets of features. The fitness value for each subset F to be maximized is expressed as: $Fitness(F) = Inf(F) - Cost(F) + Acc(F)$. $Inf(F)$ is a value based on a technique that estimates the discriminatory power of each feature calculated using an entropy measure. $Cost(F)$ is a simple measure of cost which is directly proportional of the cardinality of subset F. Finally, $Acc(F)$ is a measure of the classification accuracy of feature subset F obtained by inducing a decision tree.

[1]Since this new evaluation method is similar to the Information Gain criterion used by several filter feature selection algorithms, we have considered this feature selection method to fall under the Information Gain category.

The BBHFS (Boosting Based Hybrid Feature Selection) algorithm [5] is an extension of the filter BDSFS-2. The BDSFS (Boosted Decision Stump Feature Selection) algorithm [5] applies a forward selection search strategy. The selection of the next feature to be considered is based on the information gain criterion and takes into consideration the weight of each dataset instance. This selected feature is then added to the set that will be returned and used to create a decision stump (used as the weak learner), which updates the weights of the dataset examples by assigning higher weights to examples that have often been misclassified in this round. The process repeats until a pre-specified number of features has been selected, in a process very similar to boosting. A variant of this algorithm, referred to as BDSFS-2, avoids the pre-specification of the number of features to be returned. To achieve that, new features should be added to the final subset as long as this addition results in increased training accuracy. In BBHFS, a learning algorithm is used to drive the search. For that matter, the reweight process is changed so that the weak hypothese used in each round of the boosting process are the concepts the learning algorithm would learn from the unweighted training set when using just the features in the set thus far. However, both the selection of the next feature to be added, performed on the basis of weighted information gain, and stopping criterion remain the same as in BDSFS-2.

The hybrid algorithm ADHOC [11] comprises two main steps, namely the *Data Reduction* step and the *Feature Selection* step. In the first step an iterative process is applied to explore dependencies between data and to partition the set of observed features into a small number of clusters (factors). The search for true association between the data is based on the concept of feature *profile*, that denotes which other features one is related to. In the feature selection step, ADHOC selects at most one feature from each of the factors (data dimension) that has been discovered in the data reduction step by using a wrapper approach. Several heuristics were investigated in this phase, with genetic algorithms (GA) having excellent results. In addition to its selection capabilities, ADHOC is able to rank the features by analyzing the distribution of features in the final population generated by the genetic algorithm.

2.3 Discussion We can make a few remarks about the framework for hybrid feature selection algorithms described above. First, it is clear that the number of hybrid solutions proposed to this date is still very small. Second, one can verify that there is a concentration of methods based on only a few learning systems. Finally, we can confirm that there is a considerable number

Filter Evaluation Measure	**Classifier**			
	Decision Tree	**k-Nearest Neighbour**	**Gaussian Classifier**	**Others**
Distance				
Information Gain	Bala96 BBHFS Xing01$^+$	Sebban02 Xing01	Xing01	Xing01$^+$
Dependency	ADHOC	ADHOC$^+$	ADHOC$^+$	ADHOC$^+$
Consistency				

Table 1: A Framework for Hybrid Feature Selection.

of combinations of evaluation methods that have not been tried. In addition, even though Xing01 and ADHOC allow for the use of any classifier, no practical attempt has been made to use the bias of other learning algorithms such as Naive Bayes, Neural Nets, SVM, and others.

As a conclusion, it is reasonable to assume that the feature selection field could benefit from a general hybrid algorithm that could assume any position in the framework just by "plugging" different evaluation methods. That flexibility would allow for such an algorithm to have its behaviour shifted when required.

3 The FortalFS Algorithm

The idea behind FortalFS is to extract and combine the best characteristics of filters and wrappers into one algorithm, namely, an efficient heuristic used to search through subsets of features and a precise evaluation criterion, respectively. Thus, the FortalFS algorithm uses results from another feature selection system as a starting point in the search through subsets of features that are evaluated by a machine learning algorithm. Therefore, with an efficient heuristic, we can decrease the number of subsets of features to be evaluated by the learning algorithm, consequently decreasing computational effort (the major advantage of filters) and still be able to select an accurate final subset (the major advantage of wrappers).

Initially, the k best subsets returned by a single run of a feature selection system (or the single results of k different runs, if such an algorithm returns only one best subset per execution) are stored into a two-dimensional array, see Figure 1. This array will then be condensed into a new array, called *Adam*, that will simply store the number of times each feature appeared in the k best subsets. Next, FortalFS will iteratively generate new subsets of features in a stochastically guided fashion using *Adam* as a seed and evaluate them with a learning system. The generation of a new subset is such that features with high value in *Adam* have a better chance of being selected than those with a low one at each iteration. At the end, the subset with best accuracy will be returned. If subsets tie it terms of accuracy, the one with the lowest cardinality is returned.

```
FortalFS($D, NumIter$)

$O$ = FeatureSelector($D$)
$Adam$ = CalculateAdam($O$)
for $i = 1$ to $NumIter$
    $S$ = GenerateSubset($Adam$)
    if ErrorRate($S, D$) < ErrorRate($S_{best}, D$) then
        $S_{best} = S$
    else
        if ErrorRate($S, D$) = ErrorRate($S_{best}, D$) and
          $Card(S) < Card(S_{best})$ then
            $S_{best} = S$
return $S_{best}$

———-
where:
    $D$ - dataset.
    $NumIter$ - number of iterations.
```

Figure 1: The FortalFS Algorithm.

We describe next, in detail, each of the methods used by FortalFS.

FeatureSelector(D) runs a feature selection system getting the k best subsets generated and storing them into the two-dimensional vector O.

There are a few characteristics that make a feature selection algorithm suitable to be used as underlying algorithm in FortalFS. First, the algorithm must be *non-deterministic*, otherwise, the k best subsets would be the same and FortalFS would consequently select this same subset all the time. For instance, the Focus algorithm [1] is not a good candidate because of its deterministic behaviour. Second, it should be ideally an *anytime algorithm*, that is, being able to output several partial results during processing. This way, one can obtain the k best results in one single run of the algorithm. LVF [9] is an example of such algorithms. Finally, the algorithm should in fact be a *selection algorithm*, not a weighting algorithm such as the original Relief algorithm [8]. However, FortalFS can be modified to work with feature weights directly. We present and evaluate this modification later on.

CalculateAdam(O) uses the following equation:

$$Adam = \{a_i, 1 \le i \le n\}$$

where: $a_i = \sum o_{ji}$, with $1 \le j \le k$ and $1 \le i \le n$.

to create the *Adam* vector, which stores the number of occurrences of each feature in O.

GenerateSubset($Adam$) generates a new subset of features S in a stochastically guided fashion using *Adam* as a seed. The generation process works as described below. Let i denote a particular feature in *Adam*. Let S be a vector of n elements where n is the total number of features in O. Element S_i (of S) $= 1$ if feature i is included in the subset of features represented by S. $S_i = 0$, otherwise. Vector S is computed as follows:

$S_i = 1$, if $a_i > random(k)$ and $S_i = 0$ otherwise,

where $random(k)$ returns a random number between 0 and k.

This procedure is such that features with high frequency have a better chance of being selected than those with a low one at each iteration.

ErrorRate(S, D) makes use of a learning algorithm, inputting the subset S to generate a prediction model and receiving the error rate calculated for this model over dataset D.

4 FortalFS in the Hybrid Feature Selection Framework

Some research in the feature selection field have focused on the development of hybrid solutions. Researchers have been trying to combine different evaluation functions and learning algorithm biases in order to find a good match that will improve selection, as exemplified in the framework for hybrid features selection algorithms described previously.

The FortalFS algorithm, as described in the previous section, allows the use of any evaluation criterion as well as any learning system in a way that different combinations can be applied. This flexibility permits us to shift the FortalFS behaviour toward different categories under the hybrid feature selection framework. In fact, FortalFS is a *general* hybrid solution that can be configured to assume any position in the framework just by attaching different evaluation methods.

5 Empirical Evaluation

In this section, we first describe our experimental setting and then present and discuss our results.

5.1 Methodology In order to evaluate FortalFS, several feature selection algorithms were implemented and their performances compared to our new hybrid algorithm. The algorithms used are Best-First Search [16], Genetic Search (GA) [14], LVF [9], Relief[2] [8], Focus [1], Forward Wrapper [7], Backward Wrapper [7] and a Random Wrapper[3]. We performed then a series of experiments using three different classifiers (C4.5, Naive Bayes and k-Nearest Neighbour) and 13 datasets from the UCI Repository [3]: Credit (15 features, 690 instances), Labor (16, 57), Vote (16, 435), Primary Tumor (17, 339), Lymph (18, 148), Mushroom (22, 8124), Colic (23, 368), Autos (25, 205), Ionosphere (34, 351), Soybean (35, 683), Splice (60, 3190), Sonar (60, 208), Audiology (69, 226).

Performance measures such as the accuracy of the selected subset, the time used for selection and the number of features selected were obtained in each experiment. The accuracy for each selected subset was obtained as follows: the original dataset was randomly and equally split into a selection set and a testing set. The feature selection in all cases was performed considering only the selection set. For the wrappers we evaluated the subsets of features with 5-fold cross validation on the selection set. Finally, the selected subset was then evaluated using 5-fold cross validation on the testing set. All methods are compared using this independent evaluation to avoid the overfitting problem discussed in [10].

5.2 Experimental Settings The following configurations were used in the experiments: for the BestFirst algorithm the number of non-improving expansions before termination was set to 5. For the Genetic algorithm maximum number of populations is 200, the size of each population was set to 50, the mutation probability is 0.001 and crossover probability is 0.6. For LVF the inconsistency threshold is initial inconsistency of the dataset and the number of iterations is $77 \cdot N^5$, where

[2]The Relief version we use in our experiments is a "selection" version of the original "weighting" Relief algorithm.

[3]In this random wrapper, adapted from [7], subsets of features are iteratively and randomly generated and evaluated with the help of a machine learning algorithm. At the end, the subset with best accuracy is returned.

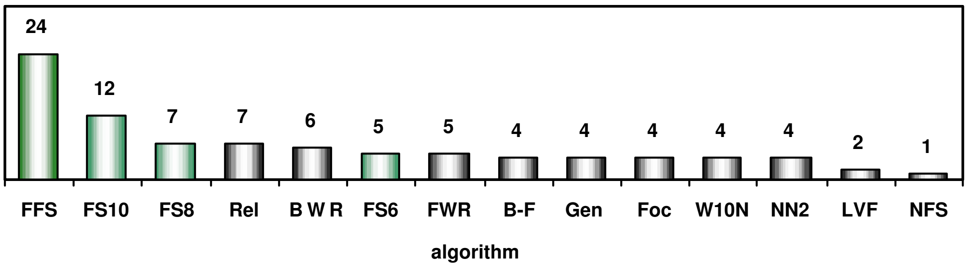


Figure 2: Overall performance of all algorithms in terms of accuracy. Number of experiments which each algorithm performed the best or tied with the best.

N is the number of features in the original dataset. The number of iterations in Relief is number of instances in the dataset, the number of $NearHits$ and $NearMisses$ considered was set to 10 and the selection threshold to 0.01. For the Random Wrappers the number of iterations is $10 \cdot N$ and N^2. Finally, for FortalFS the number of iterations tried are $6 \cdot N$, $8 \cdot N$ and $10 \cdot N$, the underlying selection algorithm is LVF and k was set to 10.

5.3 Experimental Results and Analysis In the next section, we will present and discuss the results[4] obtained in our experiments with FortalFS and other feature selection algorithms in a general manner.

5.3.1 Overall Results and Analysis As shown in Figure 2, the three FortalFS settings (FS10, FS8 and FS6) are among the best algorithms in terms of accuracy. FortalFS($10 \cdot N$) had the best performance in 12 cases, FortalFS($8 \cdot N$) and Relief in 7, the Backward Wrapper in 6, and FortalFS($6 \cdot N$) and the Forward Wrapper in 5. When considering the best FortalFS result in each case (FFS), FortalFS performs at least as well as all other algorithms in 24 out of the 39 experiments.

As expected, the FortalFS performance was proportional to the number of subsets considered, that is, FortalFS($10 \cdot N$) performed better than FortalFS($8 \cdot N$) that was better than FortalFS($6 \cdot N$).

The Random Wrappers did worse than FortalFS, Forward and Backward Wrappers in most cases. An important and expected conclusion that can be extracted from this result is that the strength of FortalFS, Forward and Backward Wrappers come also from the search heuristic they apply and not only from their strong evaluation method.

In terms of time consumption (Figure 3), as expected, the wrappers and FortalFS are down in the list, which shows the impact of the evaluation method. However, the three FortalFS settings were faster than all wrappers.

Figure 4 shows the percentage of the features selected by each algorithm. The Forward Wrapper, Best-First and Genetic algorithms selected the smallest number of features overall choosing respectively 16.91%, 19.76% and 21.95% of all features. FortalFS was able to achieve a dimensionality reduction of over 60% and still select very accurate subsets. The Backward Wrapper, with 1072 features selected (87.15%), is on the top of the list.

5.3.2 Pairwise Comparisons In this section, we examine with more details the differences in terms of accuracy obtained in the experiments between FortalFS and the three filters (LVF, Focus and Relief) along with the wrappers (Forward, Backward and Random). Table 2 summarizes the results of these pairwise comparisons.

By comparing the results obtained with FortalFS versus LVF, we can get a good measure of the ability of the first algorithm to improve the performance of the second (since we used LVF in our FortalFS implementation as underlying feature selector). Table 2 shows us that FortalFS significantly (at least within the 0.1 significance level) outperformed LVF in 27 out of the 39 experiments and it is significantly outperformed only once.

Specifically compared to Focus, FortalFS outperformed this algorithm in 30 cases (significantly in 25 of them) and it is outperformed only in 6.

[4]The following acronyms will be used on the tables/figures to refer to each algorithm: NFS (No feature Selection - C4.5, Naive Bayes or k-Nearest Neighbour with original dataset), FFS (best FortalFS result among the three settings), FS10 (FortalFS($10 \cdot N$)), FS8 (FortalFS($8 \cdot N$)), FS6 (FortalFS($6 \cdot N$)), FWR (Forward Wrapper), BWR (Backward Wrapper), W10N (Random Wrapper($10 \cdot N$)), WN2 (Random Wrapper(N^2)), B-F (Best-First Search), Gen (Genetic Search), LVF (LVF), Rel (Relief), Foc (Focus).

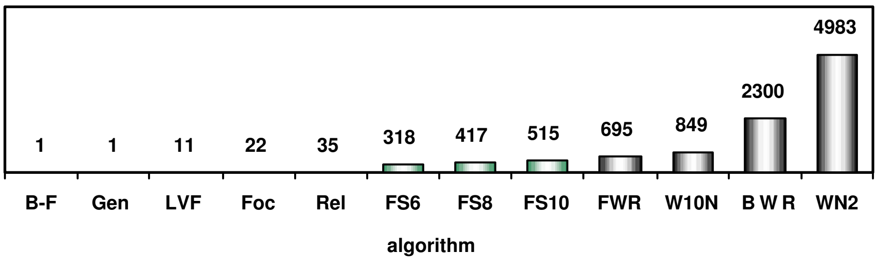


Figure 3: Overall time consumption (in minutes) for each algorithm considering all experiments.

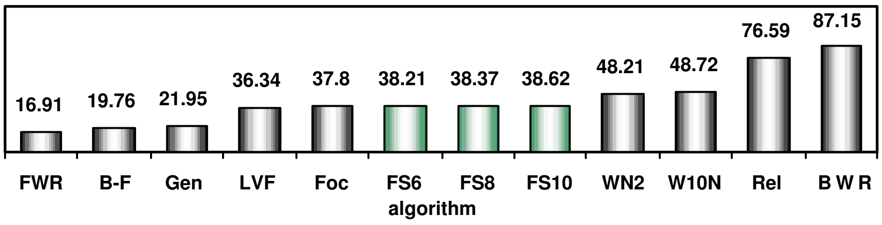


Figure 4: Percentage of the features selected by each algorithm in all experiments from a total number of 1230 features.

	<0.001	**<0.005**	**<0.01**	**>0.01**
FortalFS vs LVF	19 x 0	4 x 0	4 x 1	6 x 3
FortalFS vs Focus	21 x 1	4 x 2	0 x 1	5 x 2
FortalFS vs Relief	14 x 4	5 x 2	0 x 0	7 x 6
FortalFS vs Forward Wrapper	14 x 2	1 x 1	4 x 1	9 x 3
FortalFS vs Backward Wrapper	14 x 3	2 x 0	2 x 0	8 x 9
FortalFS vs Random Wrapper (N^2)	10 x 2	3 x 1	3 x 1	11 x 4

Table 2: Score of the number of experiments (out of 39) each algorithm performed better within each significance level (calculated with the student's t-test). A score "A x B" for a certain algorithm f and significance level s means that the best FortalFS setting performed better than f within s A times. Similarly, it also means that algorithm f outperformed the best FortalFS setting B times within s.

Relief was the filter algorithm that performed the best in terms of accuracy overall. It was able to perform as well as FortalFS($8 \cdot N$). However, when one considers the best FortalFS result, Relief is significantly outperformed in about half of the time.

The importance of the comparison between FortalFS and both Forward and Backward Wrappers relies on the fact that all these algorithms take advantage of the same evaluation method. Such a fact give us a common background to analyse exclusively the performance differences between the search heuristics. With that in mind one can conclude that the FortalFS search method performs better in most cases. In fact, FortalFS outperformed the Forward Wrapper algorithm within the 0.01 significance level in 14 experiments (out of 39), and in other 5 within 0.1. On the other hand, the wrapper did significantly better in 4 cases. When we compare the performance of FortalFS with the Backward Wrapper, we find somehow similar results to what was found with the Forward Wrapper. Indeed, FortalFS outperformed significantly the Backward Wrapper 18 times and it is outperformed only 3.

By comparing FortalFS and a Random Wrapper, we are able to analyse the relative effect of the search heuristic applied by FortalFS in the selection process. Here, we decided to study the performance differences between FortalFS and our best random wrapper, that uses N^2 iterations. From Table 2 we can see that even when using a much smaller number of iterations, FortalFS($10 \cdot N$) outperforms the Random Wrapper in 27 out of the 39 cases and is outperformed in 8 others.

For detailed results and analyses of the results, including pairwise comparisons between FortalFS and all other feature selection algorithms used in these experiments regarding the number of features selected by each algorithm and the time required for selection, please see [13].

5.4 Final Remarks The results presented here show that FortalFS significantly outperformed well-known filters in our study, namely LVF, Focus and Relief. In addition, since the FortalFS, Forward, Backward and Random Wrapper algorithms share the same evaluation technique, the experiments give us a good background to compare the different search heuristics. The experimental results allow us to conclude, first, that the absence of a non-random search heuristic hurts the selection process. In addition, we could find a clear difference in terms of performance between the FortalFS search strategy and the forward and backward searches in favor of FortalFS. The most relevant conceptual difference between the FortalFS and forward and backward search strategies is the fact that the last two are greedy methods. As such, both sequential methods result in nested feature subsets in a way that features included to the final subset to be returned cannot be removed later on the process, which can cause performance problems.

6 Conclusion

In this paper, we have first developed a framework for classifying hybrid feature selection algorithms by taking into consideration both the type of filter evaluation measure and the classifier used by such methods. From the study of this framework a few facts may come to one's attention in what refers to hybrid solutions. First, one can conclude that more research in this area could bring benefits. Furthermore, it could be useful to have a general algorithm that could assume any position in the framework by employing different evaluation methods at different times.

With that in mind, we designed FortalFS, a new general hybrid solution for the feature selection problem in machine learning. The results obtained in our experiments demonstrated the power of FortalFS in selecting relevant features. The fact that FortalFS selected more accurate subsets than any other algorithm and it achieves that faster than all wrappers proves the potential of this new hybrid solution for feature selection.

References

[1] H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'91)*, volume 2, pages 547–552, Anaheim, CA, 1991. AAAI Press.

[2] J. Bala, K. DeJong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3):297–311, 1996.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[5] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[6] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis - An International Journal*, 1(3):131–156, 1997.

[7] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, pages 121–129, 1994.

[8] K. Kira and L.A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1992. Morgan-Kaufmann.

[9] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, pages 319–327, 1996.

[10] J. Reunanen. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3:1371–1382, 2003. Special Issue on Variable and Feature Selection.

[11] M. Richeldi and P. Lanzi. ADHOC: A tool for performing effective feature selection. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 102–105, 1996.

[12] M. Sebban and R. Nock. A hybrid filter/wrapper approach of feature selection using information theory. *Pattern Recognition*, (35):835 846, 2002.

[13] J.T. Souza, S. Matwin, and N. Japkowicz. *Feature Selection with a General Hybrid Algorithm.* PhD thesis, University of Ottawa, School of Information Technology and Engineering (SITE), Ottawa, ON, 2004.

[14] H. Vafaie and K. De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Proceedings of the Fourth International Conference on Tools with Artificial Intelligence*, pages 200–204, Arlington, VA, 1992.

[15] E.P. Xing, M.I. Jordan, and R.M. Karp. Feature selection for high-dimensional genomic microarray data. In *18th International Conference on Machine Learning*, pages 601–608, San Francisco, CA, 2001. Morgan Kaufmann.

[16] L. Xu, P. Yan, and T. Chang. Best first strategy for feature selection. In *Proceedings of the Ninth International Conference on Pattern Recognition*, pages 706–708. IEEE Computer Society Press, 1989.

Minimum Redundancy and Maximum Relevance Feature Selection and Recent Advances in Cancer Classification

Hanchuan Peng [12] and Chris Ding [3]

[1] Genomics Division, [2] Life Sciences Division, and [3] Computational Research Division,
Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, 94720, USA

Abstract

In many biomedical and pattern recognition applications, it is often important to consider the variable/feature selection problem, for instance, how to select a small subset out of the thousands of genes in microarray data is a key to accurate classification of phenotypes. This technique is especially useful for cancer diagnosis/classification/prediction. Widely used methods typically rank genes according to their differential expressions among phenotypes and pick the top-ranked genes. We observe that feature sets so obtained have certain redundancy and study methods to minimize it. We have proposed a minimum redundancy – maximum relevance (MRMR) feature selection framework. Genes selected via MRMR provide a more balanced coverage of the space and capture broader characteristics of phenotypes. They lead to significantly improved class predictions in extensive experiments on 6 cancer gene expression data sets: NCI, Lymphoma, Lung, Child Leukemia, Leukemia, and Colon. Improvements are observed consistently among 4 classification methods: Naïve Bayes, Linear discriminant analysis, Logistic regression and Support vector machines.

Keywords: Cancer classification, Gene selection, Gene expression analysis, Redundancy, Relevance, Dependency

1. MRMR Feature Selection Methods

For cancer diagnosis based on DNA microarray gene expression profiles, feature selection or gene marker selection is especially useful. Instead of using all available variables (features or attributes) in the data, one selectively chooses a subset of features to be used in the discriminant system. Typically, of the tens of thousands of genes in experiments, only a smaller number of them show strong correlation with the targeted phenotypes. For example, for a two-class cancer subtype classification problem, 50 informative genes are usually sufficient [13]. There are studies suggesting that only a few genes are sufficient [22][32]. Thus, computation is reduced while prediction accuracy is increased via effective feature selection. When a small number of genes are selected, their biological relationship with the target diseases is more easily identified. These "marker" genes thus provide additional scientific understanding of the problem.

Many possible feature selection methods can be roughly categorized as two general approaches: filters and wrappers [17][19]. Filter type methods select features based on the intrinsic data characteristics, which determine the relevance or discriminant powers of the selected features with regard to the target classes. Simple methods based on mutual information [4], statistical tests (*t*-test, *F*-test) have been shown to be effective [13][7][10][23]. More sophisticated methods are also developed [18][3]. Filter methods can be computed easily and very efficiently. The characteristics in the feature selection are uncorrelated to that of classifiers. Therefore they have better generalization property. In wrapper type methods, feature selection is "wrapped" around a classifier: the usefulness of a feature is directly judged by the estimated classification accuracy of specific classifier. One can often obtain a compact set of features [17][5][22][32], which give high prediction accuracy, because these features match well with the characteristics of the classification method. Wrapper methods typically require extensive computation to search the best features.

One simple way to use filters is to simply select the top-ranked genes, say the top 50 [13]. A deficiency of this approach is that the features could be correlated among themselves. For example, if gene g_i is ranked high for the classification task, other genes highly correlated with g_i are also likely to be selected. It is frequently observed [22][32] that simply combining a "very effective" gene with another "very effective" gene does not form a better feature set. One reason is that these two genes could be highly correlated. This suggests "redundancy" of feature set is one critical issue to consider. For a long time, people already realized the "*n* best features are not the best *n* features" [6], and used many implicit methods (e.g. wrappers or floating searching of filters) to remove the redundancy. Recently, there appear a few specific models [28][8][9][34][16] to minimize the redundancy in the selected features and improve the prediction performance.

One framework proposed in our earlier work is called minimum redundancy – maximum relevance (MRMR) feature selection [8]. The idea is to select features which are maximally dissimilar to each other (for example, their Euclidean distances are maximized, or they have correlation close to 0). These minimum redundancy criteria are supplemented by the usual maximum relevance criteria such as maximal mutual information with the target phenotypes (classification variable). The benefits of MRMR are two-fold. (1) With the same number of features, we expect the MRMR feature set to be more representative of the target phenotypes, therefore leading to better generalization property. (2) Equivalently, we can use a smaller MRMR feature set to effectively cover the same space as a larger conventional feature set does.

The MRMR principle is easy to implement in a variety of forms, as shown in [8]. For example, one way is to consider the mutual information of variables as the quantity of both relevance and redundancy. The mutual information I of two variables x and y is defined based on their joint probabilistic distribution $p(x,y)$ and the respective marginal probabilities $p(x)$ and $p(y)$: $I(x,y) = \sum_{i,j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$. Let h be the target classification variable, and g_i denote the ith selected feature. We define the redundancy and relevance as:

$$W_I = \frac{1}{|S|^2} \sum_{i,j \in S} I(i,j), \quad (1)$$

$$V_I = \frac{1}{|S|} \sum_{i \in S} I(h,i), \quad (2)$$

where for simplicity we have used $I(i,j)$ to represent $I(g_i,g_j)$, $I(h,i)$ for $I(h,g_i)$. $|S|$ $(= m)$ is the number of features in S.

The MRMR feature set is obtained by optimizing the conditions in Eqs.(1) and (2) simultaneously. Optimization of both conditions requires combining them into a single criterion function. The simplest combinations are Mutual Information Difference (MID) in Eq. (3) and Mutual Information Quotient (MIQ) in Eq. (4). The simple linear incremental search can be used to produce the expected number of features.

$$\max(V_I - W_I), \quad (3)$$

$$\max(V_I / W_I). \quad (4)$$

We note that another feature selection model in [34] which considers the information gain is similar to our MRMR approach. One difference is that our MRMR is a more general framework which can be implemented in many different ways [8] including but not being limited to mutual information or information gain. Particularly, in [28], for mutual information, we have also reformulated the feature selection problem as the Max-Dependency problem, which searches a subset of variables/features so that their joint distribution has the maximum statistical dependency on the target classification variable. Using information theory, we have proved that MRMR is an optimal first-order approximation for the generic Max-Dependency feature selection criterion, which is combinatorial in nature and often less robust/efficient than MRMR. In [28], we have also proposed and discussed different combinations of the MRMR method with other feature selection schemes like wrappers in forward, backward and floating search schemes. In addition, the MRMR scheme can also be used to learn the Bayesian networks [27][29][14] and applied to other model selection problems (unpublished data).

2. MRMR for Cancer Classification

Cancer classification is one typical application of feature/gene selection. In the following, we present a comprehensive investigation to answer a few questions in subsections §2.1 ~ 2.5. We consider 4 most used classifiers and 6 microarray gene expression datasets for cancer classification.

The 4 classifiers include

- Naïve Bayes (NB),
- Support Vector Machine (SVM),
- Linear Discriminant Analysis (LDA), and
- Logistic Regression (LR).

These 6 datasets are summarized in Tables 1 and 2, including

- 2 two-class datasets (Leukemia [13] and colon cancer [2]) and
- 4 multi-class datasets (NCI [30][31], Lung cancer [12], Lymphoma [1] and child leukemia [21][33]).

For the first 5 datasets, we assessed classification performance using the "Leave-One-Out Cross Validation" (LOOCV). CV accuracy provides a realistic assessment of classifiers which generalize well to unseen data. For the child leukemia data, we selected features using only the training data, and show the testing errors on the testing set in Table 3. This gives an example where the testing samples have never been met in feature selection process.

Table 1. Two-class datasets used in our experiments

DATASET	LEUKEMIA		COLON CANCER	
SOURCE	Golub et al (1999)		Alon et al (1999)	
# GENE	7070		2000	
# SAMPLE	72		62	
CLASS	CLASS NAME	# SAMPLE	CLASS NAME	# SAMPLE
C1	ALL	47	Tumor	40
C2	AML	25	Normal	22

Table 2. Multi-class datasets used in our experiments (#S is the number of samples)

DATASET	NCI		LUNG CANCER		LYMPHOMA		CHILD LEUKEMIA	
SOURCE	Ross et al (2000) Scherf et al (2000)		Garber et al (2001)		Alizadeh et al (2000)		Yoeh et al (2002) Li et al (2003)	
# GENE	9703		918		4026		4026	
# S	60		73		96		96	
# CLASS	9		7		9		9	
CLASS	CLASS NAME	# S	CLASS NAME	# S	CLASS NAME	# S	CLASS NAME	# S
C1	NSCLC	9	AC-group-1	21	Diffuse large B cell lymphoma	46	BCR-ABL	9/6
C2	Renal	9	Squamous	16	Chronic Lympho. leukemia	11	E2A-PBX1	18/9
C3	Breast	8	AC-group-3	13	Activated blood B	10	Hyperdiploid>50	42/22
C4	Melanoma	8	AC-group-2	7	Follicular lymphoma	9	MLL	14/6
C5	Colon	7	Normal	6	Resting/ activated T	6	T-ALL	28/15
C6	Leukemia	6	Small-cell	5	Transformed cell lines	6	TEL-AML1	52/27
C7	Ovarian	6	Large-cell	5	Resting blood B	4	Others	52/27
C8	CNS	5			Germinal center B	2		
C9	Prostate	2			Lymph node/tonsil	2		

The original gene expression data are continuous values. We can directly classify them using some classifiers. However, a more effective way as used in practice is to pre-process the data so that each gene has a few categorical/discrete states. Usually, this reduces the noise in the data and improves the robustness of the classification. For most experiments in the following, we discretized the observations of each gene expression variable using the respective σ (standard deviation) and μ (mean) for this gene's samples: any data larger than $\mu+\sigma/2$ were transformed to state 1; any data between $\mu-\sigma/2$ and $\mu+\sigma/2$ were transformed to state 0; any data smaller than $\mu-\sigma/2$ were transformed to state -1. These three states correspond to the over-expression, baseline, and under-expression of genes. In §2.4, we also compared different discretization schemes; partial results are summarized in Table 4.

In the following, we only show results using the mutual information based MRMR methods. For results obtained by other MRMR schemes like correlation, *F*-statistics, *t*-statistics, etc, the interested readers can refer to [9][8].

2.1 What is the Role of Redundancy Reduction?

To demonstrate the effectiveness of the MRMR approach, for the first 60 features selected using different methods, we calculated the average relevance V_I and average redundancy W_I (see Eqs.(2) and (1)) and the LOOCV error, as plotted in Fig. 1 (a)~(c). In Fig.1 (a), the relevance of MID is close to the baseline method which considers only the relevance term V_I in feature selection. The relevance of MIQ features is rather low, which seemingly suggests these MIQ features were not good. However, in Fig. 1 (b), we see that both the MID and MIQ features have low redundancy. Compared to the baseline features, the MIQ features have much lower redundancy, mainly because the quotient combination in Eq. (4) has a considerable penalty on the redundancy term. In Fig. 1 (c), the fact that the MIQ feature set leads to the least amount of LOOCV errors indicates that explicitly reducing redundancy is critical in improving the discriminative strength of the features.

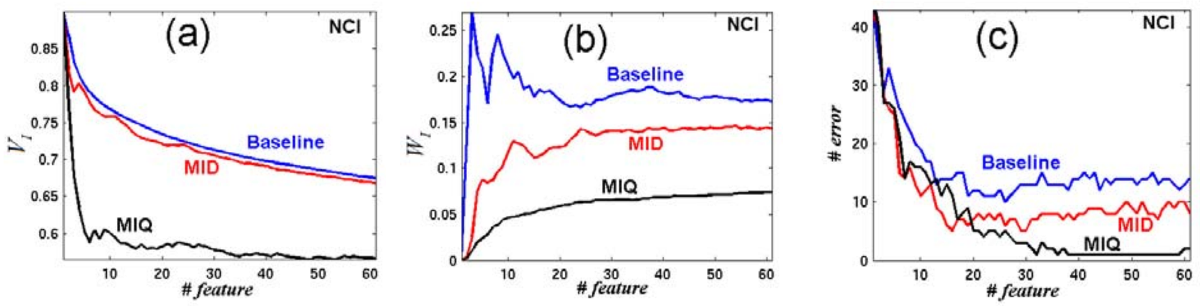
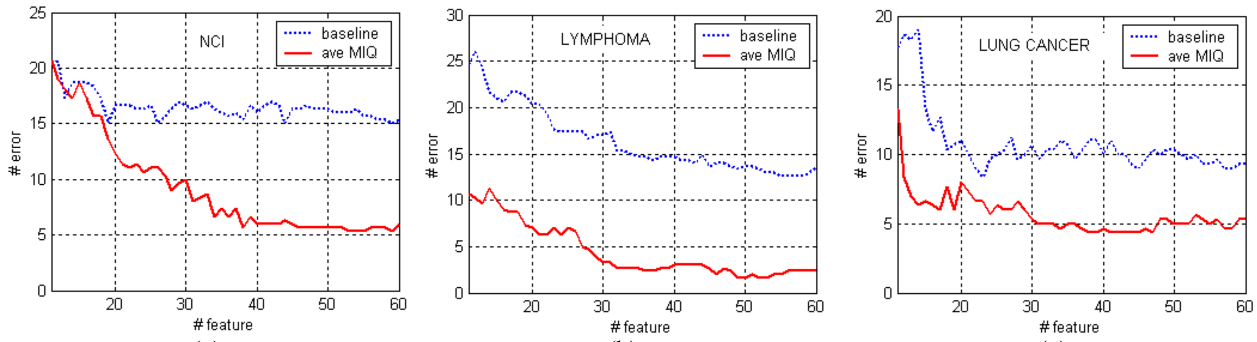


Figure 1. (a) Relevance V_I and (b) redundancy W_I for MRMR features on discretized NCI dataset. (c) The respective LOOCV errors obtained using the Naïve Bayes classifier.

2.2 Do MRMR Features Better Cover the Data Distribution Space?

It is often difficult to quantitatively measure how the "data distribution space" is covered by the selected features. However, a convenient way is to test if the selected features consistently lead to improved classification accuracy using multiple different classifiers, which have different mechanisms to classify samples in the data distribution space. In Fig. 2, we plot the average LOOCV errors of both the baseline features and MIQ features, using NB, SVM, and LDA. It is evidently that the MRMR scheme always leads to significantly lower errors, no matter which classifier or dataset is used. Besides the three plots in Fig.2, this phenomenon has been constantly observed for all other datasets we have tested. The generic improvement of the classification accuracy independent of the classifiers indicates that the MRMR features better cover the data distribution space and better characterize the most critical classification information.

Another way to address this question is to consider combination of MRMR and other feature selection methods like wrappers. The detailed discussion in [28] has led to the same conclusion as above.



(a) (b) (c)

Figure 2. Average LOOCV errors of three different classifiers, NB, SVM, and LDA on three multi-class datasets.

2.3 Do MRMR Features Generalize Well on Unseen Data?

The very low cross validation errors in Figs. 1 and 2 indicate that MRMR features generalize well on unseen data. This is appropriate for datasets where the number of samples is small. Another way to test is to select features using a training set and predict the class labels using a separate testing set. We considered the Child Leukemia data, where there are 215 training samples and 112 testing samples. The results are shown in Table 3. MRMR methods lead to evidently lower errors than the baseline method. This suggests that

MRMR is largely independent of the set of data (i.e. the whole set of data or the training set only) used to select features.

2.4 What is the Relationship of MRMR Features and Various Data Discretization Schemes?

How the discretization method affects the feature selection results? We tested many different discretization parameters to transform the original continuous gene expression data to either 2-state or 3-state categorical variables. The features consequently selected via MRMR always outperform the respective features selected using baseline method. For simplicity, we only show two exemplary results for the NCI and Lymphoma data sets using the SVM classifier. The data were binarized using the mean value of each gene as the threshold of that gene's samples. As illustrated in Table 4, we see that MRMR features always lead to better prediction accuracy than the baseline features. For example, for NCI data, 48 baseline features lead to 13 errors, whereas MIQ features lead to only 2 errors (3% error rate). For lymphoma data, the baseline error is never less than 10, whereas the MIQ features in most cases lead to only 1 or 2 errors (1~2% error rate). These results are consistent with those shown above. This demonstrates that under different discretization schemes the superiority of MRMR over conventional feature selection schemes is prominent.

Table 3. Child Leukemia data (7 classes, 215 training samples, 112 testing samples) testing errors. M is the number of features used in classification.

Classifier	M / Method	3	6	9	12	15	18	24	30	40	50	60	70	80	90	100
LDA	Baseline	55	47	46	38	34	27	19	28	22	19	15	14	11	8	8
	MID	50	43	32	29	30	29	22	15	13	10	10	9	7	8	9
	MIQ	43	43	34	27	23	21	18	16	11	11	6	4	6	6	4
SVM	Baseline	56	55	49	37	33	33	27	35	29	30	23	20	18	14	13
	MID	45	42	33	33	25	25	29	25	26	22	20	13	10	12	9
	MIQ	38	30	34	33	27	26	24	21	14	15	17	10	7	11	9

Table 4. LOOCV testing results (#error) for binarized NCI and Lymphoma data using SVM classifier.

Data Sets	M / Method	3	6	9	12	15	18	21	24	27	30	36	42	48	54	60
NCI	Baseline	34	25	23	25	19	17	18	15	14	12	12	12	13	12	10
	MRMR (MIQ)	35	22	22	16	12	11	10	8	5	3	4	4	2	2	3
Lymphoma	Baseline	58	52	44	39	44	17	17	14	16	13	11	10	13	10	12
	MRMR (MIQ)	24	17	7	8	4	2	1	2	4	3	2	2	2	2	2

2.5 Comparison with Other Work

Results of similar class prediction on microarray gene expression data obtained by others are listed in Table 5. For NCI, our result of LOOCV error rate is 1.67% using NB, whereas Ooi & Tan [26] obtained 14.6% error rate. On the 5-class subset of NCI, Nguyen & Rocke [25] obtained 0% rate, which is the same as our NB results on the same 5-class subset.

For Lymphoma data, our result is LOOCV error rate of 1%. Using 3 classes only, Nguyen & Rocke [25] obtained 2.4%; on the same 3 classes, our LDA result is 0% error rate.

For child leukemia data, Li et al [21] obtained 5.36% error rate using collective likelihood. In our best case, the MRMR features lead to the 2.68% error rate.

The Leukemia data* is a most widely studied dataset. Using MRMR feature selection, we achieve 100% LOOCV accuracy for every classification methods. Furey et al [11] obtained 100% accuracy using SVM, and Lee & Lee [20] obtained 1.39% error rate.

For Colon data*, our result is 6.45% error rate, which is the same as Nguyen & Rocke [25] using PLS. The SVM result of [11] is 9.68%.

* Many classification studies have used Leukemia and Colon datasets. For simplicity, we only list two for each dataset in Table 5.

Table 5. Comparison of the best results (lowest error rates in percentage) of the baseline and MRMR features. Also listed are results in literature (the best results in each paper). [a] Ooi & Tan, using the genetic algorithm [26]. [b] Nguyen and Rocke [25] used a 5-class subset of NCI dataset and obtained 0% error rate; using the same 5-class subset, our NB achieves also 0% error rate. [c] Nguyen & Rocke used 3-class subset in lymphoma dataset and obtain 2.4% error rate. Using the same 3 classes, our NB led to zero errors. [d] Li et al, using prediction by collective likelihood [21]. [e] Furey et al, using SVM [11]. [f] Lee & Lee, using SVM [20]. [g] Nguyen & Rocke, using PLS [24].

Data	Method	NB	LDA	SVM	LR	Literature
NCI	Baseline	18.33	26.67	25.00	--	14.63 [a]
	MRMR	1.67	13.33	11.67	--	5-class: 0 [b], 0 [b]
Lymphoma	Baseline	17.71	11.46	5.21	--	3-class: 2.4 [c], 0 [c]
	MRMR	3.13	1.04	1.04	--	
Lung	Baseline	10.96	10.96	10.96	--	--
	MRMR	2.74	5.48	5.48	--	
Child Leukemia	Baseline	29.46	7.14	11.61	--	5.36 [d]
	MRMR	13.39	2.68	6.25	--	
Leukemia	Baseline	0	1.39	1.39	1.39	0 [e]
	MRMR	0	0	0	0	1.39 [f]
Colon	Baseline	11.29	11.29	11.29	11.29	9.68 [e]
	MRMR	6.45	8.06	9.68	9.68	6.45 [g]

3. Discussions

In this paper we emphasize the redundancy issue in feature selection. Our feature selection framework, the minimum redundancy – maximum relevance (MRMR) optimization approach, literally minimizes the redundancy in the selected features. Our experiments on 6 gene expression datasets using Naïve Bayes, Linear discriminant analysis, Logistic regression and SVM class prediction methods, show that MRMR feature sets consistently outperform the baseline feature sets based solely on maximum relevance.

The main benefit of MRMR feature set is that by reducing mutual redundancy within the feature set, these features capture the class characteristics in a broader scope. Features selected within the MRMR framework are independent of class prediction methods, and thus do not directly aim at producing the best results for any prediction method. The fact that MRMR features improve prediction for all four methods we tested confirms that these features have better generalization property. This also implies that with fewer features the MRMR feature set can effectively cover the same class characteristic space as more features in the baseline approach.

For biologists, sometimes the redundant features might also be important. A Bayesian clustering method can be developed to identify the highly correlated gene clusters. Then, representative genes from these clusters can be combined to produce good prediction results. We find that our MRMR approach is also consistent with the Bayesian network learning and variable selection method in [14][27][29] using the conditionally independence constraints.

Acknowledgements

This work is partly supported by US Department of Energy, Office of Science (MICS office and LBNL LDRD) under the contract DE-AC03-76SF00098. Hanchuan Peng is also supported by National Institutes of Health (NIH), under contract No. R01 GM70444-01.

References

[1] Alizadeh, A.A., et al. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling, *Nature*, *403*, 503-511.

[2] Alon, U., Barkai, N., Notterman, D.A., et al. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, *PNAS*, *96*, 6745-6750.

[3] Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., & Yakhini, Z. (2000), Tissue classification with gene expression profiles, *J Comput Biol*, *7*, 559--584.

[4] Cheng, J., & Greiner, R. (1999). Comparing Bayesian network classifiers, *UAI'99*.

[5] Cherkauer, K.J., & J. W. Shavlik, J.W. (1993). Protein structure prediction: selecting salient features from large candidate pools, *ISMB 1993*, 74-82.

[6] Cover, T.M., "The best two independent measurements are not the two best," IEEE Trans. Systems, Man, and Cybernetics, vol. 4, pp. 116-117, 1974.

[7] Ding, C. (2002). Analysis of gene expression profiles: class discovery and leaf ordering, *RECOMB 2002*, 127-136.

[8] Ding, C., and Peng, H.C., "Minimum redundancy feature selection from microarray gene expression data," Proc. 2nd IEEE Computational Systems Bioinformatics Conference, pp.523-528, Stanford, CA, Aug, 2003.

[9] Ding, C., and Peng, H.C., "Minimum redundancy feature selection from microarray gene expression data,"
Journal of Bioinformatics and Computational Biology, Vol. 3, No. 1, 2005. (In press)

[10] Dudoit, S., Fridlyand, J., & Speed, T. (2000). Comparison of discrimination methods fro the classification of tumors using gene expression data, *Tech Report 576*, Dept of Statistics, UC Berkeley.

[11] Furey,T.S., Cristianini,N., Duffy, N., Bednarski, D., Schummer, M., and Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data, *Bioinformatics*,16, 906-914.

[12] Garber, M.E., Troyanskaya, O.G., et al. (2001). Diversity of gene expression in adenocarcinoma of the lung, *PNAS USA*, *98*(*24*), 13784-13789.

[13] Golub, T.R., Slonim, D.K. et al, (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, *Science*, *286*, 531-537.

[14] Herskovits, E., Peng, H.C., and Davatzikos, C., "A Bayesian morphometry algorithm," IEEE Transactions on Medical Imaging, 24(6), pp.723-737, 2004.

[15] Jaakkola, T., Diekhans, M., & Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies, *ISMB'99*, 149-158.

[16] Jaeger,J., Sengupta,R., Ruzzo,W.L. (2003) Improved Gene Selection for Classification of Microarrays, *PSB'2003*, 53-64.

[17] Kohavi, R., & John, G. (1997). Wrapper for feature subset selection, *Artificial Intelligence*, *97*(*1-2*), 273-324.

[18] Koller D., & Sahami, M. (1996). Toward optimal feature selection, *ICML'96*, 284-292.

[19] Langley, P. (1994). Selection of relevant features in machine learning, *AAAI Fall Symposium on Relevance*.

[20] Lee, Y., and Lee, C.K. (2003). Classification of multiple cancer types by multicategory support vector machines using gene expression data, *Bioinformatics*, 19, 1132-1139.

[21] Li, J., Liu, H., Downing, JR, Yeoh, A, and Wong, L. (2003) "Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients," Bioinformatics. 19, pp.71-78.

[22] Li,W., & Yang,Y. (2000). How many genes are needed for a discriminant microarray data analysis?, *Critical Assessment of Techniques for Microarray Data Mining Workshop*, 137-150.

[23] Model, F., Adorján, P., Olek, A., & Piepenbrock, C. (2001). Feature selection for DNA methylation based cancer classification, *Bioinformatics*, 17, S157-S164.

[24] Nguyen, D.V., & Rocke, D. M. (2002). Tumor classification by partial least squares using microarray gene expression data, *Bioinformatics*, 18, 39-50.

[25] Nguyen, D.V., & Rocke, D.M. (2002). Multiclass cancer classification via partial least squares with gene expression profiles, *Bioinformatics*, 18, 1216-1226.

[26] Ooi, C. H., and Tan, P. (2003). Genetic algorithms applied to multi-class prediction for the analysis of gene expression data, *Bioinformatics,* 19, 37-44.

[27] Peng, H.C., and Long, F.H., "A Bayesian learning algorithm of discrete variables for automatically mining irregular features of pattern images," Proc of Second International Workshop on Multimedia Data Mining (MDM/KDD'2001) in conjunction with ACM SIG/KDD2001, pp.87-93, San Francisco, CA, USA, 2001.

[28] Peng, H.C., and Long, F.H., "An efficient max-dependency algorithm for gene selection," 36th Symposium on the Interface: Computational Biology and Bioinformatics, Baltimore, Maryland, May 26-29, 2004.

[29] Peng, H.C., Herskovits, E, and Davatzikos, C., "Bayesian clustering methods for morphological analysis of MR images," Proc. of 2002 IEEE Int. Symposium on Biomedical Imaging: From Nano to Macro, pp.485-488, Washington, D.C., USA, July, 2002.

[30] Ross, D.T., Scherf, U., et al. (2000). Systematic variation in gene expression patterns in human

cancer cell lines, *Nature Genetics*, *24*(*3*), 227-234.

[31] Scherf, U., Ross, D.T., et al. (2000). A cDNA microarray gene expression database for the molecular pharmacology of cancer, *Nature Genetics*, 24(3), 236-244.

[32] Xiong, M., Fang, Z., & Zhao, J. (2001). Biomarker identification by feature wrappers, Genome Research, 11, 1878-1887.

[33] Yeoh, A., …, Wong, L., and Downing, J., (2002), "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling," Cancer Cell, 1, pp.133-143.

[34] Yu, L., and Liu, H., "Redundancy Based Feature Selection for Microarry Data", SIGKDD, KDD 2004, August, 22 - 25, 2004. Seattle, Washington.

Gene Expression Analysis of HIV-1 Linked p24-specific CD4+ T-Cell Responses for Identifying Genetic Markers

Sanjeev Raman and Carlotta Domeniconi
Information and Software Engineering Department
George Mason University
sraman@gmu.edu carlotta@ise.gmu.edu

Abstract

The Human Immunodeficiency Virus (HIV) presents a complex knot for scientists to unravel. After initial contact and attachment to a cell of the immune system (e.g. lymphocytes, monocytes), there is a cascade of intracellular events. The endproduct of these events is the production of massive numbers of new viral particles, death of the infected cells, and ultimate devastation of the immune system. HIV is an epidemic and a crisis in many continents [1]. Since there are many variations of the virus and differences in people's genetic make-up, rapid diagnosis and monitoring of tailored treatments are essential for future medicine. To combat this problem, microarray technology can perform a single scan on thousands of genes. However, without a proper research design and data mining techniques, the results from such a technology can be very skewed. Thus, using a normalized, clean dataset (time-series) from the CD4+ T-cell line CEM-CCRF, we designed and implemented hierarchical clustering and pattern-based clustering algorithms to identify specific cellular genes influenced by the HIV-1 viral infection. This research can contribute to the HIV Pharmacogenomics field by confirming HIV genetic markers, which would lead to rapid diagnosis and customized treatments.

Keywords: pattern-based clustering, hierarchical clustering, HIV, gene expression analysis, genetic markers.

1. Introduction

Since viruses (i.e. human immunodeficiency virus type 1 - HIV-1) can impact a diverse set of host cell's biochemical processes, many of these interactions can be characterized by changes in cellular mRNA levels that could depend on both the stage of infection and the biological stage state of the infected cell [2]. For example, viral infection induces the interferon antiviral response, modulates the cell's transcriptional, translational, and trafficking machinery. Thus, the recent emergence of high-density DNA arrays (microarrays and oligonucleotide chips) has revolutionized gene expression studies by providing a means to measure mRNA levels for thousands of genes simultaneously [3].

In this paper we conducted a gene expression analysis, which is a novel approach to identifying and profiling genes related to the pathology and responsiveness of a potential treatment. In the case of HIV-1, where the infection is worldwide and the subtypes are many, measuring the efficacy of a potential treatment in distinct populations from a molecular level is essential. Since people can have different responses to treatments based on their genetic make-up, the Food and Drug Administration is going to mandate pharmacogenomic studies to be submitted with drug submission research [4].
Thus, we focused on two main objectives:

1. Researching and discussing the various techniques and approaches for gene expression analysis.

2. Identifying and confirming global genetic markers for HIV-1 by designing and implementing data mining algorithms.

Our approach utilized two proven computational techniques: hierarchical clustering and pattern-based clustering. All the data analysis will be based on time series data and genes from the CD4+ T-cell line CEM-CCRF in order to identify specific cellular genes influenced by HIV-1 viral infection. The details of the research design are discussed in subsequent sections. Prior research has been conducted in this field, however, the research was done when the technology to do the data analysis was very new to the market (1998 and 1999) and thus, the analysis was very broad. This is because the focus was on classes of genes. In contrast, in this work our objective is the identification of potential global genetic markers [5].

1.1 Motivation and Contribution

The results from this study can give great insight of how to quickly measure the effectiveness of a treatment according to a person's genetic make-up and what specific genes are important in the regulation of HIV/AIDS. This study will help to confirm previous results from a molecular level and contribute to the overall knowledge domain of pharmacogenomic-HIV research [6], which will eventually lead to customized diagnosis and treatment of the disease.

2. Background and Related Work

HIV is a retrovirus and thus, contains a genome composed of two copies of single stranded RNA housed in a cone-shaped core surrounded by a membrane envelope. A transfer RNA is located near the 5' end of each RNA and serves as an initiation site for reverse transcription. Viral enzymes housed in the core include reverse transcriptase, protease, and integrase. The envelope proteins consist of a transmembrane portion (gp41) and a surface molecule (gp120), which is the attachment site to the receptor on the host cell. Like all retroviruses, HIV-1 genome encodes for gag, pol, and env. However, HIV-1 also contains six accessory gene products that are somewhat essential for HIV replication and reproduction (tat, rev, vif, vpu, vpr, and nef) [7].

Microarray expression analysis has become one of the most widely used functional genomics tools. Efficient application of this technique requires the development of robust and reproducible protocols. This process involves several aspects of optimization such as Polymearse Chain Reaction amplification of target cDNA clones, microarray printing, probe labeling and hybridization, and developed strategies for data normalization and analysis [8].

Efficient expression analysis using microarrays requires the development and successful implementation of a variety of laboratory protocols and strategies for fluorescence intensity normalization. The process of expression analysis can be broadly divided into three stages [9]: (1) Array Fabrication; (2) Probe Preparation and Hybridization; and (3) Data Collection, Normalization and Analysis.

The genome of an organism is the genetic code that regulates the expression of various features and functions of the organism. This regulation is brought about by the co-ordination of various genes in the genome. These genes communicate with each other to trigger or suppress the expression of each other. A typical experiment on the gene expression would therefore have to take into account the simultaneous observation of these genes.

2.1 Hierarchical Clustering

Hierarchical clustering is by far the most popular method to cluster microarray data. There are two types of hierarchical clustering – agglomerative and divisive. Agglomerative clustering takes an entity (i.e. a gene) as a single cluster to start off with and then builds bigger and bigger clusters by grouping similar entities together until the entire dataset is encapsulated into one final cluster. Divisive hierarchical clustering works the opposite way around – the entire dataset is first considered to be one cluster and is then broken down into smaller and smaller subsets until each subset consists of one single entity. The sequence of clustering results is represented by a hierarchical tree, called a dendogram, which can be cut at any level to yield a specific number of clusters [10].

The agglomerative approach is most commonly used in microarray analyses. The reason is that divisive clustering is more computationally expensive when it comes to making decisions in dividing a cluster in two, given all possible choices. However, the divisive approach retains the "super structure" of the data. This means that one can confidently say that the root or "upper levels" of the dendogram are highly representative of the original structure of the data. Although, this does not mean that the agglomerative approach is not just as robust [10]. We focused on the agglomerative approach.

The basic rules for agglomerative hierarchical clustering are as follows [11]:

1. Derive a vector representation for each entity (i.e. gene expression values for each experiment make up the vector elements for a specific gene);

2. Compare every entity with all other entities by calculating a distance. Input that distance into a matrix. Calculation of the distance depends on:

a. the linkage method (distance between clusters) being implemented;
b. the actual distance measure used;

3. Group closest two entities (or clusters) together (which makes a new cluster) and go back to step 2, considering the new cluster as a single entity, recalculate distances between entities and cluster closest entities together. Step 2 should be repeated until all entities are contained within one big cluster.

The distance between clusters is usually computed in one of three different ways: *Single linkage* is the minimum distance between a point in one cluster and a point in the other cluster; *average linkage* is the average of the distances between points in one cluster and points in the other cluster; *complete linkage* is the largest distance between a point in one cluster and a point in the other cluster. Thus, an agglomerative hierarchical clustering approach can be implemented using, for example, the Euclidean distance measure and the average linkage method.

2.2 Pattern Based Clustering

Pattern-based clustering (or p-clustering) groups a set of objects based on their coherent trend in a subset of dimensions. This differs slightly from subspace clustering as subspace uses global distance/similarity measures, which may not be able to detect coherent trends. There are two distinct features of pattern-based clustering: there is no global defined similarity/distance measure, and clusters may not be exclusive. When using pattern-based analysis, subsets of genes whose expression levels change coherently under a subset of conditions are identified. This analysis can be critical in revealing the significant connections in gene regulatory networks.

There are two issues to be concerned with when performing pattern-based clustering. Issue one is that there can be many pattern-based clusters, thus maximal pattern-based clusters must be determined. Second, the methodology to mine maximal pattern-based clusters must be efficient [12]. Traditionally, a pattern score is used to calculate the similarity between two objects. For example, [12] defines the pattern score of two objects r_x, r_y on two attributes a_u, a_v as follows:

$$pScore\left(\begin{bmatrix} r_x.a_u & r_x.a_v \\ r_y.a_u & r_y.a_v \end{bmatrix}\right) = \left\|\left(r_x.a_u - r_y.a_u\right) - \left(r_x.a_v - r_y.a_v\right)\right\|$$

Also, a threshold is established. For example, for any objects $r_x, r_y \in R$ and any attributes $a_u, a_v \in D$, in [12] it is required:

$$pScore\left(\begin{bmatrix} r_x.a_u & r_x.a_v \\ r_y.a_u & r_y.a_v \end{bmatrix}\right) \le \delta \quad (\delta \ge 0)$$

In regards to maximal pClusters, if (R, D) is a δ-pCuster (that is, all pairwise objects in R have a $pScore \le \delta$ with respect to attributes in D), then every cluster (R', D'), where $R' \subseteq R$ and $D' \subseteq D$, is a δ-pCuster (anti-monotonic property). That is, a large pCluster is accompanied with many small clusters. Therefore, the idea is to mine only the maximal pClusters. A δ-pCuster is maximal if there exists no proper super cluster that is a δ-pCuster [12].

3. Research Design and Methodology

As mentioned in the previous section, gene expression analysis can be divided into sequential stages: array fabrication, probe preparation and hybridization, data collection, normalization, and analysis. In this section, we explain and describe in detail the specific design and techniques needed to perform the gene expression analysis of HIV-1 linked p24-specific CD4+ T-cell responses for identifying genetic markers.

The human immunodeficiency virus type 1 (HIV-1) infection alters the expression of host cell genes at both the mRNA and protein levels. To obtain a more comprehensive view of the global effects of HIV infection of CD4-positive T-cells at the mRNA level, we analyze a cDNA microarray dataset generated from the University of California, San Diego [5]. We perform p-clustering and hierarchical clustering analysis on mRNA expressions of approximately 6800 genes. These mRNA expressions were monitored at eight time points [0.5h, 2h, 4h, 8h, 16h, 24h, 48h, 72h] from a CD4+ T-cell line (CEM-GFP) during HIV-1 infection. The CEM-GFP cells were inoculated with HIV-1 at a multiplicity of infection of 0.5, an inoculum sufficient to ensure that every cell is contracted by virus particles. Aliquots of cells were obtained as described above. A mock infection served as a control at each time point, essentially replacing the volume of viral input by an equivalent volume of culture medium from uninfected cells. Each sample was tested on two chips and the average was taken. Normalization for this dataset was done using global normalization and scaling. The objective is to identify a specific set of universal genes that can be used as genetic markers for measuring the effectiveness of a potential treatment based on time series patterns and levels consistently changing more than 1.5-fold. A fold is defined mathematically as $\log_2(Cy5/Cy3)$, where typically, $Cy5$ represents treated/infected samples and $Cy3$ represents untreated/uninfected samples. Thus, for example, if the log ratio is 2.0 for a given condition, then this means the gene is over-expressed by 2 fold, and is usually represented with a red light indicator in the visual output for that spot from the microarray chip. Vise versa, if the log ratio is -2.0, then this means the gene is under-expressed by 2 fold, and is usually represented with a green light indicator in the visual output for that spot from the microarray chip. Therefore, the expression values will be clustered by trends over a period of time and by fold regulation [13].

3.1 Data Normalization and Tools

We implemented a normalization technique based on fluorescence intensities. This is a popular method based on total intensity normalization, where each fluorescent intensity value is divided by the sum of all the fluorescent intensities [14].

The normalization, cleaning, and analysis of the data take place in Oracle 10*i*. Oracle10*i* Data Mining simplifies the process of normalizing and extracting intelligence from large amounts of data. It eliminates off-loading vast quantities of data to external special-purpose analytic servers for data mining and scoring. With Oracle 10*i* Data Mining, all the data mining functionality is embedded in Oracle10*i* Database, so the data, data preparation, model building, and model scoring activities remain in the database. Because Oracle 10*i* Data Mining performs all phases of data mining within the database, each data mining phase results in significant improvements in productivity, automation, and integration. Significant productivity enhancements are achieved by eliminating the extraction of data from the database to special purpose data mining tools and the importing of the data mining results back into the database. These improvements are notable in data preparation, which often can constitute as much as 80% of the data mining process. With Oracle 10*i* Data Mining, all the data preparation can be performed using standard SQL manipulation and data mining utilities within Oracle9*i* Data Mining [15].

3.2 Preprocessing

We performed hierarchical clustering, p-clustering, and plotting analysis on mRNA expressions of approximately 6800 genes using the cDNA microarray dataset generated from the University of California, San Diego [5]. It is important to note the difference between p-clustering and subspace clustering. These mRNA expressions were monitored at eight time points [0.5h, 2h, 4h, 8h, 16h, 24h, 48h, 72h] from a CD4+ T-cell line (CEM-GFP) during HIV-1 infection. The CEM-GFP cells were inoculated with HIV-1 at a multiplicity of infection of 0.5, an inoculum sufficient to ensure that every cell is contracted by virus particles. Aliquots of cells were obtained as described above. A mock infection served as a control at each time point, essentially replacing the volume of viral input by an equivalent volume of culture medium from uninfected cells. Each sample was tested on two chips and the average was taken. Normalization for this dataset was done using global normalization and scaling. Other cleaning techniques were applied to the dataset, as described below:

1. % Present >= X. This removes all genes that have missing values in greater than (100 - *X*) percent of the columns. In our case, X was 90.
2. SD (Gene Vector) >= X. This removed all genes that have standard deviations of observed values less than *X*. In our case, X was 2.0.
3. At least X Observations abs(Val) >= Y. This removes all genes that do not have at least *X* observations with absolute values greater than *Y*. We require at least 8 observations with absolute value greater than 2.0.
4. MaxVal-MinVal >= X. This removes all genes whose maximum minus minimum values are less than *X*. In our case, X was 2.0.

For cleaning technique 1, we set X = 90 because if a gene had a missing value for just one column, this would be very significant since there are only eight time points. So, by setting 90 as a threshold, we select only the genes with values for all columns, which leads to more accurate data analysis.
For cleaning technique 2, X=2.0 because in order to do fairly accurate data analysis, the gene expression values should not be too small. Otherwise, results could be skewed. Thus, 2.0 would serve as a fair standard deviation tolerance to delete genes that could potentially affect the final results.
For cleaning technique 3, again, to avoid skewing of the results because of the gene expression values

being too small, we made sure every gene included in the analysis had values greater than 2 for each and every time point.

For cleaning technique 4, it was more efficient to delete genes that would be of no significance for the analysis. Setting X=2 as the difference between the maximum and minimum values was an easy way to dismiss genes (less than or equal to X) that were of no significance.

After normalization and cleaning of the data, 167 genes out of 6823 genes (2.5%) were deleted from the dataset. Then, the data was organized into two smaller datasets for analysis. The first dataset was the mock infection and the second dataset was the actual infection.

3.3 Analysis and Results

Discovering co-expressed genes and coherent expression patterns in gene expression data is an important data analysis task in bioinformatics research and biomedical applications. It is often an important task to identify the co-expressed genes and the coherent expression patterns from the gene expression data. A group of *co-expressed genes* are the ones with similar expression profiles, while a *coherent expression pattern* characterizes the common trend of expression levels for a group of co-expressed genes. In practice, co-expressed genes may belong to the same or similar functional categories and indicate co-regulated families. Coherent expression patterns may characterize important cellular processes and suggest the regulating mechanism in the cells [16].

To find co-expressed genes and discover coherent expression patterns, many gene clustering methods have been proposed [12]. In our case, each cluster was considered as a group of co-expressed genes. The coherent expression pattern was identified via a comparative analysis of the percentage increase/decrease of each gene. Finally, the mean (or centroid) of the expression profiles of the genes in the resulting sub-clusters gives the corresponding coherent expression pattern. While clustering algorithms have been shown useful to identify co-expressed gene groups and discover coherent expression patterns, due to the specific characteristics of gene expression data and the special requirements from the biology domain, several great challenges for clustering gene expression data remain [17].

An interesting phenomenon in gene expression data sets is that *groups of co-expressed genes may be highly connected by a large amount of "intermediate" genes*. Technically, two genes g_x and g_y that have very different expression profiles in a data set may be bridged by a series of intermediate genes such that each two consecutive genes on the bridge have similar profiles. An empirical study has shown that such "bridges" are common in gene expression data sets. The high connectivity in the gene expression data raises a challenge: *It is often hard to find the (clear) borders among the clusters*. Many existing clustering methods use one of the following two strategies. On the one hand, the data set is decomposed into numerous small clusters. While some clusters consist of groups of biologically meaningful co-expressed genes, many clusters may consist of only intermediate genes. Since there is no biologically meaningful criteria (e.g., size, compactness) to rank the resulted clusters, it may take a lot of effort to examine which clusters are meaningful groups of co-expressed genes. On the other hand, an algorithm may form several large clusters. Each cluster contains both the co-expressed genes and a large amount of intermediate genes. However, those intermediate genes may mislead the centroids of the clusters into going astray. The centroids then no longer represent the true coherent patterns in the groups of co-expressed genes [17].

In a gene expression data set, there are usually multiple groups of co-expressed genes as well as the corresponding coherent patterns. Moreover, there is typically a hierarchy of co-expressed genes and coherent patterns in a gene expression data set. At the high levels of the hierarchy, large groups of genes approximately follow some "rough" coherent expression patterns. At the low levels of the hierarchy, the large groups of genes break into smaller subgroups. Those smaller groups of co-expressed genes follow some "fine" coherent expression patterns, which inherit some characteristics from the "rough" patterns, and add some distinct characteristics [17].

In our analysis, after cleaning the data, we proceeded to use an agglomerative hierarchical clustering approach based on average linkage [16] to hierarchically cluster the genes. Then we examined the clustered results and identified a cross-sectional point to start the coherent analysis. The cross-sectional point was three levels in from the root level. This level was chosen because it was the last level that had sibling nodes that covered all the genes analyzed from the microarray. This approach proved to be more effective and accurate than just simply taking the mean of each hierarchical cluster because not every gene which displays a similar pattern is necessary similar in function.

At that point, we developed and implemented an algorithm similar to the p-clustering concept. When examining all the sibling nodes (starting at 3 levels in), we computed the percentage increase/decrease between adjacent time points for each gene in each of the sibling nodes, and computationally compared such percentage variations for all the genes in that cluster. Using a 10% dis-similarity tolerance between the percentages, we were able to computationally reclassify the genes into sub-clusters based on pattern similarity. More formally, we can represent a gene as a eight dimensional vector. Let $g_x = (g_{x1}, \cdots, g_{x8}), g_y = (g_{y1}, \cdots, g_{y8})$ be such two gene vectors. We define the *pSiminarity* between the *ith* and *(i+1)th* components of two genes g_x and g_y as follows:

$$pSimilarity(g_x, g_y, i) \cong$$
$$\left|(((g_{xi+1} - g_{xi})/g_{xi}) \times 100) - (((g_{yi+1} - g_{yi})/g_{yi}) \times 100)\right|$$

The above equation computes the (absolute value of the) difference between the percentage decrease/increase between the corresponding sequential time points of two genes g_x and g_y. Genes that are under or equal to a 10 percent dissimilarity for all 7 (8 time points) percentages are clustered in the same sub-group. That is:

$$g_x, g_y \in same\ \ cluster$$
$$if \quad pSimilarity(g_x, g_y, i) \leq 10 \quad \forall i = 1, \cdots, 7$$

In the example below, g_x is constant through out the loop and g_y represents the gene that is being compared to g_x from the same hierarchical cluster at level 3. Thus, the loop continues until all genes from that cluster is computationally compared to gene g_x.

```
1oopcount = 1
While (loopcount <= X)     //X = the number of genes in the given hierarchical cluster at level 3
{
        if
(|(((g_{xi+1} - g_{xi})/g_{xi}) × 100) - (((g_{yi+1} - g_{yi})/g_{yi}) × 100)| ≤ 10)
∀i = 1,⋯,7

        then     cluster=true;

        else     cluster=false;
loopcount = loopcount + 1;
}
```

After g_x was compared, and all similar genes were clustered with g_x, the next non-clustered gene replaced g_x and was compared to all other non-clustered genes. The loopcount was also modified to the number of non-cluster genes left. This cycle continued until all genes belonged to disjoint clusters. For clusters that visually displayed ‘rough’ patterns (i.e., when the majority of genes in the cluster were close to the 10% dissimilarity threshold), we re-ran the algorithm to generate more ‘fine’ sub-clusters using a higher degree for the tolerance (i.e. 5%). Once all the ‘rough’ patterns were refined, we took the average for each time point for all the genes in each cluster to represent the pattern trend for that cluster. Thus, when each cluster was plotted, it was very easy to decipher which clusters had potential genetic markers for HIV-1/AIDS because they exhibited sharp pattern trends.

After identifying a set of genes as potential genetic markers from the lower level clusters, we traced them back to the original dendogram to see if they were similar based on expression profiles, which would indicate similar functionality of these genes as well. We also used the public genome database to help confirm the results, which are discussed below.

From the analysis, we were able to single out individual genes that would serve as potential genetic markers by breaking down the clusters into smaller sub-clusters using the algorithm described. The reason is that we were strictly looking for genetic markers as in genes that show a significant, constant change in their expression profile when exposed to the virus. Whether this behavior was triggered by other genes is irrelevant because we are not looking for a deep understanding of the gene other than knowing at a basic level why the gene could have been affected. The use of the public genome database is a sure way of confirming the results. The accession number for the first gene is *J04423*. Because this gene was of high interest during the microarray experiment, six different probe sets were used with each resulting in a significant fold regulation by 72 hours. The probe that yielded the highest fold increase had an upfold regulation of 1.85 ($\log_2$ (25448.1/7187.9)) at 72 hours. The next gene - accession number *XO3453* - was analyzed with two different probe sets. The probe that yielded the highest fold regulation had an upfold regulation of 1.55 ($\log_2$ (65440.2/22487.1)) at 72 hours. The other four genes (accession numbers stated below) of interest were only analyzed using one probe set and yielded the following results:

- *U14573*: upfold regulation of 1.5 (log$_2$ (95340.6/34555.2)) at 72 hours
- *AB000905*: upfold regulation of 1.5 (log$_2$ (210.2.9/76)) at 72 hours
- *D43951*: upfold regulation of 2.45 (log$_2$ (111.6/20.7)) at 72 hours
- *M21388*: upfold regulation of 1.5 (log$_2$ (28749.2/10162.9)) at 72 hours

In Figures 1-6, the pink line represents infected CEM-GFP cells, while the blue line represents non-infected CEM-GFP cells. The graphs show the expression value for each time point and the over all pattern for all the time points for the given gene.

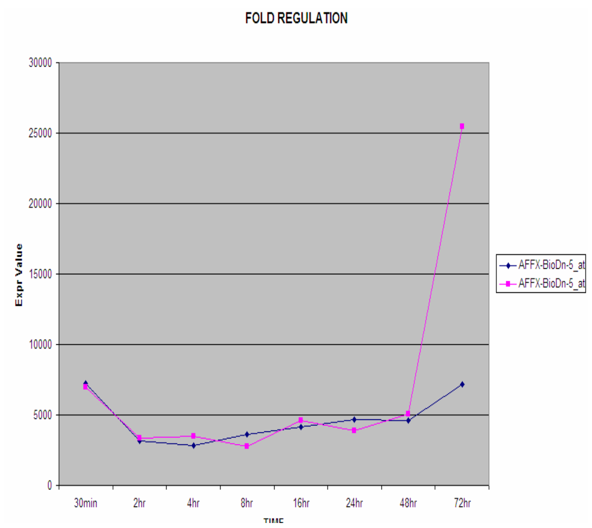


Figure 1: *J04423*

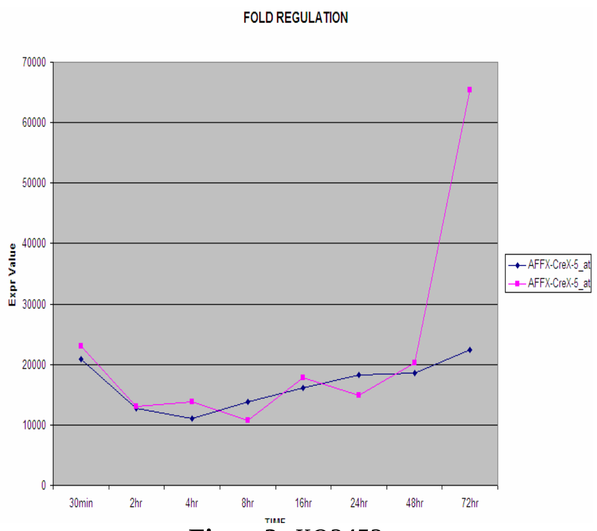


Figure2: *XO3453*

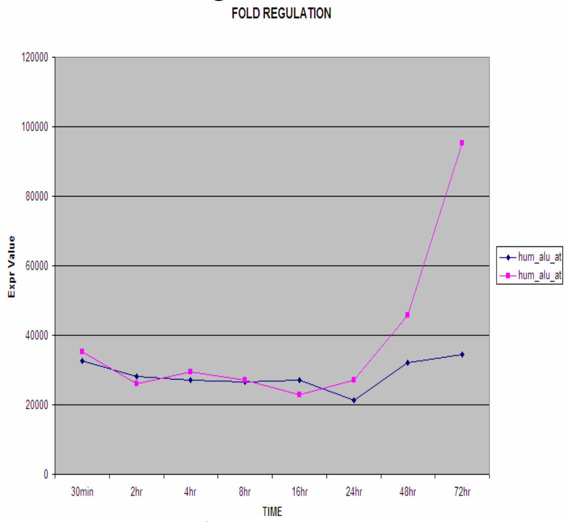


Figure 3: *U14573*

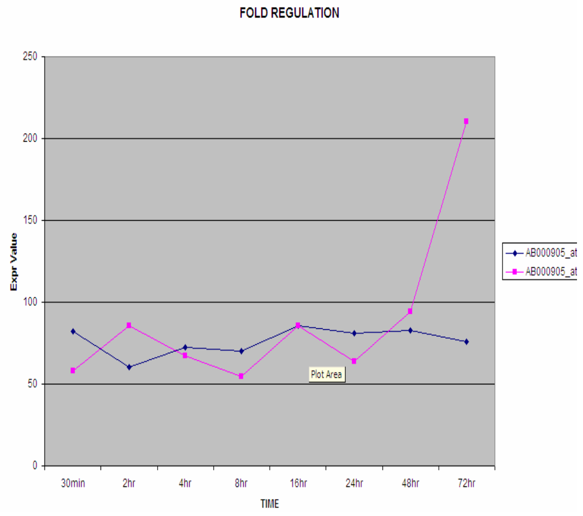


Figure 4: *AB000905*

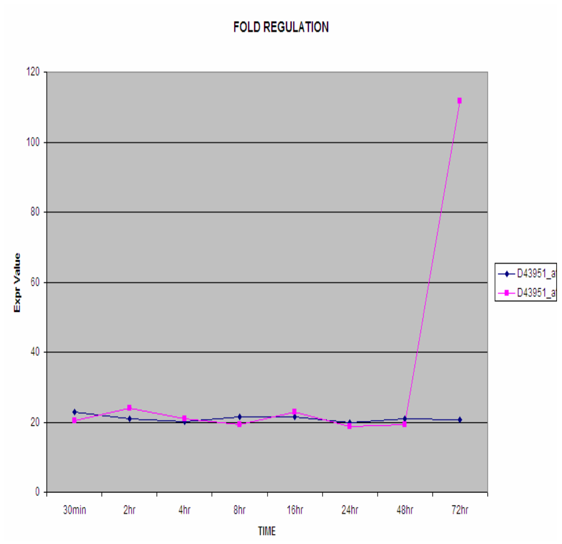


Figure 5: *D43951*

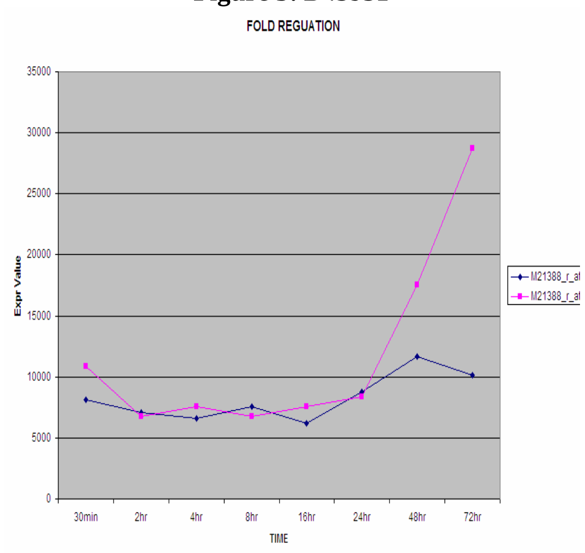


Figure 6: *M21388*

Thus, using 1.5 increase or decrease fold regulation as the cut-off between 48 hours and 72 hours, we obtain 6 different genes that we can use as potential genetic markers. We choose to pay close attention to Day 2 and Day 3 because previous published research has indicated that drastic changes in gene expression profiles for infected HIV genes occur after 48 hours [8]. Thus, the accession numbers for these genes are:

1. J04423 with AFFX-BioDn-5_at as the probe set
2. X03453 with AFFX-CreX-5_at as the probe set
3. U14573 with hum_alu_at as the probe set
4. AB000905 with AB000905_at as the probe set
5. D43951 with D43951_at as the probe set
6. M21388 with M21388_r_at as the probe set

From looking up the 6 different genes in the GenBank and NCBI databases, we were able to confirm the results as shown in Table 1 [17]:

Accession Number	Gene	Gene Type	Gene Product
J04423	bioD	Protein Coding	enzyme called dethiobiotin synthetase
X03453	cre	Protein Coding	Enzyme called cyclization recombinase
U14573	Alu	Protein Coding	actively transcribed by pol III, altered protein sequences
AB000905	HIST1H4I	Protein Coding	histone 1, H4i
D43951	PUM1	Protein Coding	Assist in RNA binding and mRNA metabolism
M21388	GLA	Protein Coding	Enzyme called alpha-galactosidase

Table 1: Potential genetic HIV-1 markers and their confirmed functionality

Although some of these genes belong to different chromosomes, we can infer that they are affected in a similar fashion when exposed to HIV-1 virus after 3 days. Therefore, we can see why it is important to not only look for co-expressed genes, but also for coherent genes in order to obtain a full snap shot of the gene's profile.

4. Conclusions

All of the gene products listed in the given table are highly affected by the HIV-1 virus. However, to really confirm whether these genes can be used as genetic markers in real life, *in-vivo* samples should be tested as well to help confirm these results. This is because *in-vivo* samples come directly from the individual and not post-infected outside the body. *In-vivo* samples from the different stages of HIV/AIDS should also be used.

Overall, the results presented in this paper are promising, and provide a good starting point for further research in this area. This research can contribute to the HIV Pharmacogenomics field by confirming HIV genetic markers, which would lead to rapid diagnosis and customized treatments. In fact, doctors can easily use these markers, along with other markers for other diseases, to rapidly diagnose a patient's profile in one genetic scan. At the same time, these markers can be used to monitor the progression or treatment of the disease.

Acknowledgements

C. Domeniconi is supported in part by the NSF CAREER Award IIS-0447814.

References

1. *AIDS Epidemic Update*, report, UN AIDS, December 2000.
2. Holodniy, M., Kuritzkes, D.R., Byer, D, Murray, P. "HIV viral load markers in clinical practice." Nature Medicine. Volume 2, pp.625-629, 1996.
3. Bumgarner, E., Geiss, G.K., V'houte, D., Haglin, J. "Large scale Monitoring of Host Cell Gene Expression during HIV-1 infection Using cDNA Microarrays." Acedemic Press. December 1999.
4. Conrad, J. Impact of Pharmacogenomics on FDA's Drug Review Process, SACGHS Meeting, Washington, DC, October 22, 2003.
5. Corbeil, J., Genini, D., Sheeter,D. "Temporal Gene Regulation During HIV-1 Infection of Human CD4+ T Cells." Genome Research. 2 April, 2001.
5. Weiner, M.P., Hudson, T.J. "Introduction to SNPs: Discovery of Markers for Disease." Biotechniques. Volume 32, pp. s5-s32, 2002.
6. Gary K. Geiss, G.K., Hammand, D. "Pathogenesis (HIV): Virus can alter the way genes function within days of exposure." Virology. Volume 46, pp. 23-27, 2000.
7. University of Tokyo Japan Laboratory of DNA Information Analysis of Human Genome Center, Institute of Medical Science. Distance/Similarity measures, 2002.
8. Fugen, L., Stormo, G. "Selection of optimal DNA oligos for gene expression arrays." Bioinformatics. Volume 17(11), pp. 1067-1079, 2001.
9. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D., "Cluster analysis and display of genome-wide expression patterns". Proceedings of the National Academy of Science USA, 95 14863-14868, December 1998
10. Luo, F., Khan, L. "Hierarchical Clustering of Gene Expression Data", Department of Computer Science, University of Texas, Dallas. March 2003.
11. Yeung, K.Y., Jung, L. "Model-Based Clustering and Data Transformations for Gene Expression Data". The Third Georgia Tech-Emory International Conference on Bioinformatics. 2001.
12. Jiang, D., Zhang, X., Pei, J. "Interactive exploration of coherent patterns in time-series gene expression data." In proceedings of the ninth ACM SIGKDD International Conference of Knowledge Discovery and Data Mining (KDD '03), Washington, DC, USA, August 24-27, 2003.
13. Kano, M., Kashima, H., Slyder, E. "A method for Normalization of Gene Expression Data." Genome Informatics. Volume 14, pp. 336-337, 2003.
14. Oracle Data Mining Technical White Paper. Oracle Corporation. December 2002.
15. Tavazoie S., Hughes D., Campbell M., Cho R.J. Church G. Systematic determination of genetic network architecture. *Nature Genet*, pages 281–285, 1999.
16. Jiang, D., Pei, J., Zhang, A. Towards Interactive Exploration of Gene Expression Patterns. State University of New York at Buffalo, 2002.
17. Rahmann, S. Rapid Large-scale oligonucleotide selection for microarrays. WABI, 2002.

Feature Filtering with Ensembles Using Artificial Contrasts

Eugene Tuv[*]and Kari Torkkola[†]

Keywords: Feature ranking cut-off, Ensemble methods, Artificial contrast variables

Abstract

In contrast to typical variable selection methods such as CFS, tree-based ensemble methods can produce numerical importances of input variables considering all variable interactions, not just one or two variables at a time. However, they do not indicate a cut-off point: how to set a threshold to the importance. This paper presents a straightforward approach to doing this using artificial contrast variables. The result is a truly autonomous variable selection method that considers all variable interactions, and does not require a pre-set number of important variables.

1 Ensemble Methods in Feature Ranking

In this paper we try to address a problem of feature filtering, or removal of irrelevant inputs in very general supervised settings: target variable could be numeric or categorical, input space could have variables of mixed type with non-randomly missing values, underlying $X-Y$ relationship could be very complex and multivariate, and data could be massive in both dimensions (tens of thousands of variables, and millions of observations). Ensembles of unstable but very fast and flexible base learners such as trees (with embedded feature weighting) can address most of the listed challenges. They have proved to be very effective in variable ranking in problems with up to a hundred thousand predictors [2, 7].

Relative feature ranking provided by such ensembles, however, does not separate relevant features from noise. Only a list of importance values is produced without a clear indication which variables to include, which to discard. The main idea in this work relies on the following reasonable assumption: a stable feature ranking method, such as an ensemble of trees, that measures relative relevance of an input to a target variable Y would assign a significantly (in statistical sense) higher rank to a legitimate variable X_i than to a artificial variable created from the same distribution as X_i, independently of Y.

2 The Algorithm: Artificial Contrasts with Ensembles

In order to determine a cut-off point for importances, there needs to be a *contrast* variable that is known to be truly independent of the target. By comparing the derived importances to this contrast (or several), one can then use a statistical rank test to determine which variables are truly important.

We propose to obtain these artificial contrast variables by randomly permuting values of original N variables across the K examples. Generating just random variables from some simple distribution, such as Gaussian or uniform, is not sufficient, because the values of original variables may exhibit some special structure.

Importances and their ranks are then computed for all variables, including the artificial contrasts. To gain statistical significance, this is repeated T times, recording the rankings of all variables including contrasts. The quantile (which can be minimum or median) over the N contrasts of the ranks of the contrasts is evaluated. A statistical rank test (Wilcoxon test) is performed to compare the ranks of the original variables to the quantile ranks of the contrasts. Variables with significantly better rank than contrasts are set aside and included in important variables.

The target is now predicted using only these important variables, and a residual of the target is computed. The process is repeated until no variables remain whose rank would be significantly higher than that of the contrasts. Algorithm 1 lists all these steps using the notation in Table 1.

An important part of the algorithm is Step 10, in which the target is estimated using only important variables found in the current iteration, and this estimate is subtracted from the current target variable. This step removes the effect of these important variables from the target and leaves only the residual for the next iteration, to be explained by the residual variables.

As the function $g(.,.)$ we have used ensembles of trees. Any classifier/regressor function can be used, from which variable importances from all variable interactions can be derived. To our knowledge, only ensembles of trees can provide this conveniently.

To account for possible biases in the learning engine (for example, multilevel categorical vs. numeric), in step 6, one can compare variable rank only to the rank(s) of permuted version(s) of itself instead of to the

[*]Intel, Analysis and Control Technology, Chandler, AZ, USA, eugene.tuv@intel.com

[†]Motorola, Intelligent Systems Lab, Tempe, AZ, USA, Kari.Torkkola@motorola.com

quantile over all permuted variables.

Algorithm 1. Artificial Contrasts with Ensembles

```
1.   set $\Phi \leftarrow \{\}$; set $F \leftarrow \{X_1, ..., X_N\}$
2.   for $i = 1, ..., T$ do
3.       $\{Z_1, ..., Z_N\} \leftarrow$ permute$\{X_1, ..., X_N\}$
4.       set $F_P \leftarrow F \cup \{Z_1, ..., Z_N\}$
5.       $\mathbf{M}_{i.} = g_I(F_P, Y); \mathbf{R}_{i.}$ =ranks($\mathbf{M}_{i.}$)
     endfor
6.   $\mathbf{r}_m = \text{quantile}_{j \in \{Z_1,...,Z_N\}} \mathbf{R}_{.j}$
7.   Set $\hat{\Phi}$ to those $\{X_k\}$ for which $\mathbf{R}_{.k} < \mathbf{r}_m$
     with rank test significance 0.05
8.   If $\hat{\Phi}$ is empty, then quit.
9.   $\Phi \leftarrow \Phi \cup \hat{\Phi}; F = F \setminus \hat{\Phi}$
10.  $Y = Y - g_Y(\hat{\Phi}, Y)$
11.  Go to 2.
```

Table 1: Notation used in Algorithm 1.

X	set of original variables
Y	target variable
Z	permuted versions of X
F	current working set of variables
Φ	set of important variables
$\mathbf{M}_{i.}$	ith row of matrix $\mathbf{M}$
$\mathbf{M}_{.j}$	jth column of matrix $\mathbf{M}$
$g_I(F, Y)$	function that trains an ensemble based on variables F and target Y, and returns a row vector of importances for each variable in F
$g_Y(F, Y)$	function that trains an ensemble based on variables F and target Y, and returns a prediction of Y
ranks($\mathbf{m}$)	function that returns a row vector of ranks given an input vector $\mathbf{m}$ of real-valued importances.

3 Experiments

As advocated by [5], an experimental study must have relevance and it must produce insight. The former is achieved by using real data sets. However, such studies often lack the latter component failing to show exactly why and under which conditions one method excels another. The latter component can be achieved by using synthetic data sets, because they let one vary systematically domain characteristics of interest, such as the number of relevant and irrelevant attributes, the amount of noise, and the complexity of the target concept.

We describe now preliminary experiments with the proposed method using synthetic data sets. The final version of the paper will report experimentation with real data sets.

A very useful data generator is described by [3]. This generator produces data sets with multiple non-linear interactions between input variables. Any greedy method that evaluates importance of a single variable only at a time is bound to fail with these data sets.

Figure 1 presents average results with 50 generated datasets, 500 samples each. For each, twenty $N(0,1)$ distributed input variables were generated. The target is a multivariate function of ten of those, thus ten are pure noise. The target function is generated as a sum of twenty multidimensional Gaussians, each Gaussian involving about four randomly drawn input variables at a time. Thus all of the "important" ten input variables are involved in the target, to a varying degree. Figure 1 illustrates how well they can be detected as important variables. We used Gradient Boosting Trees (GBT) [4] as the ensemble of 500 trees.

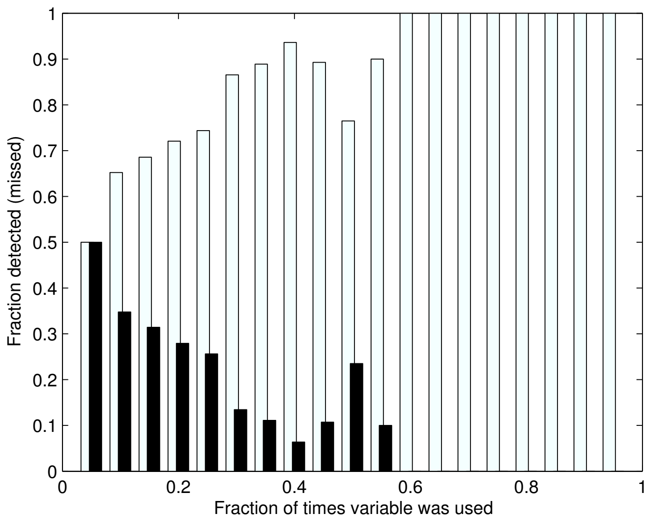


Figure 1: Experiment with Friedman's data generator. Horizontal axis: The fraction of the Gaussians the variable was involved in, when generating the multivariate relationship to the target in one dataset. Vertical axis: The fraction of times such a variable was detected as an important variable. Darker bars denote false rejections.

Figure demonstrates several things. First, it shows that multivariate interactions between variables are discovered by an ensemble of trees. Second, as long as the variable is involved in at least 20% of the interactions, is can be detected with a rate higher than 70%. These numbers are naturally a function of the size of the data set. The final paper will present a larger array of experiments.

Regarding false acceptance rates (not illustrated), only 1.5% of the true noise variables were falsely accepted as important variables. One must emphasize that typical filter methods based on evaluating any pairwise relevance criterion will fail with these data sets because of the underlying multivariate relationships.

We also experimented using data with linear relationships, where the target is a simple linear combination of a number of input variables plus noise. This would be a simple problem for stagewise linear regression, but typically linear problems are harder for trees. However, with data sets of 500 samples, we detected 100% of the important variables, as long as their variance was larger than 1.5 times the variance of additive noise. False acceptance rate remained at 1.5%. Using a smaller data set, just 200 samples, decreases the detection rate of a variable with variance 1.5 times the variance of additive noise to 65%.

4 Discussion

Stagewise tree regression itself is not a novel idea. In fact, this is the basis of GBT [4]. Stagewise *linear* regression in feature selection has been used earlier by [6]. However, this method operates greedily on one variable at a time, it applies only to numerical variables by projecting the data on the null space of earlier discovered variable. Furthermore, it applies only to classification problems.

The idea of adding random "probe variables" to the data for feature selection purposes has been used by [1]. Adding permuted original variables as random "probes" has been used by [8] in the context of comparing gene expression differences across two conditions. Statistical tests have not been used to compare *ranks* of artificial probes to real variables in the context of variable selection.

The presented method retains all good features of ensembles of trees: mixed type data can be used, missing variables can be tolerated, and variables are not considered in isolation. The method is applicable to both classification and regression. For correlated variables the method can work in one of two modes: either report correlated variables or ignore them by the virtue of computing the residual of the target.

References

[1] J. Bi, K.P. Bennett, M. Embrects, C.M. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, March 2003.

[2] A. Borisov, V. Eruhimov, and E. Tuv. Dynamic soft feature selection for tree-based ensembles. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, New York, 2005.

[3] J.H. Friedman. Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University, 1999.

[4] J.H. Friedman. Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University, 1999.

[5] P. Langley. Relevance and insight in experimental studies. *IEEE Expert*, 11:11–12, October 1996.

[6] H. Stoppiglia, G. Dreyfus, R.Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, March 2003.

[7] Kari Torkkola and Eugene Tuv. Ensembles of regularized least squares classifiers for high-dimensional problems. In Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, 2005.

[8] V.G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5116–5121, April 24 2001.

Speeding Up Multi-class SVM Evaluation by PCA and Feature Selection

Hansheng Lei, Venu Govindaraju
CUBS, Center for Unified Biometrics and Sensors
State University of New York at Buffalo
Amherst, NY 14260
Email: {hlei,govind@cse.buffalo.edu}

Abstract

Support Vector Machine (SVM) is the state-of-the-art learning machine that has been very fruitful not only in pattern recognition, but also in data mining areas. E.g., SVM has been extensively and successfully applied in feature selection for genetic diagnosis. In this paper, we do the contrary,i.e., we use the fruits achieved in the applications of SVM in feature selection to improve SVM itself. We propose combining Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) into multi-class SVM. We found that SVM is invariant under PCA transform, which qualifies PCA to be a desirable dimension reduction method for SVM. On the other hand, RFE is a suitable feature selection method for binary SVM. However, RFE requires many iterations and each iteration needs to train SVM once. This makes RFE infeasible for multi-class SVM if without PCA dimension reduction, especially when the training set is large. Our experiments on the benchmark database MNIST and other commonly-used datasets show that PCA and RFE can speed up the evaluation of SVM by an order of 10 while maintaining comparable accuracy.

Keywords: Support Vector Machine, Principle Component Analysis, Recursive Feature Elimination, Multi-class Classification

1 Introduction

The Support Vector Machine (SVM) was originally designed for binary classification problem [1]. It separates two classes with maximum margin. The margin is described by Support Vectors (SV) which are determined by solving a Quadratic Programming(QP) optimization problem. The training of SVM, dominated by the QP optimization, used to be very slow and lack of scalability. A lot of efforts have been done to crack the QP problem and enhance its scalability [13]. The bottleneck lies in the kernel matrix. Suppose we have N data points for training, then the size of the kernel matrix will be $N \times N$. When N is more than thousands (say, $N = 5000$), the kernel matrix is too big to stay in the memory of a common personal computer. This had been a challenge for SVM until the Sequential Minimum Optimization (SMO) was invented by [13]. The space complexity of SVM training is dramatically brought down to $O(1)$ by SMO. Thus, the training problem was almost solved, although there might be lurking more powerful solutions. With the support of SMO, the great scalability of SVM has demonstrated its promising potentials in data mining areas [17].

In the past decade, SVM has been widely applied in pattern recognition as well as data mining with fruitful results. However, the SVM itself also needs improvement in both training and testing (evaluation). A lot of work have been done to improve the SVM training and the SMO can be considered as the state-of-the-art solution for that. Comparatively, only a few efforts have been put at the evaluation side of SVM[2, 4].

In this paper, we propose a method for SVM evaluation enhancement via Principle Component Analysis (PCA) and Recursive Feature Elimination (RFE). PCA is an orthogonal transformation of coordinate system that preserves the Euclidean distance of original points (each point is considered as a vector of features or components). By PCA transform, the energy of points are concentrated into the first few components. This leads to dimension reduction. Feature selection has been heavily studied, especially for the purpose of gene selection on microarray data. The common situation in the gene related classification problem is: there are thousands of genes but only no more than hundreds of samples, i.e., the number of dimensions is much more than the number of samples. In this condition, the problem of *overfitting* arises. Among those genes, which of them are discriminative? Finding the minimum subset of genes that interact can help cancer diagnosis. RFE in the context of SVM has achieved excellent results on feature selection [5]. Here, we do the contrary, i.e., we use the fruits of the application of SVM in feature selection to improve SVM itself.

The rest of this paper is organized as follows. After the introduction, we briefly discuss the background of

SVM, PCA and RFE as well as some related works in §2. Then, we prove SVM is invariant under PCA and describe how to incorporate PCA and RFE into SVM to speed up SVM evaluation in §3. Experimental results on benchmark datasets are reported in §4. Finally, conclusion is drawn in §5.

2 Background and Related Works

In this section, we discuss the basic concepts of SVM and how RFE is incorporated into SVM for feature selection on gene expressions. In addition, PCA is also introduced. We prove that SVM is invariant under PCA transformation and the propose combining PCA and RFE safely to improve SVM evaluation.

2.1 Support Vector Machines (SVM) The basic form of a SVM classifier can be expressed as:

$$g(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b, \tag{2.1}$$

where input vector $\mathbf{x} \in \Re^n$, $\mathbf{w}$ is a normal vector of a separating hyper-plane in the *feature space* produced from the mapping of a function $\phi(\mathbf{x}) : \Re^n \mapsto \Re^{n'}$ ($\phi(\mathbf{x})$ can be linear or non-linear, n' can be finite or infinite), and b is a bias. Since SVM was originally designed for two-class classification, the sign of $g(\mathbf{x})$ tells vector $\mathbf{x}$ belongs to class 1 or class -1.

Given a set of training samples $\mathbf{x}_i \in \Re^n$, $i = 1, \cdots, N$ and corresponding labels $y_i \in \{-1, +1\}$, the separating hyper-plane (described by $\mathbf{w}$) is determined by minimizing the *structure risk* instead of the *empirical error*. Minimizing the structure risk is equivalent to seeking the optimal margin between two classes. The width of the margin is $\frac{2}{\mathbf{w}\cdot\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$. Plus some trade-off between structure risk and generalization, the training of SVM is defined as a constrained optimization problem:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_{i=1}^{N} \xi_i \\ &\text{subject to} \\ &\quad y_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ &\quad \xi_i \geq 0, \forall i, \end{aligned} \tag{2.2}$$

where parameter C is the trade-off.

The solution to (2.2) is reduced to a QP optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{a}} \quad \boldsymbol{a}^T\boldsymbol{a} - \frac{1}{2}\boldsymbol{a}^T\mathbf{H}\boldsymbol{a} \\ &\text{subject to} \\ &\quad 0 \leq \alpha_i \leq C, \forall i, \\ &\quad \sum_{i=1}^{N} y_i\alpha_i = 0, \end{aligned} \tag{2.3}$$

where $\boldsymbol{a} = [\alpha_1, \cdots, \alpha_N]^T$, and $\mathbf{H}$ is a $N \times N$ matrix, called the *kernel matrix*, with each element $\mathbf{H}(i,j) = y_iy_j\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$.

Solving the QP problem yields:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i), \tag{2.4}$$

$$b = \sum_{j=1}^{N} \alpha_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) + y_i, \forall i. \tag{2.5}$$

Each training sample $\mathbf{x}_i$ is associated with a Lagrange coefficient α_i. Those samples whose coefficient α_i is nonzero are called *Support Vectors* (SV). Only a small portion of training samples become SVs (say, 3%).

Substituting eq. (2.4) to (2.1), we have the formal expression of SVM classifier:

$$\begin{aligned} g(\mathbf{x}) &= \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \end{aligned} \tag{2.6}$$

where the K is a kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. By the kernel functions, we do not have to explicitly know $\phi(\mathbf{x})$. The most commonly-used kernel functions are: 1)Linear kernel, i.e., $\phi(\mathbf{x}_i) = \mathbf{x}_i$, thus, $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i^T\mathbf{x}_j$; 2) Polynomial kernel, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$, where c and d are some positive constants. 3)Gaussian Radial Basis (RBF) kernel, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = e^{(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})}$. If the used kernel is linear, then the SVM is called *linear SVM*, otherwise, *non-linear SVM*.

The QP problem (2.3) involves a matrix $\mathbf{H}$ that has a number of elements equal to the square of the number of training samples. If there are more than 5000 training samples, $\mathbf{H}$ will not be able to fit into a 128-Megabyte memory (assume each element is 8-byte double). Thus, the QP problem will become intractable via standard QP techniques. There were a lot of efforts to meet his challenge and finally the SMO gave a perfect solution [13] in 1999. SMO breaks the large QP problem into a series of smallest possible QP problems. Solving those small QP optimizations analytically only needs $O(1)$ memory. In this way, the scalability of SVM is enhanced dramatically. Since the training problem was cracked by SMO, the focus of efforts has been transferred to the evaluation of SVM [2]: improve accuracy and evaluation speed. Increasing accuracy is quite challenging, while the evaluation speed has much margin to work on. To enhance the SVM evaluation speed, we should refer back to eq. (2.6), from which we can imagine there are following possible ways to achieve speedup:

1. Reduce the number of SVs directly. $\mathbf{w}$ is described by a linear combination of SVs and to obtain $g(\mathbf{x})$, $\mathbf{x}$ needs to do inner product with all SVs. Thus, reducing the total number of SVs can directly reduce the computational time of SVM in test phase. Burges et al. proposed such a method[2]. It tries to find a $\mathbf{w}'$ that approximates $\mathbf{w}$ as close as possible in the feature space. Similarly as $\mathbf{w}$, $\mathbf{w}'$ is expressed by a list of vectors (called *reduced set*) associated with corresponding coefficients (α_i'). However, the method for determining the reduced set is computationally very expensive. Exploring in this direction further, Downs et al. found a method to identify and discard *unnecessary* SVs (those SVs who linearly depend on other SVs) while leaving the SVM decision unchanged [4]. A reduction in SVs as high as 40.96% was reported therein.

2. Reduce the size of quadratic program and thus reduce the number of SVs indirectly. A method called *RSVM* (Reduced Support Vector Machines) was proposed by Lee et al.[10]. It preselects a subset of training samples as SVs and solves a smaller QP. The authors reported that RSVM needs much less computational time and memory usage than standard SVM. A comparative study on RSVM and SVM by Lin et al.[11] showed that standard SVM possesses higher generalization ability, while RSVM may be suitable in very large training problems or those that have a large portion of training samples becoming SVs.

3. Reduce the number of vector components. Suppose the length of vector is originally n, can we reduce the length to $p \ll n$ by removing non-discriminative components or features? To the best of our knowledge, there are very few efforts,if any, have been done in this direction. Since there are mature fruits in dimension reduction and feature selection, we propose to make use of successful techniques gained by the applications of SVM in feature selection to improve SVM, especially multi-class SVM.

2.2 Recursive Feature Elimination (RFE) RFE is an iterative procedure to remove non-discriminative features [6] in binary classification problem. The framework of RFE consists of three steps: 1)Train the classifier; 2)Compute the ranking of all features with a certain criterion in term of their contribution to classification; 3)Remove the feature with lowest ranking. Goto step 1) until no more features.

Note that in step 3), only one feature is eliminated each time. It may be more efficient to remove several features at a time, but at expense of possible performance degradation. There are many feature selection methods besides RFE. However, in the context of SVM, RFE has been proved to be one of the most suitable feature selection methods by extensive experiments [6]. The outline of RFE in SVM is as follows:

Algorithm.1 SVM-RFE

Inputs: Training samples $\mathbf{X_0} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ and class labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$.

Outputs: feature ranked list $\mathbf{r}$.

```
1: $\mathbf{s} = [1, 2, \cdots, n]$; /*surviving features*/
2: $\mathbf{r} = []$;            /*feature ranking list*/
3: while $\mathbf{s} \neq []$ do
4:     $\mathbf{X} = \mathbf{X}_0(\mathbf{s}, :)$; /*only use surviving features of every sample*/
5:     Train linear SVM on $\mathbf{X}$ and $\mathbf{Y}$ and obtain $\mathbf{w}$;
6:     $c_i = w_i^2, \forall i$ /*weight of the $i$th feature in $\mathbf{s}$ */
7:     $f = argmin_i(c_i)$; /*index of the lowest ranking*/
8:     $\mathbf{r} = [\mathbf{s}(f), \mathbf{r}]$; /*update list*/
9:     $\mathbf{s} = \mathbf{s}(1 : f-1, f+1 : length(\mathbf{s}))$; /*eliminate the lowest ranked feature*/
10: end while. /*end of algorithm*/
```

Note that the ranking criterion for the discriminability of features is based on $\mathbf{w}$, the normal vector of the separating hyper-plane, as shown in line 6. The idea here is that one may consider a feature is discriminative if it significantly influences the width of the margin of the SVM. Recall that SVM tries to seek the maximum width of margin and the width is $\frac{2}{\|\mathbf{w}\|^2} = 2/\sum_{i=1}^{n} w_i^2$.So, if a feature with large w_i^2 is removed, the change of the margin is also large, thus, this feature is very important.

Also note that in line 5, linear SVM is usually used in gene selection applications. According to our experience, linear SVM works better than non-linear SVM in gene selection because the gene expression samples tend to be linear separable. However, in regular domains, like handwritten character or facial classification using images as samples, non-linear SVM is usually better.

RFE requires many iterations. If we have n features in total and want to choose the top p features, the number of iterations is $n-p$ if only one feature is eliminated at a time. More than one features can be removed a time by modifying line 7-9 in the algorithm above, at the expense of possible degradation of performance. Usually half features can be eliminated at a time on mircoarray gene data until the number of features come down to no more than 100 (then features should be eliminated with caution).

2.3 Principal Component Analysis (PCA) It is well known that the training of SVM is very slow compared to other classifiers. RFE works smoothly in gene selection problems, because there are usually no more than hundreds of training samples of gene expressions in that situation. When we come back to conventional classification problem, the training set usually has large size (say, tens of thousands of samples). In this case, it is desirable to reduce the computational time of each training as well as the total number of iterations. We use PCA to preprocess the data and perform dimension reduction before RFE.

Given a set of centered vectors $\mathbf{x}_i \in \Re^n, k = 1, \cdots, N$, $\sum_{i=1}^{N} \mathbf{x}_i = 0$, PCA diagonalizes the scatter matrix:

$$S = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T. \tag{2.7}$$

To do this, the eigen problem has to be solved:

$$S\mathbf{v} = \lambda \mathbf{v}, \tag{2.8}$$

where eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v} \in \Re^n$. Those eigenvectors $\mathbf{v}_i, i = 1, \cdots, n$, are called the *Principal Components*. Arbitrary pair of principle components are orthogonal to each other, i.e., $\mathbf{v}_i^T \mathbf{v}_j = 0 (i \neq j)$. And, the eigenvectors are normal, i.e., $\mathbf{v}_i^T \mathbf{v}_i = 1$. Those components span a new orthogonal coordinate system with each component as an axis. The original vector $\mathbf{x}_i$ has its new coordinates in the new system by projecting it to every axis (suppose $V = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$, the projection of $\mathbf{x}_i$ to the new coordinate system is $\mathbf{x}'_i = V^T \mathbf{x}_i$). The projected vector $\mathbf{x}'_i$ has the same dimension n as $\mathbf{x}_i$, since there are n eigenvectors together.

The benefit of projecting $\mathbf{x}_i$ into the PCA space is: those eigenvectors associated with large eigenvalues are more principle, i.e., the projecting values to those $\mathbf{v}_i$ are larger and thus more important. Those less important components can be removed and thus the dimension of the space is safely reduced. How many components should be eliminated depends on the applications. PCA is a suitable dimension reduction method for SVM classifiers because SVM is invariant under PCA transform. We will give proof in next section.

3 Speeding Up SVM by PCA and RFE

We speed up SVM evaluation via PCA for dimension reduction and RFE for feature selection. RFE has been combined with linear SVM in the applications of gene selection. Here, we incorporate RFE into standard SVM (linear or non-linear) to eliminate the non-discriminative features and thus enhance SVM in test phase. A motivation here is for the conventional classification problems, like handwriting recognition, face detection, the input vectors of features are the pixels of images (if we do not use predefined features). For a 28×28 image, the vector will be 784 long. Reducing the length of every vector while maintaining the accuracy is of practical interests. Another motivation is that not all feature are discriminative, especially in multi-class problem. For example, in the handwritten digit recognition, when we try to distinguish '4' and '9',usually only the upper part (closed or open) is necessary to tell them apart. The other parts or features are actually of little use here. An efficient implementation method of multi-class SVM is 'One-vs-One' (OVO) [8]. OVO is constructed by training binary SVMs between pairwise classes. Thus, OVO model consists of $\frac{M(M-1)}{2}$ binary SVMs for M-class problem. Each of the binary SVM classifier casts one vote for its favored class, and finally the class with maximum votes wins. There all other multi-class SVM implementation methods besides OVO, such "One-vs-All" [16, 15] and DAG SVM [14]. None of them outperforms OVO significantly, if comparable. But most of them are to decompose multi-class problem to a series of binary problems. Therefore, the concept of RFE originally for two-class is also applicable in multi-class. Some features are discriminative for this pair of classes, but may be not useful for another pair. We do not have to use a fixed number of features for every binary SVM classifier. Using PCA and RFE, we propose the following Feature Reduced Support Vector Machines (FR-SVM) algorithms in the framework of OVO.

Algorithm.2 Training of FR-SVM

Inputs: Training samples $\mathbf{X_0} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$; class labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$; M(number of classes); P(number of chosen most principal components); and F(number of chosen top ranked features).

Outputs: Trained multi-class SVM classifier OVO; [V,D,P](the parameters for PCA transformation).

```
1: $[V, D] = PCA(X_0)$; /*Eigen vectors & values */
2: $\mathbf{X} = PCA\_Transform(V, D, \mathbf{X}_0, P)$; /*dimension
   reduction by PCA to $P$ components */
3: for i=1 to M do
4:   for j=i+1 to M do
```

```
5:      $C_1 = \mathbf{X}(:, find(\mathbf{Y} == i))$; /*data of class $i$*/
6:      $C_2 = \mathbf{X}(:, find(\mathbf{Y} == j))$; /*data of class $j$*/
7:      $\mathbf{r}$ = SVM-RFE($C_1, C_2$);      /*ranking list*/
8:      $Fc \leftarrow F$ top ranked features from $\mathbf{r}$;
9:      $C_1' = C_1(Fc, :)$; /*only the $F$ components*/
10:     $C_2' = C_2(Fc, :)$;
11:     Binary SVM model $\leftarrow$ Train SVM on $C_1'$ & $C_2'$;
12:     $OVO\{i\}\{j\} \leftarrow$ {Binary SVM Model, Fc};
        /*save the model and selected features*/
13:   end for
14: end for/*end of algorithm*/
```

Algorithm.3 Evaluation of FR-SVM

Inputs: Evaluation samples $\mathbf{X_0} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$; Trained multi-class SVM classifier OVO; M(number of classes); [V, D, P](the parameters for PCA transformation).

Outputs: Labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$.

```
1: $X = PCA\_Transform(V, D, X_0, P)$;
2: for k=1 to N do
3:   $\mathbf{x} = \mathbf{X}(:, k)$; /*one test sample*/
4:   Votes=zeros(1,M); /*votes for each class*/
5:   for i=1 to M do
6:     for j=i+1 to M do
7:       {Binary SVM, Fc} $\leftarrow OVO\{i\}\{j\}$ ;
8:       $\mathbf{x}' = \mathbf{x}(Fc)$; /*only the selected components*/
9:       Label $\leftarrow$ SVM($\mathbf{x}'$); /*binary SVM evaluation*/
10:      Votes(Label)=Votes(Label)+1;
11:    end for
12:  end for
13:  $\mathbf{Y}(k) = find(Votes == max(Votes))$; /*the one with max votes wins*/
14: end for/*end of algorithm*/
```

Algorithm.2 and 3 are the training and evaluation of FR-SVM respectively. Note that for every binary SVM, we apply RFE to obtain the top F most discriminative features(components). Then, in evaluation, we only use those selected features. The F features may be different across different pair of classes. Instead of predefining the number of features F , one might wants to determine such an optimal F automatically by cross validation. This is a choice but too time-consuming when the training set or the number of classes is large. To make the FR-SVM more feasible, we let F be user-defined. Similarly, FR-SVM leaves P (number of chosen principal components) to the users.

We use PCA for dimension reduction because SVM is invariant under PCA transformation. We state it as a theorem and prove it.

Theorem 3.1. *The evaluation result of SVM with linear, polynomial and RBF kernels is invariant if input vector is transformed by PCA.*

Proof. Suppose $V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$, where $\mathbf{v}_i$ is an eigenvector. The PCA transformation of vector $\mathbf{x}$ is $V^T\mathbf{x}$. Recall the optimization problem (2.3). If kernel matrix $\mathbf{H}$ does not change, then the optimization does not change (under the same constraints). Since $\mathbf{H}(i,j) = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, all we need for proof of invariance is $K(V^T\mathbf{x}_i, V^T\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$. Note that all eigenvectors are normal,i.e., $\mathbf{v}_i^T\mathbf{v}_i = 1, \forall i$ and mutually orthogonal, i.e., $\mathbf{v}_i^T\mathbf{v}_j = 0 (i \neq j)$. Therefore, we have $V^TV = I$, where I is a $n \times n$ unit matrix.

Now, for linear case, $K(V^T\mathbf{x}_i, V^T\mathbf{x}_j) = V^T\mathbf{x}_i \cdot V^T\mathbf{x}_j = (V^T\mathbf{x}_i)^T V^T\mathbf{x}_j = \mathbf{x}_i^T(VV^T)\mathbf{x}_j = \mathbf{x}_i^T I \mathbf{x}_j = \mathbf{x}_i^T\mathbf{x}_j = K(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, we can prove the polynomial case.

For RBF kernel, it is enough to show $\|V^T\mathbf{x}_i - V^T\mathbf{x}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. Expanding the Euclidean norm, we have $\|V^T\mathbf{x}_i - V^T\mathbf{x}_j\|^2 = (V^T\mathbf{x}_i)^2 + (V^T\mathbf{x}_j)^2 - 2(V^T\mathbf{x}_i)^T(V^T\mathbf{x}_j) = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i\mathbf{x}_j = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. $\triangle$

Backed up by the theorem above, we can safely utilize PCA to preprocess the training samples and reduce their dimensions. Of course, this preprocessing is optional, if the original dimension of samples is not high (say, like below 50), we do not have to carry PCA transformation. Our recommendation to use PCA to reduce dimension before RFE is due to two concerns: one is that the RFE needs many iterations. The number of iterations is directly related to the dimension (i.e., the number of original features). Thus, dimension reduction leads to reduction of RFE iterations; another is that SVM training is quite slow, dimension reduction saves the computational time of each iteration of training. We will see how PCA and RFE contribute to the speedup of SVM by experiments in the next section.

4 Experiments

The main tasks of the experiments are to: 1)test the accuracy of FR-SVM. For every classifier, one of the most important measures is its accuracy. Since our goal is to enhance the SVM evaluation speed, we should be careful not to jeopardize the performance of SVM. 2)observe how much speedup we can achieve without negative influence on the performance of SVM.

Three datasets were used in our experiments. The description of them are summarized in table 1. The Iris is one of the most classical datasets for testing classification, available form [7]. It has 150 samples with 50 in each of the three classes. We used the first 35 samples from every class for training and the left 15 for testing. The MNIST database [9] contains 10 classes

Table 1: Description of the multi-class datasets used in the experiments.

Name	# of Training Samples	# of Testing Samples	# of Classes	# of Attributes
Iris	105	45	3	4
MNIST	60000	10000	10	784
Isolet	6238	1559	26	617

of handwritten digits (0-9). There are 60,000 samples for training and 10,000 samples for testing. The digits have been size-normalized and centered in a fixed-size image (28×28). It is a benchmark database for machine learning techniques and pattern recognition methods. The third dataset Isolet were generated from spoken letters [7]. We chose it because it has 26 classes, which is reasonably high among publicly available datasets. The number of the three class varies from 3 to 26. The number of samples varies from hundreds (Iris) to tens of thousand (MNIST). We hope the typicality of the datasets makes the experimental results convincing.

The software package we used was the OSU SVM Classifier Matlab Toolbox [12], which is based on the software LIBSVM [3]. On each dataset, we trained multi-class OVO SVM. The kernel we chose was the RBF_kernel, since it has been widely observed RBF_kernel usually outperforms other kernels, such as linear and polynomial ones. The regularizing parameters C and σ were determined via cross validation on the training set. The validation performance was measured by training on 70% of the training set and testing on the left 30%. The C and σ that lead to best accuracy were selected. We did not scale the original samples to range [-1,1], because we found that doing so did not help much. Our FR-SVM were also trained with exactly the same parameters (C and σ) and conditions as OVO SVM except that we varied additional two parameters P (the number of principal components) and F (the number of top ranked features) to see the performance. We compared the performance of OVO SVM and FR-SVM basically in three aspects: classification accuracy, training and testing speed.

4.1 PCA Dimension Reduction First experiment we did is: vary P and let $F = P$, which means PCA is applied alone without feature selection. Since PCA reduces dimensions before SVM training and testing, thus PCA is able the enhance both the training and testing speed. Table 2 summarizes the results on the Iris dataset. The first line with $\emptyset$ means without PCA transformation, i.e., it shows the results of standard OVO SVM. The execution time of PCA is separated

Table 2: The results of FR-SVM on the Iris(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of OVO SVM. $C = 500$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	100%	NA	0	0
4	100%	0.016	0	0
3	**100%**	0.015	0	0
2	97.78%	0.015	0	0
1	97.78%	0.015	0	0

from the regular SVM training time. The former is shown in the second column and latter in the third column. Therefore, the total training time of FR-SVM is actually the sum of PCA and SVM training.

Since the Iris is very small, the contribution of PCA on training and testing time is not obvious (all 0). Only PCA transformation itself costs some time. However, the interesting observation is that the accuracy remains perfect with dimension reduction as 0 ($P = 4$) or 1 ($P = 3$) and even only 1 component ($P = 1$) can issue a 97.78% accuracy.

The results on the MNIST and Isolet are shown in Table 3 and 4 respectively. It is surprising that the PCA dimension reduction enhances the accuracy of SVM on the MNIST dataset with a proper value of P. When $P = 50$, the accuracy of FR-SVM is 98.30%, while that of OVO SVM is 97.90%. Although the enhancement is not significant, it is still encouraging. When $P = 25$, a speedup of 11.5 (278.9/24.3) in testing and 6.8 (4471.6/(228.6+441.4)) in training is achieved. The speedups are in our expectation, since the dimensions of samples are reduced before SVM training and testing. The results on the Isolet are similar, as shown in table 4. The difference from MNIST is that the accuracy of FR-SVM decreases as P decreases, but not significantly before $P = 100$. The computational time of training and testing is saved dramatically by PCA dimension reduction. The interesting observation on Isolet is that training time is reduced from 249.4(secs) to 86.5(secs) with only PCA transformation but no dimension reduction (when $P = 617$). The reason is unknown and maybe can be traced into the implementation of the SVM optimization toolbox. From all table 2, 3 and 4, we can see that the accuracy of SVM remains the same by PCA transformation but without dimension reduction (when P =the number of original dimensions). This confirms our theorem 3.1.

Table 3: The results of FR-SVM on the MNIST(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of OVO SVM. $C = 5$ and $\sigma = 20$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	97.90%	NA	4471.6	278.9
784	97.90%	258.6	4467.4	275.7
500	97.84%	255.8	4230.7	250.1
300	97.58%	252.4	3800.8	237.2
200	97.92%	243.6	3650.2	231.9
150	97.96%	237.5	2499.2	175.7
100	98.20%	234.3	1529.6	102.2
50	**98.30%**	230.1	628.8	**48.8**
25	**97.94%**	228.6	441.4	**24.3**
10	92.76%	228.0	128.7	12.8

Table 4: The results of FR-SVM on the Isolet(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of OVO SVM. $C = 10$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	96.92%	NA	**249.4**	36.7
617	96.92%	21.2	86.5	36.6
400	96.86%	19.7	66.1	24.5
300	96.86%	19.3	57.9	18.4
200	96.60%	18.6	43.5	12.7
150	96.54%	18.3	38.3	9.3
100	**96.28%**	18.0	33.4	**6.4**
50	95.51%	18.0	24.7	3.6
25	93.39%	18.0	20.3	2.3
10	81.21%	17.4	12.1	2.0

4.2 RFE Feature Selection The second experiment we did is to vary F without PCA transformation. Since the feature elimination procedure requires many iterations, we fixed the number of features eliminated each time as 1 on Iris, and 20 on MNIST and Isolet empirically. The accuracy, time of RFE, training time and testing time were reported, as shown in table 5, 6 and 7 on the three datasets respectively. The first line shows the results of OVO SVM actually, because no feature is eliminated. Like the previous experiment, the execution time of RFE is also separated from the regular SVM training time. The total training time of FR-SVM here is actually the sum of RFE (the second column) and SVM training (the third column).

On the Iris, only RFE requires some time, since the size of dataset is very small. The interesting thing is that only one feature is enough to distinguish a pair of Iris classes (When $F = 1$, the accuracy is still 100%), as shown in table 5.

Table 5: The results of FR-SVM on the Iris(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 500$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

F	Accuracy (secs)	RFE	Training (secs)	Testing (secs)
4	100%	0	0	0
3	100%	0.016	0	0
2	100%	0.016	0	0
1	**100%**	0.016	0	0

Table 6: The results of FR-SVM on the MNIST(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 5$ and $\sigma = 20$ for both OVO SVM and FR-SVM.

F	Accuracy	RFE (secs)	Training (secs)	Testing (secs)
784	97.90%	0	4471.6	278.9
500	97.87%	35306	3179.7	251.1
300	97.82%	45120	2535.3	243.6
200	97.74%	47280	1094.4	221.0
150	97.40%	49344	648.0	114.1
100	96.62%	**49432**	398.7	110.6
50	94.56%	49440	286.2	47.1
25	89.86%	49446	208.8	25.5
10	75.60%	52126	171.9	13.5

On the MNIST, the accuracy steadily decreases as F decreases, but not significantly. Compared to PCA(as shown in table 3), the RFE seems not as reliable as PCA on MNIST in term of accuracy, while the speedup gained in testing is quite close. In addition, we can see that RFE is very computationally expensive because of its recursive iterations. To eliminate 684 features (let $F = 100$), the number of iterations is 34 (284/20, 20 features eliminated at a time), which takes 13.7 hours (49432 secs). Without PCA, the procedure of RFE is painfully long. From the results on the Isolet as shown in table 7, we have the similar observations.

4.3 Combination of PCA and RFE The third experiment was to see how the combination of PCA and RFE contributes to multi-class SVM classification. We chose the minimum P in the first experiment which guarantees the accuracy is as high as that of the standard SVM. Then we chose F as small as possible that also issues a comparable accuracy. The parameters C and σ remained the same as previous experiments. Table 8 summarizes the results. Training speedup is calculated as $\frac{\text{Training time of OVO SVM}}{\text{Training time of FR-SVM}}$. Note that the

Table 7: The results of FR-SVM on the Isolet(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 10$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

F	Accuracy	RFE (secs)	Training (secs)	Testing (secs)
617	96.92%	0	249.4	36.7
400	96.98%	758.4	80.2	22.3
300	96.86%	868.6	50.0	15.7
200	96.60%	981.1	30.0	12.4
150	96.73 %	1065.3	23.6	11.5
100	96.28 %	1164.0	12.8	5.3
50	96.09 %	1284.8	4.6	3.1
25	95.2 %	1301.8	3.9	3.0
10	93.14 %	1314.7	2.6	2.8

Table 8: Combination of PCA and RFE. P is the number of principal components chosen. F is the number of top ranked features chosen for each pair of classes. Speedup is FR-SVM vs. OVO SVM.

Dataset	P	F	Accuracy	Training Speedup	Testing Speedup
Iris	3	3	**100%**	1	1.3
MNIST	50	40	**98.14%**	3.90	**10.9**
Isolet	200	60	**96.28%**	0.98	**11.1**

training time of FR-SVM is actually the sum of PCA, RFE, and SVM training. Similarly, the evaluation speedup is $\frac{\text{Testing time of OVO SVM}}{\text{Testing time of FR-SVM}}$. On the Iris dataset, the training speedup of FR-SVM over OVO SVM is 1 because both execution time are negligible. On the MNIST, FR-SVM achieved a speedup in both training and testing. The gain in training is due to the dimension reduction by PCA. The gain in testing is due to the combinatorial contribution of PCA and RFE. On the Isolet, the training of FR-SVM is slightly slower than OVO SVM because of RFE (with training speedup as 0.98). However, in testing, we can also see a significant enhancement by an order of over 10 while the accuracy is still comparable. When $P = 50$ and $F = 40$, the accuracy of FR-SVM on MNIST is 98.14%, higher than that of OVO SVM (97.90%). When $P = 200$ and $F = 60$, the accuracy of FR-SVM on Isolet is 96.28%, slightly lower than that of OVO SVM (96.92%). To sum up, PCA and RFE can significantly enhance the evaluation speed of standard SVM with proper settings of P and F while maintaining comparable accuracy.

5 Conclusion

Incorporating both PCA and RFE into standard SVM, we propose FR-SVM for efficient multi-class classification. PCA and RFE reduce dimensions and select the most discriminative features. Choosing a proper number of principle components and a number of top ranked features for each pairwise classes, a significant enhancement in evaluation can be achieved while comparable accuracy is maintained.

References

[1] B. Boser, I.Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

[2] C. Burges and B. Schölkopf. Improving speed and accuracy of support vector learning machines. In *Advances in Kernel Methods: Support Vector Learnings*, pages 375–381, Cambridge, MA, 1997. MIT Press.

[3] C.Chang and C. Lin. Libsvm: a library for support vector machines. *http://www.kernel-machines.org/*, 2001.

[4] T. Downs, K. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, vol. 2:293–297, 2001.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, vol. 3:1157–1182, 2003.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, vol. 46:389–422, 2002.

[7] S. Hettich and S. Bay. The UCI KDD archive. *http://kdd.ics.uci.edu*, 1999.

[8] U. Kreßel. Pairwise classification and support vector machines. In *Advances in Kernel Methods: Support Vector Learnings*, pages 255–268, Cambridge, MA, 1999. MIT Press.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86:2278–2324, Nov 1998.

[10] Y. Lee and O. Mangasarian. Rsvm: Reduced support vector machines. In *The First SIAM International Conference on Data Mining*, 2001.

[11] K.-M. Lin and C.-J. Lin. Lin a study on reduced support vector machines. *IEEE Transactions on Neural Networks*, vol. 14:1449–1559, 2003.

[12] J. Ma, Y. Zhao, and S. Ahalt. Osu svm classifier matlab toolbox. *http://www.kernel-machines.org/*, 2002.

[13] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[14] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 2000.

[15] R. Rifin and A. Klautau. In defense of one vs-all-classification. *Journal of Machine Learning Research*, vol. 5:101–141, 2004.

[16] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[17] H. Yu. *Data Mining via Support Vector Machines: Scalability, Applicability, and Interpretability*. PhD thesis, Univeristy of Illinois at Urbana-Champaign, May 2004.

Detecting Outlying Subspaces for High-Dimensional Data: A Heuristic Search Approach

Ji Zhang
Department of Computer Science,
University of Toronto, Canada
jzhang@cs.toronto.edu

Abstract

In this paper, we identify a new task for studying the outlying degree of high-dimensional data, i.e. finding the subspaces (subset of features) in which given points are outliers, and propose a novel detection algorithm, called High-D Outlying subspace Detection (HighDOD). We measure the outlying degree of the point using the sum of distances between this point and its k nearest neighbors. Heuristic pruning strategies are proposed to realize fast pruning in the subspace search and an efficient dynamic subspace search method with a sample-based learning process has been implemented. Experimental results show that HighDOD is efficient and outperforms other searching alternatives such as the naive top-down, bottom-up and random search methods.

Keywords: Outlying Subspaces, High-dimensional Data, Heuristic Search, Sample-based Learning.

1 Introduction

Outlier detection is a classic problem in data mining that enjoys a wide range of applications such as the detection of credit card frauds, criminal activities and exceptional patterns in databases. Outlier detection problem can be formulated as follows: Given a set of data points or objects, find a specific number of objects that are considerably dissimilar, exceptional and inconsistent with respect to the remaining data [5].

Numerous research works in outlier detection have been proposed to deal with the outlier detection problem defined above. They can broadly be divided into distance-based methods [7], [8], [11] and local density-based methods [4], [6], [10]. However, many of these outlier detection algorithms are unable to deal with high-dimensional datasets efficiently as many of them only consider outliers in the entire space. This implies that they will miss out the important information about the subspaces in which these outliers exist.

A recent trend in high-dimensional outlier detection is to use the evolutionary search method [2] where outliers are detected by searching for sparse subspaces. Points in these sparse subspaces are assumed to be the outliers. While knowing which data points are the outliers can be useful, in many applications, it is more important to identify the subspaces in which a given point is an outlier, which motivates the proposal of a new technique in this paper to handle this new task.

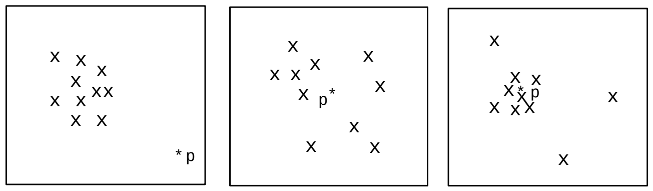


Figure 1: 2-dimensional views of the high-dimensional data

To better demonstrate the motivation of exploring outlying subspace detection, let us consider the example in Figure 1, in which three 2-dimensional views of the high-dimensional data are presented. Note that point p exhibits different outlying degrees in these three views. In the leftmost view, p is clearly an outlier. However, this is not so in the other two views. Finding the correct subspaces so that outliers can be detected is informative and useful in many practical applications. For example, in the case of designing a training program for an athlete, it is critical to identify the specific subspace(s) in which an athlete deviates from his or her teammates in the daily training performances. Knowing the specific weakness (subspace) allows a more targeted training program to be designed. In a medical system, it is useful for the Doctors to identify from voluminous medical data the subspaces in which a particular patient is found abnormal and therefore a corresponding medical treatment can be provided in a timely manner.

The major contribution of this paper is the proposal of *a dynamic subspace search algorithm, called HighDOD, that utilizes a sample-based learning process to* *efficiently identify the subspaces in which a given point is an outlier*. Note that, instead of detecting outliers in specific subspaces, our method searches from the space lattice for the associated subspaces whereby the given data points exhibit abnormal deviations. To our best knowledge, this is the first such work in the literature so far. The main features of HighDOD include:

1. The outlying measure, OD, is based on the sum of distances between a data and its k nearest neighbors [1]. This measure is simple and independent of any underlying statistical and distribution characteristics of the data points;

2. Heuristic pruning strategies are proposed to aid in the search for outlying subspaces;

3. A fast dynamic subspace search algorithm with a sample-based learning process is proposed;

4. The heuristic on the minimum sample size based on the hypothesis testing method is also presented.

The reminder of this paper is organized as follows. Section 2 discusses the basic notions and problem to be solved. In Section 3, we present our outlying subspace detection technique, called HighDOD, for high-dimensional data. Experimental results are reported in Section 4. Section 5 concludes this paper.

2 Outlying Degree Measure and Problem Formulation

Before we formally discuss our outlying subspace detection technique, we start with introduction of the outlying degree measure that will be used in this paper and formulation of the new problem of outlying subspace detection we identify.

2.1 Outlying Degree OD. For each point, we define the degree to which the point differs from the majority of the other points in the same space, termed the *Outlying Degree* (*OD* in short). OD is defined as the sum of the distances between a point and its k nearest neighbors in a data space [1]. Mathematically speaking, the OD of a point p in space s is computed as:

$$OD_s(p) = \sum_{i=1}^{k} Dist(p, p_i) | p_i \in KNNSet(p, s)$$

where $KNNSet(p, s)$ denotes the set composed by the k nearest neighbors of p in s. Note that the outlying degree measure is applicable to both numeric and nominal data: for numeric data we use Euclidean distance while for nominal data we use the simple match method. Mathematically, the Euclidean distance between two numeric points p_1 and p_2 is defined as $Dist(p_1, p_2) = [\sum((p_{1i} - p_{2i})/(Max_i - Min_i))^2]^{1/2}$, where Max_i and Min_i denote the maximum and minimum data value of the i^{th} dimension. The simple match method measures the distance between two nominal points p_1 and p_2 as $Dist(p_1, p_2) = \sum |p_{1i} - p_{2i}|/t$, where $|p_{1i} - p_{2i}|$ is 0 if p_{1i} equals to p_{2i} and is 1 otherwise. t is the total number of attributes.

2.2 Problem Formulation. We now formulate the new problem of outlying subspace detection for high-dimensional data as follows: given a data point or object, find the subspaces in which this data is considerably dissimilar, exceptional or inconsistent with respect to the remaining points or objects. These points under study are called *query points*, which are usually the data that users are interested in or concerned with.

A distance threshold T is utilized to decide whether or not a data point deviates significantly from its neighboring points. We call a subspace s is an outlying subspace of data point p if $OD_s(p) \geq T$.

2.3 Applicability of Existing High-dimensional Outlier Detection Techniques. The existing high-dimensional outlier detection techniques, i.e. find outliers in given subspaces, are theoretically applicable to solve the new problem identified in this paper. To do this, we have to detect outliers in all subspaces and a searching in all these subspaces is needed to find the set of outlying subspaces of p, which are those subspaces in which p is in their respective set of outliers. Obviously, the computational and space costs are both in an exponential order of d, where d is the number of dimensions of the data point. Such an exhaustive space searching is rather expensive in high-dimensional scenario. In addition, they usually only return the top-k outliers in a given subspace, thus it is impossible to check whether or not p is an outlier in this subspace if p is not in this top-k list. This analysis provides an insight into the inherent difficulty of using the existing high-dimensional outlier detection techniques to solve the new outlying subspace detection problem.

3 HighDOD

In this section, we present an overview of our High-Dimension Outlying subspace Detection (HighDOD) method (shown in Figure 2). It mainly consists of three modules. The X-tree Indexing module performs X-tree [3] indexing of the high-dimensional dataset to facilitate kNN search in every subspace. Sample-based Learning module randomly samples the dataset and performs dynamic subspace search to estimate the downward and

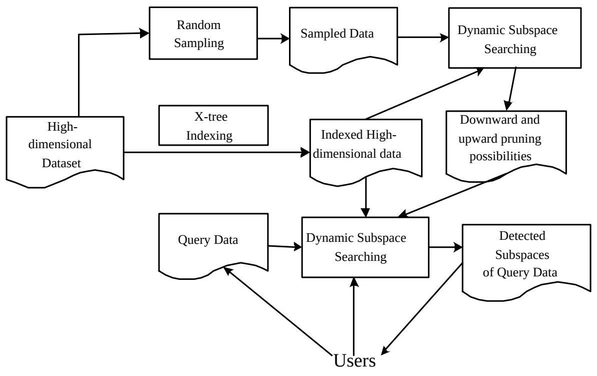

Random Sampling
Sampled Data
Dynamic Subspace Searching
High-dimensional Dataset
X-tree Indexing
Indexed High-dimensional data
Downward and upward pruning possibilities
Query Data
Dynamic Subspace Searching
Detected Subspaces of Query Data
Users

Figure 2: The overview of HighDoD

upward pruning probabilities of subspaces from 1 to d dimensions. Outlying Subspace Detection module uses the probabilities obtained in the Learning module to carry out a dynamic subspace search to find the outlying subspaces of the given query data point.

3.1 Subspace Pruning. To find the outlying subspaces of a query point, we make use of the heuristics we devise to quickly detect the subspaces in which the point is not an outlier or the subspaces in which the point is an outlier. All these subspaces can be removed from further consideration in the later stage of the search process.

In our work, we utilize a distance threshold T is used for delimiting outlying and non-outlying subspaces in the space lattice for a query data point.

OD maintains two interesting monotonic properties that allow the design of an efficient outlying subspace search algorithm.

Property 1: If a point p is not an outlier in a subspace s, then it cannot be an outlier in any subspace that is a subset of s.

Property 2: If a point p is an outlier in a subspace s, then it will be an outlier in any subspace that is a superset of s.

The above properties are based on the fact that the OD value of a point in a subspace cannot be less than that in its subset spaces. Mathematically, we have $OD_{s_1}(p) \geq OD_{s_2}(p)$ if $s_1 \supseteq s_2$.

Proof: Let a_k and b_k be the k^{th} nearest neighbors of p in the an m-dimensional subspace s_1 and n-dimensional subspaces s_2, respectively ($1 \leq n \leq m \leq d$ and $s_1 \supseteq s_2$). $MaxDist_{s2}(p)$ is the maximum distance between p and a_i, $1 \leq i \leq k$, in the subspace s_2.

We have $Dist_{s1}(p, a_k) \geq Dist_{s1}(p, a_i)|_{1 \leq i \leq k}$. Since s_1 is a superset of s_2, we thus know $Dist_{s1}(p, a_i) \geq Dist_{s2}(p, a_i)|_{1 \leq i \leq k}$. This implies $Dist_{s1}(p, a_k) \geq Dist_{s2}(p, a_i)|_{1 \leq i \leq k}$. By definition of $MaxDist_{s2}$, we have $Dist_{s_1}(p, a_k) \geq MaxDist_{s2}(p) \geq Dist_{s2}(p, b_k)$. In other words, $Dist_{s1}(p, a_k) \geq Dist_{s2}(p, b_k)$. Likewise, it is hold that $Dist_{s1}(p, a_i) \geq Dist_{s2}(p, b_i)|_{1 \leq i \leq k}$, Since $OD_{s1}(p) = \sum_1^k Dist_{s1}(p, a_i)$ and $OD_{s2}(p) = \sum_1^k Dist_{s2}(p, b_i)$. We therefore conclude: $OD_{s1}(p) \geq OD_{s2}(p)$. ■

We make use of Property 1 of OD to quickly prune away those subspaces in which the point cannot be an outlier. This is because if $OD_{s1}(p) < T$, then $OD_{s2}(p) < T$, where $s_1 \supseteq s_2$ and T is the distance threshold. In the upward pruning strategy, Property 2 of OD is utilized to detect those subspaces in which the point is definitely an outlier. The reason is that if $OD_{s2}(p) \geq T$, then $OD_{s1}(p) \geq T$.

The distance threshold T is specified as follows:

$$T = C\sqrt{\sum_{i=1}^{d} \overline{OD_{s_i}}^2}, \text{where}\, dim(s_i) = 1$$

where $\overline{OD_{s_i}}$ denotes the averaged OD value of points in the 1-dimensional subspace s_i and C is a constant factor ($C > 1$). This specification stipulates that, in any subspace, only those points whose OD values are significantly larger than the average level in the full space are regarded as outliers. The average OD level in the full space is approximated by $\sqrt{\sum_{i=1}^{d} \overline{OD_{s_i}}^2}$ and the significance of deviation is specified by the constant factor C, normally we set C=2 or 3.

3.2 Saving Factors of Subspaces Pruning. Now, we will compute the savings obtained by applying the pruning strategies during the search process quantitatively. Before that, let us first give three definitions.

Definition 1: *Downward Saving Factor (DSF) of a Subspace*

The Downward Saving Factor of a m-dimensional subspace s is defined as the savings obtained by pruning all the subspaces that are subsets of s. In other words, the Downward Saving Factor of s, denoted as DSF(s), is computed as $DSF(s) = \sum_{i=1}^{m-1} C_m^i * i$, where C_m^i denotes the combinatorial number of choosing i items out of a total of m items.

Definition 2: *Upward Saving Factor (USF) of a Subspace*

The Upward Saving Factor of an m-dimensional subspace s, denoted as USF(s), is defined as the savings obtained by pruning all the subspaces that are supersets of s. It is computed as $USF(s) = \sum_{i=1}^{d-m}[C_{d-m}^i * (m+i)]$.

Definition 3: *Total Saving Factor (TSF) of a Subspace*

The Total Saving Factor of a m-dimensional subspace, in terms of a query point p, denoted as TSF(m, p), is defined as the combined savings obtained by applying the two pruning strategies during the search process. It is computed as follows:

$TSF(m,p) = pr_{up}(m,p) * f_{up}(m) * USF(m)$, when $m = 1$;
$TSF(m,p) = pr_{down}(m,p) * f_{down}(m) * DSF(m)$
$+ pr_{up}(m,p) * f_{up}(m) * USF(m)$, when $1 < m < d$;
$TSF(m,p) = pr_{down}(m,p) * f_{down}(m) * DSF(m)$, when $m = d$.

where
(1) $f_{down}(m)$ and $f_{up}(m)$ are the percentages of the remaining subspaces to be searched. specifically, $f_{down}(m) = C_{down\_left}(m)/C_{down}(m)$ and $f_{up}(m) = C_{up\_left}(m)/C_{up}(m)$

Let $dim(s)$ denote the number of dimensions for subspace s. $C_{down\_left}(m)$ and $C_{up\_left}(m)$ are computed as: $C_{down\_left}(m) = \sum dim(s)$, where s is an unpruned or unevaluated subspace and $dim(s) < m$. $C_{up\_left}(m) = \sum dim(s)$, where s is an unpruned or unevaluated subspace and $dim(s) > m$.

$C_{down}(m)$ and $C_{up}(m)$ are the total subspace search workload in the subspaces whose dimensions are lower and higher than m, respectively. Intuitively, $f_{down}(m)$ and $f_{up}(m)$ approximate the fraction of DSF and USF of an m-dimensional subspace that are potentially achievable in each step of the search process.

(2) $pr_{up}(m,p)$ and $pr_{down}(m,p)$ are the probabilities that upward and downward pruning can be performed in the m-dimensional subspace, respectively. In other words, for a m-dimensional subspace s, $pr_{up}(m,p) = Pr(OD_s(p) \geq T)$ and $pr_{down}(m,p) = Pr(OD_s(p) < T)$. A difficulty in computing the two prior probabilities, i.e. $pr_{up}(m,p)$ and $pr_{down}(m,p)$, is that their values are unknown if there lacks any prior knowledge of the dataset. To overcome this difficulty, we first perform a sample-based learning process to obtain some knowledge about the dataset and then apply this knowledge in the later subspace search for each query point.

3.3 Sampling-based Learning. We adopt a sample-based learning process to obtain some knowledge about the dataset before subspace search of the query points are performed. This is desirable when the dataset is large so that learning the whole dataset becomes prohibitive. The task of performing this sampling-based learning is two-fold: first, we will have to estimate $\overline{OD_{s_i}}$ which will be used in specifying the distance threshold. Secondly, we will have to compute the two priors $pr_{up}(m,p)$ and $pr_{down}(m,p)$. In this learning process, a small number of points are randomly sampled from the dataset.

At first, the subspace searches are performed in the d 1-dimensional subspaces s_i on all the sampling data and $\overline{OD_{s_i}}$ is computed as the average OD values of all sampling points in subspace s_i, i.e.

$$\overline{OD_{s_i}} = \frac{1}{S}\sum_{j=1}^{S} OD_{s_i}(sp_j)$$

where S is the number of sampling points and sp_j denotes the i^{th} sampling point.

Secondly, the subspace searches are performed in the lattice of data space on the sampling data. For each sampling point sp, we have the following initial specifications regarding the two priors $pr_{up}(m,p)$ and $pr_{down}(m,p)$:

$$pr_{up}(m,sp) = pr_{down}(m,sp) = 0.5, 1 < m < d$$
$$pr_{up}(m,sp) = 1 \text{ and} pr_{down}(m,sp) = 0, m = 1$$
$$pr_{up}(m,sp) = 0 \text{ and } pr_{down}(m,sp) = 1, m = d$$

This initialization implies that we assume equal probabilities for upward and downward pruning in the subspaces of any dimension, except 1 and d, for each sampling point at the beginning. After all the m dimensional subspaces have been evaluated for sp, the $pr_{up}(m,sp)$ and $pr_{down}(m,sp)$ are computed as the percentages of m-dimensional subspaces s in which $OD_s(sp) \geq T$ and $OD_s(sp) < T$, respectively. The average pr_{up} and pr_{down} values of subspaces from 1 to d dimensions can be obtained as follows:

$$\overline{pr_{up}(m)} = \tfrac{1}{S}\sum_{i=1}^{S} pr_{up}(m,sp_i)$$
$$\overline{pr_{down}(m)} = \tfrac{1}{S}\sum_{i=1}^{S} pr_{down}(m,sp_i)$$

where we have $\overline{pr_{down}(1)} = \overline{pr_{up}(d)} = 0$.

For each query point p, we set $pr_{up}(m,p) = \overline{pr_{up}(m)}$ and $pr_{down}(m,p) = \overline{pr_{down}(m)}$ in the computation of TSF(m, p) of the query point p.

Remarks: There might be a misunderstanding that the sampling technique will fail here because the outliers are rare in the dataset. Recall that we are trying to detect outlying subspaces of query points, not outliers. Every point can become query point and every query point will have its outlying subspaces, if its set of outlying subspaces is not empty. Hence, the outlying subspaces can be regarded as a global property for all the points and a sample of sufficient size will make sense in the learning process.

3.4 Dynamic Subspace Search. In HighDOD, we use a dynamic subspace search method to find the subspaces in which the sampling points and the query points are outliers. The basic idea of the dynamic subspace search method is to commence search on those subspaces with the same dimension that has the highest TSF value. As the search proceeds, the TSF of subspaces with different dimensions will be updated and the set of subspaces with the highest TSF values are selected for exploration in each subsequent step. The search process terminates when all the subspaces have been evaluated or pruned. Note that the only difference between the dynamic subspace search method used on the sample points and query points lies in the decision of values of $pr_{up}(m, p)$ and $pr_{down}(m, p)$: *For sample points, we assume an equal probability of upward and downward pruning while for query points we use the averaged probabilities obtained in the learning process.*

3.5 Minimum Sampling Size for Training Dataset. Recall that the sampling method is utilized to obtain a training dataset that can be used to pre-compute the prior probabilities of upward and downward pruning, namely $\overline{pr_{up}(m)}$ and $\overline{pr_{down}(m)}$ $(1 \leq m \leq d)$. As such, samples of different sizes will only affect the pruning efficiency of the algorithm. They will not change the number of subspaces found.

With this in mind, we now wish to determine the minimum sample size to accurately predict $\overline{pr_{up}(m)}$ and $\overline{pr_{down}(m)}$ with certain degree of confidence. We denote X as the sample point that can be expressed as an S-dimensional vector as $X = [x_1, x_2, \ldots, x_S]$ where S is the size of the sample. Each data in the sample is a d-dimensional vector as $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]^T$ where $x_{i,j}$ denote the value of j^{th} dimension of i^{th} data in the sample. Applying dynamic subspace searching on sampling points, for each dimension m, we obtain

$$Y_{down}(m) = [pr_{down}(m, sp_1), pr_{down}(m, sp_2), \ldots, \\ pr_{down}(m, sp_S)] \quad (1 \leq m \leq d)$$

We use the S measurements, $pr_{down}(m, sp_i)(1 \leq i \leq S)$ as the training data to estimate the mean of $pr_{down}(m)$. We estimate the sample size by constructing the confidence interval of the mean of $pr_{down}(m)$. Specifically, to obtain a $(1-\alpha)$-confidence interval, the minimum size of a random sample is given as follows [9]:

$$S_{min}(m) = [\frac{t_{\alpha/2} * \sigma_m^{'}}{\delta^*}]^2$$

where $\sigma_m^{'}$ denotes the estimated standard deviation of pr_{down} in the m^{th} dimension using the training points that is defined as:

$$\sigma_m^{'} = \sqrt{\sum_{i=1}^{S}(pr_{down}(m, sp_i) - \overline{pr_{down}(m, sp)})^2/(S-1)}$$

δ^* denotes the half-width of the confidence interval.

Note that the value of $\sigma_m^{'}$ varies for different m. Let $\sigma_{max}^{'} = max(\sigma_m^{'})(1 \leq m \leq d)$, the minimum sample size S_{min} that satisfies respective minimum sample size requirement of each dimension is computed as:

$$S_{min} = [\frac{t_{\alpha/2} * \sigma_{max}^{'}}{\delta^*}]^2$$

Similarly reasoning applies to $\overline{pr_{up}(m)}$ since $\overline{pr_{up}(m)}$= 1- $\overline{pr_{down}(m)}$.

4 Experimental Results

In this section, we will carry out extensive experiments to test the efficiency of outlying subspace detection and the effectiveness of outlying subspace compression in HighDOD. Synthetic datasets are generated using a high-dimensional dataset generator and four real-life high-dimensional datasets from the UCI machine learning repository, which have been used in [2] for performance evaluation of their high-dimensional outlier detection technique, are also used.

Since the existing high-dimensional outlier detection techniques fail to handle the new outlying subspace detection problem, we thus choose to compare the efficiency of several subspace search methods, i.e. top-down, bottom-up, random and dynamic subspace search, instead.

These searching methods aim to find the outlying subspaces of the given query data using various searching strategies. The top-down search method only employs a downward pruning strategy while the bottom-up search method only uses an upward pruning strategy. The random search method, the "headless chicken" search alternative, randomly selects the layer in the lattice for search without replacement in each step. The dynamic search method, a hybrid of upward and downward search, computes the TSF of all subspaces of different dimensions and selects the best layer of subspaces for search. To evaluate the efficiency of the sample-based learning process , we run the dynamic search algorithm with and without incorporating the sample-based learning process. Note that the execution times shown in this section are the average time spent in processing each point in the learning and query process.

Effect of Dimensionality. First, we investigate the effect of dimensions on the average execution time of HighDOD (see Figure 3) . We can see that the execution time of all the five methods increase at an exponential rate since the number of subspaces increases exponentially as the number of dimension goes up, regardless of which searching and pruning strategy is utilized. On a closer examination, we see that (1) The execution

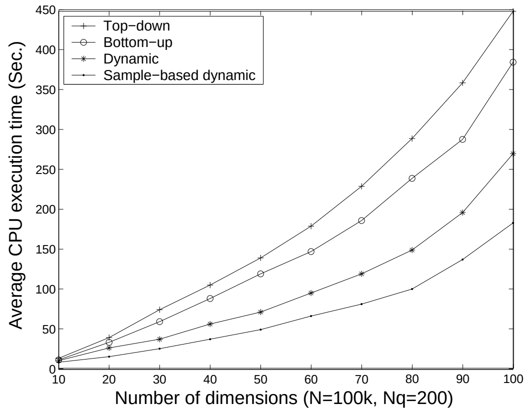


Figure 3: Execution time when varying dimension of data

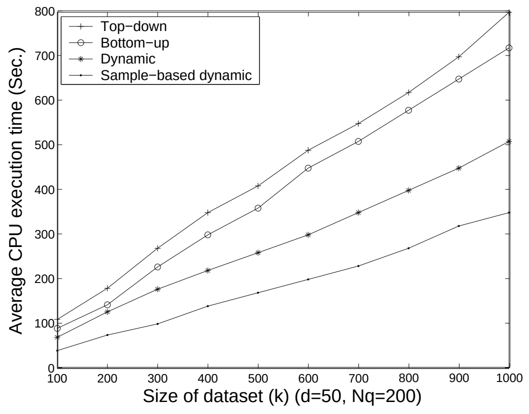


Figure 4: Execution time when varying size of dataset

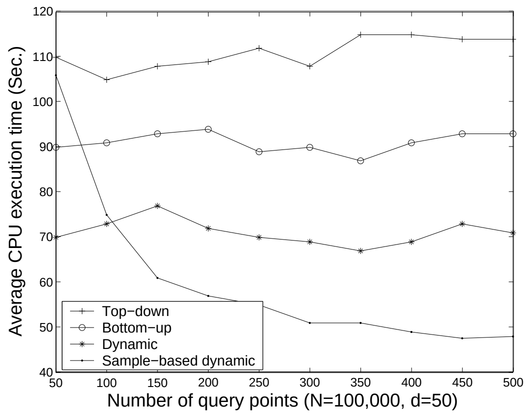


Figure 5: Execution time when varying the number of query points

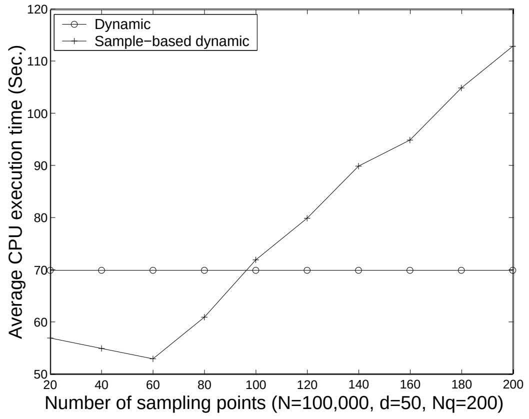


Figure 6: Execution time when varying the size of sample

time of top-down and bottom-up search methods increase much faster than the dynamic search method; (2) When using the sample-based learning process, the dynamic search method performs better than the method without using the sample-based learning process.

Effect of Dataset Size. Second, we fix the number of dimensions at 50 and vary the size of datasets from $100k$ to $1{,}000k$. Figure 4 shows that the average execution times using the five methods to process each query point are approximately linear with respect to the size of the dataset. Similar to results of the first experiment, the dynamic search method with sample-based learning process gives the best performance.

Effect of Number of Query Points. Next, we vary the number of query points N_q. Figure 5 shows the results of the five searching methods. It is interesting to note that when N_q is large, dynamic search method with sample-based learning process gives the best performance. However, when N_q is small, it is better to use dynamic search without sample-based learning. The reason is because when the number of query points is small, the saving in computation by using the learning process is not sufficient to justify the cost of the learning process itself.

Effect of Sample Size. We also investigate the effect of the number of sampling points, S, used in the learning process. A large S gives a more accurate estimation of the possibilities of upward and downward pruning in subspaces, which in turn, helps to speedup the search process. However, a large S also implies an increase in the computation during the learning process, which may increase the average time spent in the whole detection process. As shown in Figure 6, the execution time is first decreased when the number of sampling points is small, this is because the prediction of possibility is not accurate enough, which cannot

Datasets(dimensions)	Top-down	Bottom-up	Random	Dynamic	Sample+ Dynamic
Machine(8)	56	49	58	41	32
Breast Cancer (14)	165	176	150	121	110
Segmentation (19)	251	237	256	222	197
Ionosphere (34)	472	477	456	414	387
Musk (160)	5203	4860	5002	4389	3904

Table 1: Results of running five methods on real-life datasets (average CPU time in seconds for each query point)

greatly speedup the later searching process. When the sample size increases, the prediction of the possibilities are sufficiently accurate, therefore any larger size of sample will no longer contribute to the speedup of the search process, but only increase the execution time as a whole. The horizontal dot-line in Figure 6 indicates the execution time when dynamic subspace search without sample-based learning is employed.

Results on Real-life Datasets. Finally like [2], we evaluate the practical relevance of HighDOD by running experiments on five real-life high-dimensional datasets in the UCL machine learning repository. The datasets range from 8 to 160 dimensions. Table 1 shows the results of the five search methods. It is obvious that dynamic search with sampling-based learning process works best in all the real-life datasets. Furthermore, using dynamic subspace search alone is faster than top-down bottom-up or random search methods by approximately 20% while incorporating sample-based learning process into dynamic subspace search further reduces the execution time by about 30%.

5 Conclusions

In this paper, we propose a novel algorithm, called HighDOD, to address the new problem of detecting outlying subspaces for high-dimensional data. In HighDOD, heuristics for fast pruning in the subspace search and a dynamic subspace search method with a sample-based learning process are used. Experimental results justify the efficiency of outlying subspace searching in HighDOD. We believe that HighDOD is useful in revealing interesting and new knowledge in outlying analysis of high-dimensional data and can be potentially used in many practical applications.

References

[1] F. Angiulli and C. Pizzuti. Fast Outlier Detection in High Dimensional Spaces. Proc. *PKDD'02*,Helsinki, Finland, 2002.

[2] C. C Aggarwal and P.S. Yu. Outlier Detection in High Dimensional Data. Proc. *ACM SIGMOD'00*, Santa Barbara, California, 2001.

[3] S. Berchtold, D. A. Keim and H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. Proc. *VLDB'96*, Mumbai, India, 1996.

[4] M. Breuning, H-P, Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. Proc. *ACM SIGMOD'00*, Dallas, Texas, 2000.

[5] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2000.

[6] W. Jin, A. K. H. Tung, J. Han. Finding Top n Local Outliers in Large Database. Proc. *SIGKDD'01*, San Francisco, CA, August, 2001.

[7] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-based Outliers in Large Dataset. Proc. *VLDB'98*, pages 392-403, New York, NY, August 1998.

[8] E. M. Knorr and R. T. Ng. Finding Intentional Knowledge of Distance-based Outliers. Proc. *VLDB'99*, pages 211-222, Edinburgh, Scotland, 1999.

[9] A. E. Mace. *Sample-size Determination*. Reinhold Publishing Corporation, New York, 1964.

[10] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos: LOCI: Fast Outlier Detection Using the Local Correlation Integral. Proc. *ICDE'03*, pages 315, Bangalore, India, 2003.

[11] S. Ramaswamy, R. Rastogi, and S. Kyuseok. Efficient Algorithms for Mining Outliers from Large Data Sets. Proc. *ACM SIGMOD'00*, Dallas, Texas, 2000.

An Optimal Binning Transformation for Use in Predictive Modeling

Talbot Michael Katz,
TopKatz@msn.com

Abstract: A new transformation of a continuous-valued predictor for a binary-valued target partitions the range of the predictor variable into separate bins, and assigns to each bin the mean target value of a sample within that bin. The bins are chosen to minimize the sum of squared differences between the actual target values of the sample points (0 or 1) and their assigned values. This transformation is most useful in cases where the variation of the target is non-monotonic (and non-random) with respect to the predictor. The methodology can be used to create a new predictor based on combinations of two or more predictors, and it has extensions to multiple-valued targets, and even continuous-valued targets.

Keywords: Optimal, Binning, Transformation, Predictor, Target

Introduction

A continuous variable may be a good predictor of an outcome, but in cases where its direct correlation with the outcome is low, the predictive ability is obtained only after performing a transformation of the original continuous variable. Modelers typically run their continuous variables through a whole suite of transforms (square, cube, exponential, logarithm, cosine, inverse cosine, Box-Cox, etc.) and test them all separately, looking for a fit.

Sometimes a discretization is the best way to pick up essential nonlinearities and exploit the full predictive power of a variable. One simple discretization for classification, as described in [1], breaks the range of the continuous variable into deciles, and assigns to each decile the mean value of the target, or dependent variable, within that decile. This is an example of the practice of binning. A typical definition of binning can be found on the World Wide Web in [2],

"A data preparation activity that converts continuous data to discrete data by replacing a value from a continuous range with a bin identifier, where each bin represents a range of values. For example, age could be converted to bins such as 20 or under, 21-40, 41-65 and over 65."

Two of the most common methods of binning are, from [3]:

- Equal Width, dividing the range of the predictor into contiguous bins of approximately equal distance between endpoints (like the age binning example in the definition above), and
- Equal Depth, dividing the range of the predictor into contiguous bins of approximately equal numbers of sample points (like the deciling described above)

However, there is no guarantee that deciles or uniform intervals pick up the best sub-groupings of the data. For example, imagine a binary outcome and a variable for which the odd demideciles all have outcome zero and the even demideciles all have outcome one. This variable will have low correlation with the outcome, and the deciled average responses will all be 0.5, making the variable appear to be a non-predictor, although it may be a perfect predictor.

Although several data mining software packages, such as KXEN and Oracle9i Data Mining, require binning for some of their modeling algorithms, many standard texts [4],[5],[6] devote little or no discussion to the methods and merits of binning. However, more sophisticated binning transformations than equal-width and equal-depth already exist. SAS Enterprise Miner ® [7], which offers the deciling transform described above (and its extension to more general quantiles), has an algorithm that splits intervals into two pieces at the point where the maximum chi-square is attained, and recursively applies this splitting technique to the subintervals. As with any sequential "greedy" algorithm, there is no guarantee that the final outcome has the highest possible chi-square. Powerhouse^{TM} analytics software [8] offers three information / entropy based methods, including Signal-to-Noise Ratio maximization, Least Information Loss, and Equal Entropy. Some of the theory behind these methods can be found in [9].

Optimal Transformation for Binary Target Variable

The new proposed binning transformation picks out sub-segments of the range of a predictor that have the "most uniform density," i.e., those for which the sum of within group mean square residuals is minimized; then each of these sub-segments is assigned a value equal to the mean value of the target on that sub-segment. This criterion is similar to the chi-square splitting, but allows for a true optimum to be achieved via binary linear integer programming, as follows.

Choose a sample of *N* points, sorted by ascending order of the continuous predictor variable under investigation. Let *x[i]* be the value of the predictor variable and *y[i]* be the corresponding value of the target variable for *1 <= i <= N*. Our goal will be to partition the *N* points into disjoint subsets such that each subset contains a contiguous sequence of all points *k* with *i <= k <= j* for some pair *i* and *j*, i.e., sub-segments or subintervals. Let *m[i,j]* be the mean of the *y[k]* values for *i <= k <= j*, and let *s[i,j]* be the sum of squares of the residuals *(y[k] - m[i,j])* for *i <= k <= j*. Define the binary optimization variables *v[i,j]* for each pair of points *1 <= i <= j <= N*; *v[i,j] = 1* will mean that the sub-segment determined by *i* and *j* has been chosen, otherwise *v[i,j] = 0*. The objective will be to minimize the function *c[i,j] * v[i,j]*, where *c[i,j] = s[i,j] + C* for some constant *C*. The choice of *C* will be critical. If *C = 0*, the optimization will want to put each point in its own sub-segment, because *s[i,i] = 0* (at least, when *x[i]* is unique, which is likely for continuous variables); if *C* is very large, the single segment containing all *N* points will be preferred.

The *v[i,j]* variables are subject to the following conditions / constraints :
(required) This says that every point must be in exactly one sub-segment. For each point *k*, the sum of *v[i,j]* over all sub-segments containing *k (i <= k <= j)* is equal to 1.
(optional) If there is a lower bound, *LG*, on the number of sub-segments, then the sum of *v[i,j]* over all pairs of points *i* and *j* (including *i = j*) is >= *LG*.
(optional) If there is a hard upper bound, *UG*, on the number of sub-segments, then the sum of *v[i,j]* over all pairs of points *i* and *j* (including *i = j*) is <= *UG*.
(optional) If there is a lower bound, *LP*, on the number of points per sub-segment, then eliminate variables *v[i,j]* with *j+1-i < LP*.
(optional) If there is an upper bound, *UP*, on the number of points per sub-segment, then eliminate variables *v[i,j]* with *j+1-i > UP*.

If the sample data is unevenly distributed, it may also be desirable to add constraints to guarantee that the difference between sub-segment endpoint values is bounded below and / or above. Like the bounds on the number of points per sub-segment, bounds on the differences between endpoints serve to eliminate variables.

The choice of the constant *C* creates a soft upper bound of *1 + (s[1,N]/C)* on the number of sub-segments. (The optimization will pick some number of sub-segments no larger than that value.) The “default” value of *C = s[1,N] / (N - 1)* makes the single sub-segment solution and the solution consisting of all individual point sub-segments equally likely.

Optimization Considerations

Because the number of variables and constraints grows with the sample size, the number of points that can be used in a sample is limited by the power of the solver. This method works readily using the SAS ® PROC LP solver with a sample of 100 to 200 points, which should be adequate to pick up the essential behavior of most continuous variables for modeling purposes. From an optimization standpoint, the key feature is that the integer solution is the same as the LP-relaxation.

Comparison Test of “Oscillating” Predictor with Binary Target

The transformation described above has its greatest effect when the target and predictor variables have a non-monotonic relationship. A sample of 101 data points was generated with the following SAS code:

```
%let twopi = %sysevalf(4*%sysfunc(arcos(0)));
%* 2 * pi;
%let ranseed =  59137; %* seed for pseudo
random number generation;
data &_TRA.;
  do x = 0 to 2 by 0.02; * 100 data points;
    y = cos(&twopi.*x)**2;
    d = (y ge 0.8) + ((0.2 < y < 0.8) *
round(ranuni(&ranseed.),1)); * binary target;
    * b is the result of running the optimization
with d as the target and x as the predictor;
    if x < 0.17 then b = 0.77778;
    else if 0.17 <= x < 0.34889 then b = 0;
    else if 0.34889 <= x < 0.56909 then b =
0.90909;
    else if 0.56909 <= x < 0.82960 then b = 0;
    else if 0.82960 <= x < 1.10963 then b =
0.85714;
    else if 1.10963 <= x < 1.41077 then b =
0.06667;
    else if 1.41077 <= x < 1.67263 then b =
0.84615;
    else if 1.67263 <= x < 1.82933 then b = 0;
    else if 1.82933 <= x then b = 0.88889;
    output;
  end;
  stop;
run;
```

The target variable, *d*, oscillates between 0 and 1 in a not entirely deterministic fashion governed by the square of *cos(2*pi*x)* as *x* increases from 0 to 2. *d* takes on the value zero 52 times, and one 49 times. Logistic regressions were run for *d* against *x, y, b,* and *c,* the chi-square binning transformation of *x* with respect to *d*. Here are the log likelihood and confusion matrix results for each case:

Predictor	*-2*log(L)*	*d*=0, pred=0	*d*=0, pred=1
x	139.643	38	14
c	115.398	52	0
y	74.828	39	13
b	54.219	44	8

Predictor	*d*=1, pred=0	*d*=1, pred=1
x	29	20
c	37	12
y	9	40
b	1	48

Of course, the regression on *x* is nearly useless, since the relationship between *d* and *x* is clearly nonlinear. The other three all have some desirable properties, but the regression on *b*, the variable created by the new optimal transformation, compares favorably with all of them. In this case, the regression on *b* even appears to outperform the regression on *y*, the “true” model variable. This could indicate a possible instance of over-fitting, which is the main “danger” of the method.

Over-fitting

The optimization procedure custom tailors the transformation to the sample it is based on. As noted above, if the objective function constant multiplier, *C*, is set equal to 0, the optimization will attempt to make each sample point its own sub-segment. The easiest way to fight this is to do two things. First, set the value of *C* to a reasonable level, such as the default, which was chosen to be “equidistant” from the single-point-groups and entire-range-group solutions. Second, make sure that each sub-segment has sufficient support by choosing a lower bound on the number of points in each sub-segment. It would be hard to feel comfortable with intervals supported by fewer than ten points. Unfortunately, even ten points is rather small, but it’s difficult to guarantee 25 or 30 points, because the overall sample needs to be kept from growing too big for the optimizer to deal with. So, the next level of protection would be to generate several samples, run the optimization procedure on each of them, and determine a solution based on the combination of all the sample runs; one way to do this would be to compute the objective functions for each solution on each of the samples, and choose the solution which has the lowest sum of objective values for all the samples.

Extension to Multi-valued Discrete Target Variable

This transformation can be extended beyond binary classification to handle multiple outcomes in several ways. Suppose there are *S* target states. One way to extend the transformation would be to break the target into *S-1* separate binary variables, e.g., if there were three states, *A, B, C,* then there would be two target variables, such as *A* or not *A*, and *B* or not *B*; *C* or not *C* is uniquely determined by the first two, so it is not necessary to define it separately. (Naturally, there are two other equivalent formulations for the three-state case.) Then, the optimization procedure for a predictor variable would be run against both the "*A* or not *A*," and the "*B* or not *B*" variables.

Another way to run the procedure to produce one single transformation begins by using a vector representation of the states; for *S* states, use vectors of length *S* to denote the states, as follows: {1,0,…,0}, {0,1,…,0}, …, {0,0,…,1}. Then compute a mean vector for each possible sub-segment, and the associated sum of squared residuals of the state vectors from the mean vectors in some appropriate norm (e.g., Euclidean). Once the residuals have been computed, the optimization proceeds as above to choose the best set of sub-segments. If the target values are ordinal, then the transformation could assign to each sub-segment the weighted average of its mean vector components. (Note that the components of the mean vector will add up to 1.) For example, if the mean vector for a segment is {0.2,0.7,0.1}, and the components correspond to state values of 1, 2, 3, respectively, then the segment's assigned value is (0.2 * 1) + (0.7 * 2) + (0.1 * 3) = 1.9. For non-ordinal targets, the segment's mode value would have to be assigned.

Extension to Continuous-valued Target Variable

This transformation also can be adapted for continuous-valued target variables. In this case, instead of using the sum of squared residuals around the sub-segment mean of the target variable for each sub-segment as the optimization criterion, use the sum of squared residuals around the sub-segment regression line on each sub-segment. Once the optimization picks the sub-segments, the transformation could be discrete or continuous. For a discrete version, assign to each point of a sub-segment the slope of the regression line within that sub-segment. For a continuous version, assign to each point in a sub-segment the value it would take in that sub-segment's regression line.

Interactions Between Predictor Variables

Another advantage of the new transformation is the ease with which it handles variable interactions in a non-parametric manner (i.e., no assumption of functional form). Consider a pair of predictor variables, *p* and *q*, and a sample of *N* points. Now, instead of carving up the range of each single variable into arbitrary subintervals, the goal is to partition the *p-q* plane into arbitrary rectangles. The trick is to notice that *p-q* rectangles can all be obtained from *p* (or *q*) intervals, because every pair of points in the sample determines a unique rectangle in the *p-q* plane; some of the points in the interval may have to be removed from the set because they don't fit into the rectangle. It takes some computational work to find which points to keep for each sub-segment, and some of the intervals may get too small to be used. This actually makes the optimization part easier (fewer variables), although the trade-off is that you can employ larger samples. Note that this can be extended to interactions between more than two predictors.

Further Notes

Optimization using mathematical programming is computationally expensive, and, as previously noted, limits the sample sizes that can be used to create the transformation. There are ways to implement a slightly smoothed, semi-continuous version of the transform, rather than a fully discrete transform, which do not require the expense of mathematical programming. For example, instead of choosing sub-segments, make point-by-point assignments by giving each point the mean value of the sub-segment containing that point that has the lowest residual sum of squares among all sufficiently large sub-segments containing that point. These numbers should vary slowly, with occasional large breaks, mimicking the choice of subintervals.

Finally, note that other metrics besides squares of standard residuals (e.g., absolute values of residuals, residuals with respect to medians rather than means, etc.) can be used without affecting computational difficulty.

References:

[1] *Data Mining Cookbook*, Olivia Parr Rud, 2001, Wiley Computer Publishing, New York, ISBN 0-471-38564-6

[2] http://www.twocrows.com/glossary.htm

[3] *Data Mining: Concepts and Techniques*, Han, Jiawei and Kamber, Micheline, 2001 http://www.ir.iit.edu/~dagr/DataMiningCourse/Spring2001/BookNotes/3prep.pdf

[4] *Solving Data Mining Problems Through Pattern Recognition*, Ruby L. Kennedy, Yuchun Lee, Benjamin Van Roy, Christopher D. Reed, Dr. Richard P. Lippman, 1998, Prentice-Hall, New Jersey, ISBN 0-13-095083-1

[5] *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2001, Springer, New York, ISBN 0-387-95284-5

[6] *Data Mining Using SAS Applications*, George Fernandez, 2003, Chapman & Hall, Boca Raton, ISBN 1-58488-345-6

[7] http://support.sas.com

[8] http://www.powerhouse-inc.com/faq.htm

[9] *Data Preparation for Data Mining*, Dorian Pyle, 1999, Morgan Kaufmann, San Francisco, ISBN 1-55860-529-0

A Supervised Feature Subset Selection Technique for Multivariate Time Series

Kiyoung Yang[*] Hyunjin Yoon[†] Cyrus Shahabi[‡]

Abstract

Feature subset selection (FSS) is a known technique to pre-process the data before performing any data mining tasks, e.g., classification and clustering. FSS provides both cost-effective predictors and a better understanding of the underlying process that generated data. We propose *Corona*, a simple yet effective supervised feature subset selection technique for Multivariate Time Series (MTS). Traditional FSS techniques, such as Recursive Feature Elimination (RFE) and Fisher Criterion (FC), have been applied to MTS datasets, e.g., Brain Computer Interface (BCI) datasets. However, these techniques may lose the correlation information among MTS variables, since each variable is considered separately when an MTS item is *vectorized* before applying RFE and FC. *Corona* maintains the correlation information by utilizing the correlation coefficient matrix of each MTS item as features to be employed for SVM. Our exhaustive sets of experiments show that *Corona* consistently outperforms RFE and FC by up to 100% in terms of classification accuracy, and takes more than one order of magnitude less time than RFE and FC in terms of the overall processing time.

Keywords

multivariate time series, feature subset selection, support vector machine, recursive feature elimination, correlation coefficient matrix

1 Introduction

Feature subset selection (FSS) is one of the techniques to pre-precess the data before we perform any data mining tasks, e.g., classification and clustering. FSS is to identify a subset of original features from a given dataset while removing irrelevant and/or redundant features [1]. The objectives of FSS are [2]:

- to improve the prediction performance of the predictors
- to provide faster and more cost-effective predictors
- to provide a better understanding of the underlying process that generated the data

The FSS methods choose a subset of the original features to be used for the subsequent processes. Hence, only the data generated from those features need to be collected. The differences between feature *extraction* and FSS are:

- Feature subset selection maintains information on the original features while this information is usually lost when feature extraction is used.
- After identifying the subset of original features, only those features can be measured and collected ignoring all the other features. However, feature extraction in general requires measuring all the original features.

A time series is a series of observations, $x_i(t); [i = 1, \cdots, n; t = 1, \cdots, m]$, made sequentially through time where i indexes the measurements made at each time point t [3]. It is called a univariate time series when n is equal to 1, and a multivariate time series (MTS) when n is equal to, or greater than 2.

MTS datasets are common in various fields, such as in multimedia and medicine. For example, in multimedia, Cybergloves used in the Human and Computer Interface applications have around 20 sensors, each of which generates 50$\sim$100 values in a second [4, 5]. In [6], 22 markers are spread over the human body to measure the movements of human parts while walking. The dataset collected is then used to recognize and identify the person at a distance by how he or she walks. In the Neuro-rehabilitation domain, kinematics datasets generated from sensors are collected and analyzed to evaluate the functional behavior (i.e., the movement of upper extremity) of post-stroke patients [7]. In medicine, Electro Encephalogram (EEG) from 64 electrodes placed on

[*]Computer Science Department, University of Southern California, Los Angeles, CA 90089, U.S.A., kiyoungy@usc.edu

[†]Computer Science Department, University of Southern California, Los Angeles, CA 90089, U.S.A., hjy@usc.edu

[‡]Computer Science Department, University of Southern California, Los Angeles, CA 90089, U.S.A., shahabi@usc.edu

the scalp are measured to examine the correlation of genetic predisposition to alcoholism [8]. Functional Magnetic Resonance Imaging (fMRI) from 696 voxels out of 4391 has been used to detect similarities in activation between voxels in [9].

The size of an MTS dataset can become very large quickly. For example, the EEG dataset in [10] utilizes tens of electrodes and the sampling rate is 256Hz. In order to process MTS datasets efficiently, it is therefore inevitable to preprocess the datasets to obtain the relevant subset of features which will be subsequently employed for further processing. In the field of Brain Computer Interfaces (BCIs), the selection of relevant features is considered absolutely necessary for the EEG dataset, since the *neural correlates* are not known in such detail [10]. Identifying optimal and valid features that differentiate the post-stroke patients from the healthy subjects is also challenging in the Neurorehabilitation applications.

An MTS item is naturally represented as an $m \times n$ matrix, where m is the number of observations and n is the number of *variables*, e.g., sensors. However, the state of the art feature subset selection techniques, such as Recursive Feature elimination (RFE) [2], require each item to be represented in one row. Consequently, to utilize these techniques on MTS datasets, each MTS item needs to be first transformed into one row or column vector, which we call *vectorization*. For example, in [10] where an EEG dataset with 39 channels is used, an autoregressive (AR) model of order 3 is utilized to represent each channel. Hence, each 39 channel EEG time series is transformed into a 117 dimensional vector. However, if each channel of EEG is considered separately, we will lose the correlation information among the variables.

Information theory (IT) based feature subset selection methods, such as information gain and information gain ratio, have been extensively studied and employed in the data mining and machine learning community [11, 12]. However, IT based feature subset selection methods are also not directly applicable to MTS items, because, again, an MTS item is not a vector, and also each value of an MTS item is continuous, not discrete. Hence, each MTS item should first be transformed into a vector and also be discretized, which usually results in loss of important information.

In this paper, we propose a simple yet quite effective subset selection method for multivariate time series (MTS)[1], termed *Corona* (*Cor*relati*on* as Fe*a*tures). *Corona* is based on RFE. Recall that RFE, which utilizes SVM, requires each item to be represented as a vector. The performance of RFE will therefore heavily rely on how the MTS dataset is fed into SVM, i.e., how each MTS item is transformed to be utilized by SVM. *Corona* employs the correlation coefficients of an MTS item as features for SVM and hence for RFE. The intuition is based on our previous work [13] which has shown that the correlation information among the variables plays an important role in obtaining the similarity between two MTS items. Hence, *Corona* first computes the pairwise correlation coefficients of all the variables, i.e., the correlation coefficient matrix, of each MTS item. Since the correlation coefficient matrix is symmetric and its diagonal values are all 1s, only the upper triangle of the correlation coefficient matrix except the diagonal values is utilized to *vectorize* an MTS item. Consequently, an MTS dataset is transformed into a matrix, which we call a *feature matrix*, where each row represents an MTS item. *Corona* subsequently trains SVM on the feature matrix, which will produce the weights of each feature. Note that each feature in the feature matrix is the correlation coefficient of two variables. *Corona* then aggregates the weights for each variable and ranks the variables based on the aggregated weights. Subsequently, *Corona* eliminates the variable with the lowest rank. This process is repeated until the required number of variables is obtained. Our experiments show that the classification performance of the variable subsets selected by *Corona* is up to about 100% better than those selected by other feature subset selection methods, such as Recursive Feature Elimination (RFE) and Fisher Criterion (FC). Moreover, *Corona* takes more than one order of magnitude less time than RFE and FC in terms of the overall processing time which includes the time to *vectorize* an MTS dataset.

The remainder of this paper is organized as follows. Section 2 discusses the background. Our proposed method is described in Section 3, which is followed by the experiments and results in Section 4. Related work is presented in Section 5 followed by conclusions and future work in Section 6.

2 Background

Corona utilizes the correlation coefficient matrix and RFE for feature subset selection of MTS datasets. In this section, we briefly describe the correlation coefficient matrix, Support Vector Machine and Recursive Feature Elimination.

2.1 Correlation Coefficient Matrix The correlation represents how strongly one variable implies the other, based on the available data [14]. Assume that

[1]For multivariate time series, each *variable* is regarded as a feature [10]. Hence, the terms *feature* and *variable* are interchangeably used throughout this paper, when there is no ambiguity.

a and **b** are two vectors of length n. The correlation between **a** and **b** is then defined as follows [14]:

$$Corr(\mathbf{a},\mathbf{b}) = \frac{\sum_{i=1}^{n}(a_i-\bar{a})(b_i-\bar{b})}{(n-1)\sigma_a\sigma_b} \tag{2.1}$$

where $\bar{a}$ and $\bar{b}$ are the averages of vector **a** and **b**, respectively; σ_a and σ_b are the standard deviations of **a** and **b**, respectively. The correlation value ranges from -1 to 1. A value greater than 0 means that there is a positive correlation. That is, if the values of **a** increase, then the values of **b** would also increase. If the correlation is 0, then there is no correlation between **a** and **b** meaning that they are independent. The negative correlation value means that there is a negative correlation between **a** and **b**. That is, if the values of **a** increase, then the values of **b** would decrease, or vice versa.

A correlation coefficient matrix is a symmetric matrix, where the $(i,j)th$ entry in the matrix represents the correlation between the ith and jth variables. Our proposed supervised feature subset selection technique, *Corona*, utilizes the correlation coefficient matrix of each MTS item as features for SVM to obtain the *weights* of each variable, which is described in Section 3.

2.2 Support Vector Machine Support Vector Machine (SVM) is a classification technique by Vapnik [15]. SVM performs classification by obtaining and utilizing the *optimal separating hyperplane* that separates two classes and maximizes the distance to the closest point from either class, called *margin* [15, 16]. Figure 1 represents the training result of an SVM model for a simple two class dataset[2].

The hyperplane that separates the two classes shown in Figure 1 can be described as follows [18]:

$$g(\mathbf{x}) = \mathbf{w}'\mathbf{x} + w_0 \tag{2.2}$$

where **w** is the norm vector of the hyperplane $g(\mathbf{x})$ and $w_0/||\mathbf{w}||$ is the distance from the origin to the hyperplane. Given new data $\mathbf{x}_i$, the sign of $g(\mathbf{x}_i)$ determines the class of $\mathbf{x}_i$. For simplicity, we described only the case where the classes are *linearly separable*. For more details, please refer to [18, 16].

2.3 Recursive Feature Elimination Based on SVM, Guyon *et al* [19] proposed a feature subset selection method called Recursive Feature Elimination (RFE). RFE is a *stepwise backward feature elimination* method [14]. That is, RFE starts with all the features and removes features based on a ranking criterion until the required number of features are obtained. The procedure can be briefly described as in Algorithm 1 [19]:

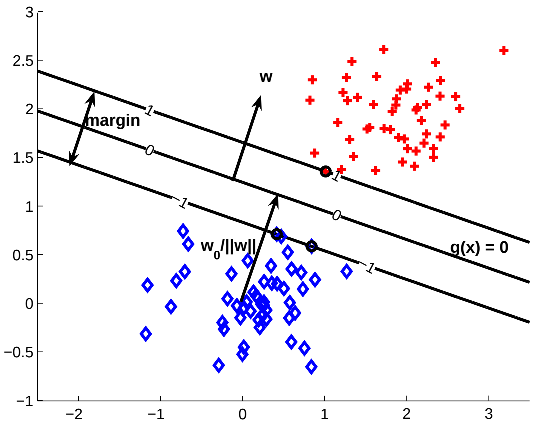


Figure 1: Two classes are linearly separable.

Algorithm 1 Recursive Feature Elimination

```
1: Train SVM;
2: Rank the features;
3: Eliminate the feature with the lowest rank;
4: Repeat until the required number of features are
   retained;
```

In order to rank the features, RFE utilizes the *sensitivity analysis* based on the weight vector **w** in Equation 2.2. That is, at each iteration, RFE eliminates one feature with the minimum weight. The intuition is that the feature with the minimum weight would least influence the weight vector norm [20], and is therefore to be removed.

RFE, however, cannot be used with MTS datasets *as is*, since an MTS item is represented as a matrix, while RFE requires each item to be represented as a vector. In [10], for example, each variable, i.e., channel, is transformed separately using the autoregressive fit coefficients of order 3. By doing so, however, the correlation information among the variables would be lost. In the following section, we propose an extension of RFE to MTS datasets, called *Corona*.

3 Proposed Method

In this section, we describe *Corona*, which is a simple yet effective feature subset selection technique for MTS datasets based on RFE. Recall that SVM, hence RFE, requires each MTS item to be represented as a vector.

[2]SVM and Kernel Methods Matlab Toolbox [17] is utilized to generate the figure.

Corona utilizes the correlation coefficients as features for an MTS item to be used for SVM. The intuition using the correlation coefficients as features for MTS items to be used for SVM comes from our previous work [13] which has shown that the correlation information of an MTS item plays a significant role in computing the similarity between two MTS items.

Hence, *Corona* first computes the correlation coefficient matrix for each MTS item. A correlation coefficient matrix is symmetric and its diagonal values, which represent the *autocorrelations* of all the variables, are all 1s. Hence, as features for an MTS item, the correlation coefficients in the upper triangle of the correlation coefficient matrix except the diagonal values are utilized, which are then vectorized as in Algorithm 2. For an n-variate MTS item, the number of features to be used for SVM is $\sum_{i=1}^{n-1} i = n \times (n-1)/2$. For example, for the HumanGait dataset where n is 66, the number of features is $66 \times 65/2 = 2145$. For an MTS dataset which has N items, this transformation results in an $N \times p$ matrix, where $p = n \times (n-1)/2$. We denote this matrix a *feature matrix*.

Corona subsequently trains SVM using the feature matrix. Utilizing the model resulted from the SVM training, we obtain the weight vector $\mathbf{w}$ for the features that have been employed in the SVM training. Note that each feature utilized for SVM training is a correlation of two variables. In order to determine the ranks of the *variables*, we construct a symmetric matrix using the weights obtained by SVM, which we call a *weight matrix* (Lines 1–10 in Algorithm 4). This is similar to *un-vectorizing* the vectorized correlation coefficient matrix except that the *weights* obtained from SVM are used, not the correlation coefficients. Hence, the ith column in the weight matrix represents all the weights of the *features*, i.e., the correlation coefficients, with which the ith *variable* is associated. After obtaining the weight matrix, *Corona* aggregates all the weights of each variable and obtains one value per variable. Finally, based on the aggregated values, *Corona* decides which variable to eliminate. In our experiments, we took the *greedy* approach, and identified a variable whose maximum weight is the minimum among the maximum weights of all the variables (Lines 11–12 in Algorithm 4). The variable whose maximum weight is the minimum is then to be removed. The intuition behind using the *max* aggregate function is to retain the variables that are associated with the correlation coefficients which contribute most to the SVM training result.

Algorithm 3 describes the overall process of *Corona*. Given an MTS dataset, *Corona* first computes the feature matrix T by vectorizing the upper triangle of the correlation coefficient matrix of each MTS item (Lines 1–4 of Algorithm 3, and Algorithm 2). Subsequently, it performs SVM on the feature matrix. Using the feature weights obtained from SVM, *Corona* ranks the variables as in Algorithm 4. The entire process is repeated until the required number of variables are identified.

Algorithm 2 Vectorize a correlation coefficient matrix using the upper triangle

```
Require: $C$ {a correlation coefficient matrix of an $n$-variate MTS item};
1: $C_{vectorized} \leftarrow []$;
2: for $i = 1$ to $n$ do
3:     $C_{vectorized} \leftarrow [C_{vectorized} \quad C[i, (i+1):n]]$;
4: end for
```

Algorithm 3 Corona

```
Require: MTS dataset, $N$ {the number of items in the dataset}, $k$ {the required number of variables};
1: for $i = 1$ to $N$ do
2:     $C \leftarrow$ correlation coefficient matrix of the $i$TtH MTS item;
3:     $T[i,:] \leftarrow$ vectorize $C$ using the upper triangle of $C$;
4: end for
5: $[rank_{SVM}, weights_{SVM}] \leftarrow$ Train SVM on $T$;
6: Rank variables using $weights_{SVM}$;
7: Remove one variable with the lowest rank;
8: Repeat until $k$ variables remain;
```

4 Performance Evaluation

In order to evaluate the effectiveness of Corona in terms of classification performance and overall processing time, we conducted several experiments on three real-world datasets. After obtaining a subset of variables using Corona, we performed classification using SVM with linear kernel as in [10]. Subsequently, we compared the performance of Corona with those of RFE [2, 10], Fisher Criterion (FC), Exhaustive Search Selection (ESS) when possible, and using all the available variables (ALL). The algorithm of Corona for the experiments is implemented in $Matlab$ and in[3] R using[4] $e1071$ and[5] RFE packages.

4.1 Datasets The **HumanGait dataset** [6] has been used for identifying a person by recognizing his/her gait at a distance. In order to capture the gait data, a

[3]http://www.r-project.org/

[4]http://cran.r-project.org/src/contrib/Descriptions/e1071.html

[5]http://www.hds.utc.fr/~ambroise/softwares/RFE/

Algorithm 4 Rank variables using $weights_{SVM}$

```
Require: $weights_{SVM}$ {weights obtained by SVM}, $n$ {the number of variables for an MTS item};
 1: $W \leftarrow []$;
 2: $count \leftarrow 1$;
 3: for $i = 1$ to $n$ do
 4:     $W[i, (i+1):n] \leftarrow weights_{SVM}[count : (count + n - i - 1)]$;
 5:     $count \leftarrow count + n - i$;
 6: end for
 7: $W \leftarrow W + transpose(W)$;
 8: for $i = 1$ to $n$ do
 9:     $W(i, i) \leftarrow 1$;
10: end for
11: $weights_{Corona} \leftarrow$ Aggregate $W$ in column-wise;
12: $rank_{Corona} \leftarrow sort(weights_{Corona})$;
```

twelve-camera VICON system was utilized with 22 reflective markers attached to each subject. For each reflective marker, 3D position, i.e., x, y and z, are acquired at 120Hz, generating 66 values at each timestamp. 15 subjects, which are the labels assigned to the dataset, participated in the experiments and were required to walk at four different speeds, nine times for each speed. The total number of data items is 540 ($15 \times 4 \times 9$) and the average length is 133.

Motor Behavior and Rehabilitation Laboratory, University of Southern California collected **Brain and Behavior Correlates of Arm Rehabilitation (BCAR) kinematics dataset** to study the effect of Constraint-Induced (CI) physical therapy on the post-stroke patients' control of upper extremity [7]. The functional specific task performed by subjects was a continuous 3 phase reach-grasp-place action; a subject sits on a chair pressing down the starting switch with his or her impaired forearm. She or he is then supposed to reach for a target object, either a cylinder or a card, grasp it, place it into a designated hole, release it, and finally bring her or his hand back to the starting switch. This specific task is repeated five times per subject under four different conditions, i.e., for 2 different objects (Cylinder/Card) by posing 2 different forearm postures (pronation/supination). The performance is traced by six *miniBIRD* trackers attached on the index nail, thumb nail, dorsal hand, distal dorsal forearm, lateral mid upper arm and shoulder, respectively. Then, 11 dependent variables are measured from the raw data, sampled at 120Hz and filtered using a 0-lag Butterworth low-pass filter with a 20Hz cut-off frequency. Unlike other datasets, BCAR dataset kept the record of 11 dependent features rather than 36 *raw* variables at each timestamp. They were defined by experts in advance and calculated from the raw variables by the device software provided with the trackers; some of them were just raw variables (e.g., wrist tracker X, Y, and Z coordinates) while others were synthesized from raw variables (e.g., aperture was computed as tangential displacement of two trackers on thumb and index nail). Note that these 11 variables were considered as original variables throughout the experiments. Four control (i.e., healthy) subjects and three post-stroke subjects experiencing a different level of impairment participated in the experiments. For each of the 4 conditions, the total number of data items is 39, and their average length is about 454 (i.e., about 3.78 seconds).

	HumanGait	BCAR	BCI MPI
# of variables	66	11	39
average length	133	454	1280
# of labels	15	2	2
# of items per label	36	22/17	1000
total # of items	540	39	2000

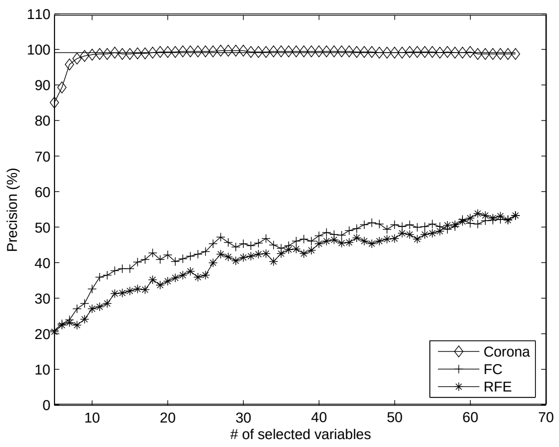
Table 1: Summary of datasets used in the experiments

The **Brain Computer interface (BCI) dataset** at the **Max Planck Institute (MPI)** [10] was collected to examine the relationship between the brain activity and the motor imagery, i.e., the imagination of limb movements. Eight right handed male subjects participated in the experiments, out of which three subjects were filtered out after pre-analysis [10]. 39 electrodes were placed on the scalp to record the EEG signals at the rate of 256Hz. The total number of items is 2000, i.e., 400 items per subject.

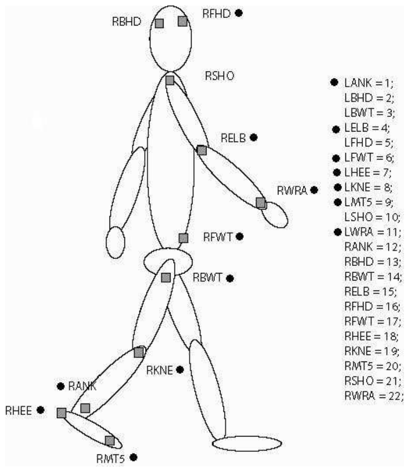
Table 1 summarizes the datasets used in the experiments.

4.2 Classification Performance We evaluated the effectiveness of Corona in terms of classification accuracy. Support Vector Machine (SVM) with linear kernel was adopted as the classifier. Using SVM, we performed leave-one-out cross validation for the BCAR dataset and 10 fold cross validation [14] for the rest since they have too large number of items to conduct leave-one-out cross validation.

For RFE and FC, we vectorized each MTS item as in [10]. That is, each variable is represented as the autoregressive (AR) fit coefficients of order 3 using the forward backward linear prediction [21]. Therefore, each MTS item with n variables is represented in a vector of size $n \times 3$. *The Spider* [22] implementation of FC is subsequently employed. For small datasets, i.e., BCAR and HumanGait, RFE in *The Spider* [22] was employed, while for large dataset, i.e., BCI MPI, RFE package for R is utilized. Note that Exhaustive Search Selection



(a)



(b)

Figure 2: (a) HumanGait dataset, Classification Evaluation (b) 22 markers for the HumanGait dataset. The markers with a filled circle represent 16 markers from which the 27 variables are selected by *Corona*, which yields better performance accuracy than using all the 66 variables.

(ESS) method was performed only on BCAR dataset due to the intractability of ESS for the large datasets. The ESS methods simply searches exhaustively among all possible combinations of variables and selects the best combination. Obviously, this is an impractical approach due to its high complexity and we only used it here (when possible) to generate the ground truth.

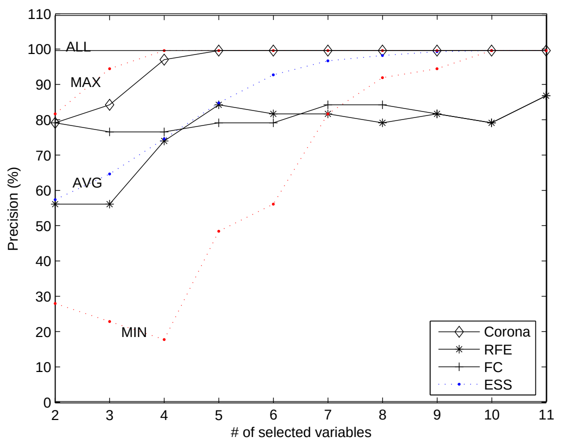
Figure 2(a) presents the generalization performances on the HumanGait dataset. It shows that a subset of 11 variables selected by *Corona* out of 66 performs the same as the one using all the variables (99.0741% accuracy), which is represented as a solid horizontal line. Moreover, a subset of 27 variables yields 100% accuracy. The 27 variables selected by Corona are from only 16 markers (marked with a filled circle in Figure 2(b)) out of 22, which would mean that the values generated by the remaining 6 markers does not contribute much to the identification of the person. From this information we may be able to better understand the characteristics of the human walking.

The performances by RFE and FC for the HumanGait dataset is much worse than Corona. Even when using all the variables, the classification accuracy is around 55%. Considering the fact that RFE on 3 AR coefficients performed well in [10], this may indicate that for the HumanGait dataset the correlation information among variables is more important than for the BCI MPI dataset. Hence, each variable should not be taken out separately to compute the autoregressive coefficients, by which the correlation information would be lost. Note that in [10], the order 3 for the autoregressive fit is identified after proper model selection experiments, which would mean that for the HumanGait dataset, the order of the autoregressive fit should be determined, again, after comparing different order models. Hence, it is not a trivial task to transform an MTS item into a vector, after which the traditional machine learning techniques, such as Support Vector Machine (SVM), can be applied.

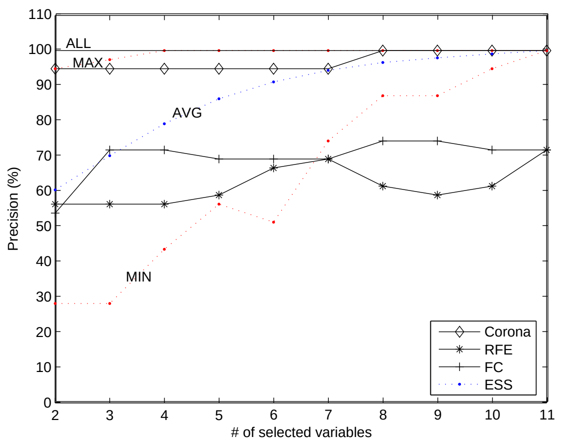
Figure 3 shows the classification performance of the selected variables on the BCAR dataset for 4 different conditions. For example, Figure 3(c) represents that a card was used as a target object and the pronated forearm posture was taken by a subject to perform the continuous reach-grasp-place task in [7].

The BCAR is the simplest dataset with 11 original variables and the number of MTS items for each condition is just 39. Hence, we applied the Exhaustive Search Selection (ESS) method to find all the possible variable subset combinations, for each of which we performed leave-one-out cross validation. It took about 87 minutes to complete the whole ESS experiments. The result of ESS shows that 100% classification accuracy can be achieved by either 4 or 5 variables out of 11. The dotted lines represent the best, the average, and the worst performance obtained by ESS, respectively, given the number of selected variables.

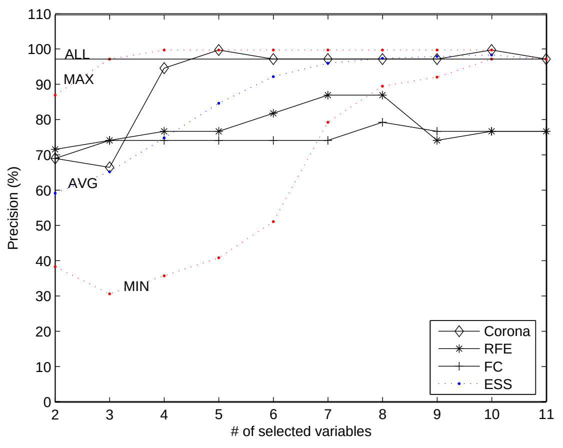
Figure 3 again shows that Corona consistently outperforms RFE and FC methods. The figure also depicts that the 5 variables selected by *Corona* produce



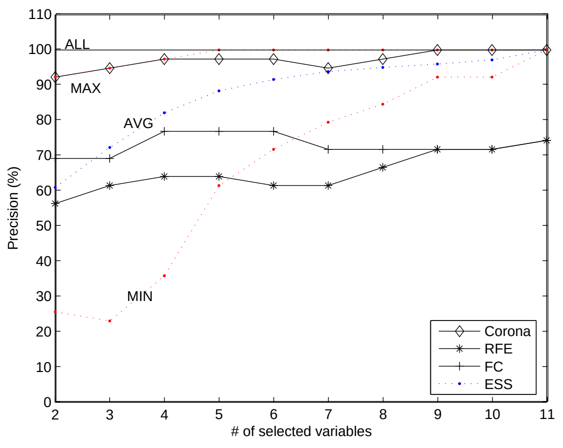
(a) Cylinder/Pronation



(b) Cylinder/Supination



(c) Card/Pronation



(d) Card/Supination

Figure 3: BCAR dataset, Classification Evaluation

100% classification accuracy for Cylinder/Pronation and Card/Pronation conditions. Besides, *Corona* outperforms or performs the same as the one using all the variables, which is represented as a horizontal solid line. This implies that *Corona* never eliminates useful information in its variable selection. For the Cylinder/Pronation condition, for example, Figure 3(a) shows that only the 4 variables selected by *Corona* produce about 98% classification accuracy, which is the same as using all the 11 variables. Moreover, the overall performance of *Corona* is close to the best performance of ESS, which is far from the average performance.

As illustrated in the figure, FC method never beats the *Corona* for 3 conditions, and for the Card/Pronation condition, *Corona* by far outperforms FC when more than 3 variables are selected. As compared to RFE, *Corona* again shows consistently better classification performance almost always.

Figure 4 represents the performance comparison using the BCI MPI dataset. Note that unlike in [10] where they applied the feature subset selection per subject, the whole items from the 5 subjects were utilized in our experiments, which would make the subset of variables selected by *Corona* more applicable for subsequent data mining tasks. Moreover, the regularization parameter $\mathcal{C}_s$ for SVM was estimated via 10 fold cross validation from the training datasets in [10], while we used the default value, which is 1. The figure again depicts that Corona performs far better than RFE and FC.

For the BCI MPI dataset, it is intractable to try all the combinations of the 39 channels to identify the best combination. Therefore, to find the ground-truth,

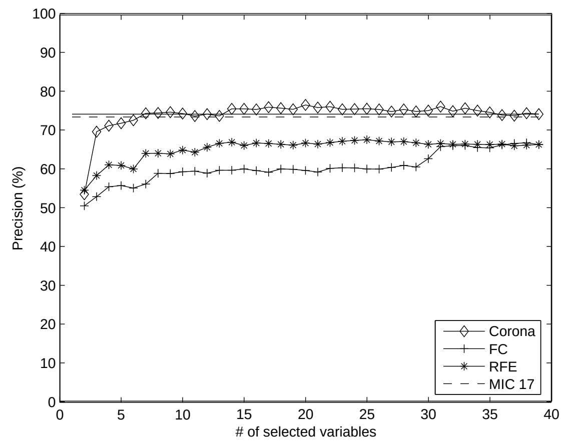


Figure 4: BCI MPI dataset, Classification Evaluation

in [10], the 17 channels located over or close to the motor cortex were manually identified as the best combination of channels using the domain knowledge. In Figure 4, the classification performance using those 17 motor imagery channels (termed MIC 17) is presented in dashed lines, while the performance using all the variables is shown in a solid horizontal line. Using the 17 variables selected by Corona, the classification accuracy is 75.45%, which is even better than the expert-selected channels of MIC 17 whose accuracy is 73.65%.

4.3 Processing Time *Corona* in fact utilizes a lot more number of features than RFE to *vectorize* an MTS item. For example, for the HumanGait dataset where there are 66 variables, each MTS item is represented with $66 \times 65/2 = 2145$ features by *Corona*, while RFE represents each MTS item with $66 \times 3 = 198$ features. Obviously, this would result in more training time for SVM on which both *Corona* and RFE are based. However, RFE takes a considerable amount of time to compute and obtain the AR coefficients of order 3. Hence, the overall processing time of *Corona*, including the time to transform the MTS dataset, is one order of magnitude less than that of RFE.

For the BCI MPI dataset, for example, it takes only 4.562 seconds to compute all the 2000 correlation coefficient matrices for *Corona*, while it takes about 7600 seconds to compute the AR coefficients of order 3 for RFE, both using Matlab. The total processing time including the transformation for *Corona* of the BCI MPI dataset is less than 480 seconds, while that of RFE is more than 7800 seconds. Table 2 summarizes the processing time of the 3 feature selection methods employed for the experiments.

Table 2: Comparison of processing time in seconds for different feature selection methods on 3 different datasets

	HumanGait	BCAR	BCI MPI
Corona	422.688	0.191	472.953
RFE	962.063	9.039	7886.844
FC	113.907	6.469	7594.941

5 Related Work

In the field of Brain Computer Interfaces (BCIs), extensive research has been conducted on Electroencephalogram (EEG) datasets. The EEG dataset is collected using multiple electrodes placed on the scalp. The sampling rate is hundreds of Hertz. The selection of relevant features is considered absolutely necessary for the EEG dataset, since the neural correlates are not known in such detail [10].

In [10], feature selection is performed on the 39 channel EEG dataset. Each EEG item is broken into 39 separate channels, and for each channel, autoregressive (AR) fit of order 3 is computed. Subsequently, each channel is represented by 3 autoregressive coefficients. Feature selection using Recursive Feature Elimination (RFE) is then performed on these transformed dataset. As shown in Section 4.2, by considering the channels separately, they lose the correlation information among channels.

In [23], EEG dataset from UCI KDD Archive [24] has been used for the experiments. EEG-1 dataset contains only 20 measurements for two subjects from two arbitrary electrodes (F4 and P8). EEG-2 dataset contains 20 measurements from the same 2 electrodes for each subject. It is not clear how the two subjects out of 122 subjects and the two electrodes out of 64 are chosen. The best accuracies obtained are $90.0 \pm 0.0\%$ using DCHMM-exact, $90.5 \pm 5.6\%$ using Multivariate HMM for the EEG-1 dataset. $78.5 \pm 8.0\%$ using Multivariate HMM.

In [25], a subset of the HumanGait dataset, a total of 45 items of 15 subjects, was used for an HMM-based clustering. They, however, achieved only 75% classification accuracy, which could have been achieved by *Corona* using only 9 variables out of 66 as shown in Figure 2(a).

In [26], Genetic Algorithm (GA) and Support Vector Machine (SVM) are used for feature subset selection. Two EEG datasets are used, TTD and NIPS 2001. The TTD (Thought Translation Device) EEG dataset were generated with 6 channels, and the other EEG dataset which was submitted to Neural Information Processing Systems (NIPS) Conference in 2001, were collected with

27 channels. For the EEG dataset with 6 channels, they also performed the exhaustive search to find out the best channels. The advantage of GA is that the optimal subset of variables is produced as output, and hence, one does not have to specify how many variables she would like to select. However, GA is known to be very time consuming.

In [27], features are firstly extracted from the original dataset, and then feature subset selection are performed using mutual information. The accuracy from training set is less than 70% and from test set is less than 85%. The EEG data used was obtained from Graz University of Technology, Austria, and Artificial Neural Network (ANN) is used for classification. Note that this approach, i.e., performing feature extraction and then feature selection, may work well in terms of classification accuracy. However, we cannot reduce the amount of data to be collected, if the features are global features for which all the *raw* data would be required.

6 Conclusion and Future Work

In this paper, we proposed a simple yet quite effective feature subset selection method for multivariate time series (MTS), termed *Corona*. *Corona* first vectorizes the correlation coefficient matrix of each MTS item to be used as features for SVM, and yields a *feature matrix*. After training SVM on the feature matrix, *Corona* computes the *weight matrix*, from which the ranks for the variables are identified. Based on the ranks, *Corona* eliminates one variable with the lowest rank, and repeats itself until the required number of variables are retained. Our experiments on the three real-world datasets show that *Corona* consistently outperforms other feature selection methods, such as Recursive Feature Elimination (RFE) and Fisher Criterion (FC) in terms of classification performance by up to 100%. Moreover, *Corona* takes more than one order of magnitude less time than RFE in terms of the overall processing time.

We intend to extend this technique to the stream of data where the feature subset selection can be performed incrementally adjusting itself based on the observations collected thus far.

Acknowledgements

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0238560 (PECASE) and IIS-0307908, and unrestricted cash gifts from Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Dr. Carolee Winstein and Jarugool Tretiluxana for providing us the BCAR dataset and valuable feedbacks, and Thomas Navin Lal for providing us the BCI MPI dataset. The authors would also like to thank the anonymous reviewers for their valuable comments.

References

[1] Liu, H., Yu, L., Dash, M., Motoda, H.: Active feature selection using classes. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. (2003)

[2] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182

[3] Tucker, A., Swift, S., Liu, X.: Variable grouping in multivariate time series via correlation. IEEE Trans. on Systems, Man, and Cybernetics, Part B **31** (2001)

[4] Kadous, M.W.: Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. PhD thesis, University of New South Wales (2002)

[5] Shahabi, C.: AIMS: An immersidata management system. In: VLDB Biennial Conference on Innovative Data Systems Research. (2003)

[6] Tanawongsuwan, Bobick: Performance analysis of time-distance gait parameters under different speeds. In: 4th International Conference on Audio- and Video Based Biometric Person Authentication, Guildford, UK (2003)

[7] Winstein, C., Tretriluxana, J.: Motor skill learning after rehabilitative therapy: Kinematics of a reach-grasp task. In: the Society For Neuroscience, San Diego, USA (2004)

[8] Zhang, X.L., Begleiter, H., Porjesz, B., Wang, W., Litke, A.: Event related potentials during object recognition tasks. Brain Research Bulletin **38** (1995)

[9] Goutte, C., Toft, P., Rostrup, E., Nielsen, F.A., Hansen, L.K.: On clustering fmri time series. NeuroImage **9** (1999)

[10] Lal, T.N., Schröder, M., Hinterberger, T., Weston, J., Bogdan, M., Birbaumer, N., Schölkopf, B.: Support vector channel selection in BCI. IEEE Trans. on Biomedical Engineering **51** (2004)

[11] Mitchel, T.M.: Machine Learning. McGraw Hill (1997)

[12] Witten, I.H., Frank, E.: 7. In: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)

[13] Yang, K., Shahabi, C.: A PCA-based similarity measure for multivariate time series. In: The Second ACM MMDB. (2004)

[14] Han, J., Kamber, M.: 3. In: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000) 121

[15] Vapnik, V.N.: Statistical Learning Theory. Wiley (1998)

[16] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2001)

[17] Canu, S., Grandvalet, Y., Rakotomamonjy, A.: Svm and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France (2003)

[18] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Second edn. Wiley Interscience (2001)

[19] Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46** (2002) 389–422

[20] Rakotomamonjy, A.: Variable selection using svm-based criteria. Journal of Machine Learning Research **3** (2003) 1357 – 1370

[21] Moon, T.K., Stirling, W.C.: Mathematical Methods and Algorithms for Signal Processing. Prentice Hall (2000)

[22] Weston, J., Elisseeff, A., BakIr, G., Sinz, F.: Spider: object-orientated machine learning library. http://www.kyb.tuebingen.mpg.de/bs/people/spider/ (2004)

[23] Zhong, S., Ghosh, J.: HMMs and coupled HMMs for multi-channel EEG classification. In: International Joint Conference on Neural Networks. (2002)

[24] Hettich, S., Bay, S.D.: The UCI KDD Archive. http://kdd.ics.uci.edu (1999)

[25] Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering clusters in motion time-series data. In: IEEE CVPR. (2003)

[26] Schröder, M., Bogdan, M., Hinterberger, T., Birbaumer, N.: Automated EEG feature selection for brain computer interfaces. In: IEEE EMBS International Conference on Neural Engineering. (2003)

[27] Deriche, M., Al-Ani, A.: A new algorithm for EEG feature selection using mutual information. In: IEEE International Conference on Acoustics, Speech, and Signal Processing. (2001)

A Hybrid Cluster Tree for Variable Selection

Zhiqian Fu
Shandong University
27 Shanda Nanlu
Jinan, Shandong, P.R.China 250100
and
Zhiwei Fu, Isa Sarac
Virginia International University
3957 Pender Drive, Fairfax, VA 22030 USA

ABSTRACT

Contemporary business have been pursuing a variety of approaches to analyze large quantities of data. The artificial intelligence and statistics community have provided sophisticated, effective methods thus far, but improvements still need to be made to better understand the variables of data while reducing its size. This paper introduces a simple hybrid tree approach for variable selection that integrates heuristic statistical cluster analysis methods with tree-structured learning system C4.5 to select the best set of variables. Experimental results have shown improved performance of the proposed approach as compared with standard data reduction methods.

Keywords: Variable selection, hybrid learning, cluster analysis, decision tree

1. INTRODUCTION

Contemporary business has gathered more and more amount of data. It has been challenging to analyze these data due to their sheer size and complexity. An effective variable selection approach needs to be developed such that the reduced subset could retain sufficient information of the original data set. In this paper, we proposed a hybrid cluster tree approach (HCT) based on the statistical clustering methods, C4.5[6], and fine-tuning heuristics. We conduct experiemnts, and study performance of the developed decision trees on unseen data, conduct statistical comparisons using *t*-test with other data reduction approaches, and make some recommendations.

2. DATA REDUCTION TECHNIQUE

The literature on variable selection exists in statistics and artificial intelligence, and numerous algorithms and approaches have been developed to reduce the complexity of variables. In statistical analyses, stepwise multiple regression, multidimensional scaling, principal component analysis, factor analysis and cluster analysis are the most common techniques, and forward/backward stepwise multiple regression are widely used trial-and-error procedures [4]. Through effective, these statistical methods have been widely used in reducing the number of variables in the original data sets, but could not preserve the identity of the original variables by transforming the original variables into new attributes [3].

In artificial intelligence, some heuristics have been well developed to deal with the variable complexity in neural network [7] and genetic algorithms [1][2][5][8]. C4.5 is a tree-structured classification learning system that uses decision trees as a model representation to find the best decision tree model that describes the structure of the data set by using heuristics based on the information theory.

3. HYBRID CLUSTER TREE (HCT)

In HCT, an initial full set of variables is provided along with a training set extracted from the original data. The cluster analysis is implemented on the training set for clustering the most similar observations in the original data set, such that the entire data set could be learned integrally and near-equally. First, we cluster the entire training data set of *N* into $n = n_0$ groups by statistical criterion, e.g., Ward's minimum variance. Second, in each group, we average the values of the target observations and the input observations for each variable. Therefore, a pair of average values consists of one new pattern. Then, we build up the cluster model and run C4.5 on the new patterns. Finally, we cluster the whole data into more groups, generating corresponding new patterns, and repeat C4.5 on newly created patterns. The process proceeds until each clustered group contains only some number of original patterns as specified. In HCT, we implement C4.5 iteratively on the induced new training sets, and study the performance of the induced decision trees thereafter, specifically in terms of generalizability as measured by classification accuracy on unseen data. By HCT, we intend to obtain the near-optimal representative of the original variables, i.e., a set of variables with much reduced complexity but still good computational performance.

4. COMPUTATIONAL EXPERIMENTS

We use the data from 1996 Summer Olympic Games in Atlanta, and other socio-economic information in our experiments. The typical variables included in the experiments are Area size, Population, Population growth rate, Death rate, Infant mortality rate, Life expectancy, Railroads, Highway mileage, Electric capacity, Imports, Exports, etc.

We first preprocess the data by randomly spliting the entire data set into a training set and a test set. Since the variables are on significantly different scales, e.g., Population and Death Rate, we then normalize all the variables with the mean of zero and standard deviation of one. Based on the preprocessed data, we conduct four experiments to test the robustness of HCT. In Experiment 1, we run C4.5 on training set and then teobtain its performance accuracy on the test set directly without any heuristics in relation to variable selection. In Experiment 2, we run stepwise regression on training set and then obtain the accuracy on the test set. In Experiment 3, we first run stepwise multiple regression on the data to filter the variables, then followed by running C4.5 on the induced data set to further select the variables. In Experiment 4, we first run cluster algorithm to reduce the record size, and then run C4.5 algorithm on the induced data set to choose the best variable selection. In cluster analysis, we use Ward's minimum variance clustering criterion with only training set as inputs. We then compute the classification accuracy of the final models on unseen data.

Table. Computational results (the *p*-values of variable reduction are 0.0000 at $\alpha = 0.05$ level)

Experimental Design	Classification Accuracy (%)	Variable Reduction (%)	Variable selected
C4.5	70.8	59%	Highways, Railroad, Birth rate, Death Rate, Population growth rate, National product per capita, Area
Stepwise Regression	70	47%	Highways, Railroad, Birth rate, Death Rate, Life expectancy, National product per capita, Area, Imports, infant mortality rate
Stepwise Regression + C4.5	84.6	38%	**Railroad**, **Airports**, **Death rate**, National product per capita, Life expectancy, **Imports**, Area, **infant mortality rate**
Fine-tuned Clustering + C4 .5	87.7	50%	Population growth rate, **Railroad**, **Birth rate**, Area, **Death rate**, Airports.

The computational results of our exeriments are shown in the table. We see that in Experiment 1, out of seventeen variables in the original full set of variables, seven variables are selected and the classification accuracy is 70.8%., We have nine variables selected In Experiment 2 by stepwise regression with the accuracy of 70.0%. In Experiment 3, after conducting stepwise regression, eight are filtered initially, and three more variables are eliminated after C4.5 was conducted. The bolded variables in the last column are the ones filtered out by C4.5. The accuracy in Experiment 3 is 84.6%. We point out that it is reasonable since there are a lot of noisy information in the studied data set, after eliminating them through efficient variable selection, the resulting model is more effective to generalize and classify the unseen data. In Experiment 4, after six variables are selected from the fine-tuned clustering algorithm, three more variables are filtered by C4.5. We now have a significantly smaller model regarding the variable complexity in Experiment 4 by HCT with only three variables instead of the original seventeen variables. However, we achieve the best classification accuracy of 87.7% on the test data in Experiment 4. Additional statistical *t*-test has shown that the *p*-values are 0.0000 and inidcates that the performance improvments are statisticaly significant at $\alpha = 0.05$ level .

5. CONCLUSIONS

It has been challenging to work with large quantities of all varieties of business data due to their large, complex dimensionality. This paper introduces a hybrid variable selection that integrates statistical cluster analysis, C4.5, and heuristics for best variable selection. Experimental results have shown that HCT outperforms traditional statistical methods. However, HCT shall go through extended tests on other data set. There may be some architectural issues worth exploring to improve the robustness and efficiency.

REFERENCES

[1] Bala, J., Huang, J., Vafaie, H., DeJong, K., and Wechsler, H. (1995) Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification. *IJCAI Conference*.

[2] DeJong, K. (1988) Learning with Genetic Algorithms: an Overview. *Machine Learning Vol. 3*.

[3] Kumar, A. (1998) New Techniques for Data Reduction in a Database System for Knowledge Discovery Applications. *Journal of Intelligent Information Systems,10,31-48.*

[4] Marcoulides, G.A., and Hershberger, S.L. (1997) *Multivariate Statistical Methods.*

[5] Piatetsky-Shapiro, G. and Frawley, W. (1991) *Knowledge Discovery in Databases.*

[6] Quinlan, J. R. (1993) *C4.5: Programming for Machine Learning*.

[7] Riply, B. D. (1996) *Pattern Recognition and Neural Networks*.

[8] Vafaie, H. and DeJong, K. (1992) Genetic Algorithms as a Tool for Variable Selection in Machine Learning. *Proceedings of the 4th International Conference on Tools with Artificial Intelligence*.

Parallelizing Feature Selection

Jerffeson Souza[*] Nathalie Japkowicz[†] Stan Matwin[‡]

Extended Abstract

The Feature Selection problem involves discovering a subset of features such that a classifier built only with this subset would attain predictive accuracy no worse than a classifier built from the entire set of features. Several algorithms have been proposed to solve this problem. For a detailed description of feature selection algorithms, please see [1]. In the last several years, we have witnessed a continuous and fast increase in the size and number of databases. This fact has stimulated data mining/machine learning researchers to seek more cost-effective approaches, since scalability with respect to large databases has become a more urgent issue. The main problem is that most machine learning tasks are inherently complex and almost always non-polynomial in the number of features of a dataset. In this scenario, parallelism is a pragmatic and promising approach to cope with the problem of cost-efficient machine learning. In particular, *feature selection* is one area in machine learning that can benefit from parallelism.

In recent years, researchers have shown great interest in applying parallelism for improving data mining algorithms/paradigms. In feature selection, surprisingly not many attempts have been made at applying parallelism. In [4], the authors propose a parallel algorithm based on the Sum of Squared Differences strategy designed to select features in images. Experimental results suggest the approach lends itself well to parallelization. Other researchers have used the inherently parallel characteristic of genetic algorithm to design new parallel feature selection algorithms. A parallel variant of genetic algorithm is used in [5]. Here, sets of individuals (representing subsets of features) of a single population are passed out to individual processors for evaluation with K-nearest-neighbor. Since evaluation dominates the rest of the GA operations, the approach can achieve near linear speedup. In [3], parallelism is applied over a genetic algorithm to create a new wrapper feature selection system. Authors report improved accuracies when applying this approach on a near-infrared (NIR) spectroscopic application and linear speedup.

Parallelizing a sequential algorithm may constitute a hard task depending on what kind of problem such an algorithm solves and how the algorithmic solution is organized. However, we can identify a few clear characteristics that make certain algorithms lend themselves better to parallelism than others. They are: Easy Partitioning, Independent Partitioning and Easy Load Balancing.

Based on these characteristics we studied the potential of parallelization of the several feature selection algorithms and classified these algorithms (Table 1) using the following classes: Hard Parallelization, Easy Parallelization and Obvious Parallelization.

Algorithm	**Parallelization**
Best-First Search	Easy
Genetic	Obviuos
LVF	Hard
Relief	Obvious
Focus	Easy
Forward Wrapper	Obvious
FortalFS	Obvious

Table 1: Potential of Parallelization of FS Algorithms.

We have also presented, discussed and evaluated a parallel version of the FortalFS feature selection algorithm [6]. The design of ParallelFortalFS was based on the Master-Slave design pattern [2]. In ParallelFortalFS, the master process distributes the work among slave processes, that will calculate a local best subset, and computes the global best subset from these results. This implementation of ParallelFortalFS will require a minimum number of iterations between processes. In fact, master and slaves will communicate only once when the slaves are created and another time when slaves send their local best. In addition, slaves will not communicate among themselves. The reduced amount of communication provided by this implementation makes us expect and predict a high efficiency level in practice.

The master process starts by running a feature selection system and calculating the Adam vector. This first part is done sequentially. Next, this component is

[*]Computer Science Department, Federal University of Ceará, Fortaleza, Ceará, 60.455-760, Brazil.

[†]School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada.

[‡]School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada.

responsible for starting the slaves in their corresponding processors (hosts) and distributing the work equally among them. In order to accomplish that, it will simply divide the total number of iterations by the number of available processors (slaves) and assign this new number of iterations to each of the slave processes. Finally, the master will receive each slave's local best subsets and calculate the global optimum. The slave component of ParallelFortalFS will iteratively generate a new subset according to the Adam vector, evaluate this subset with a ML system and update local variables that will store the best subset and its accuracy. The final local best subset and its corresponding accuracy will then be sent to the master process.

The ParallelFortalFS algorithm was implemented in Java with RMI (Remote method Invocation). The initial evaluation of ParallelFortalFS was run on a 100-Mbps local network composed of Pentium 4 PCs running Windows NT. We tried ParallelFortalFS with 1, 2, 4, 8 and 16 computers (processors), using the filter LVF as initial feature selection algorithm, $10 \cdot N$ iterations (N is the initial number of features in the dataset) and 5 UCI datasets (mushroom(22 features), ionosphere(34 features), splice(60 features), sonar(60 features) and audiology(69 features)).

For each dataset, we calculated the following values according to the equations below:

$$Speedup_p = \frac{TotTime_1}{TotTime_p} \tag{0.1}$$

$$OptSpeedup_p = \frac{TotTime_1}{(SeqTime + \frac{ParTime_p}{p})} \tag{0.2}$$

$$ParEff_p = \frac{Speedup_p}{OptSpeedup_p} \tag{0.3}$$

$$TotEff_p = \frac{Speedup_p}{p} \tag{0.4}$$

where p is the number of processors used, $TotTime_1$ is the total elapsed execution time (considering both the sequential and the parallel parts of the algorithm) when only one processor is used. $TotTime_p$ is the total execution time when p processors are used. $SeqTime$ is the execution time of the sequential part of the code alone and $ParTime_p$ the parallel execution time for p processors. Thus, $TotTime_p = SeqTime + ParTime_p$.

From the results obtained from our experiments, we can reach a few conclusions about the ability of ParallelFortalFS to speed up the FortalFS algorithm. We describe a few of these conclusions next:

1. ParallelFortalFS achieves a very high speedup when we use 2 processors, with an average parallel efficiency of 99.34% and total efficiency of 96.84%.

2. Total efficiency drops as the number of processors being used increases. Despite that, ParallelFortalFS as a whole was able to run 12 times faster on average for the five datasets when using 16 processors.

3. The drop in the total efficiency can be attributed to the sequential fraction of the code, since the sequential execution time becomes more relevant as the parallel execution time decreases.

4. If we consider the parallel efficiency alone, ParallelFortalFS is able to maintain a high performance in all cases, with averages of 99.34%, 97.34%, 98.10%, 96.44% and 93.74% for 1, 2, 4, 8 and 16 processors, respectively. Thus, this parallel implementation achieves near optimal efficiency in most cases.

References

[1] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[2] F. Buschmann. *The Master-Slave Pattern.*, pages 133–142. Pattern Languages of Program Design. Addision-Wesley, 1995.

[3] N. Melab, S. Cahon, E. Talibi, and L. Duponchel. Parallel GA-Based wrapper feature selection for spectroscopic data mining. In Bob Werner, editor, *Proceedings of the 16th International Parallel and Distributing Processing Symposium (IPDPS'02)*, pages 201–201, Ft. Lauderdale, Florida, USA, April 2002.

[4] B.N. Miller, N.P. Papanikolopoulos, and J.V. Carlis. A parallel feature selection algorithm. Technical Report UMSI 95/55, University of Minnesota Supercomputing Institute, apr 1995.

[5] W.F. Punch, E.D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In Stephanie Forrest, editor, *Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, pages 557–564, San Mateo, CA, 1993. Morgan Kaufmann.

[6] J.T. Souza, S. Matwin, and N. Japkowicz. *Feature Selection with a General Hybrid Algorithm.* PhD thesis, University of Ottawa, School of Information Technology and Engineering (SITE), Ottawa, ON, 2004.

Optimal Division for Feature Selection and Classification

(Extended Abstract)

Mineichi Kudo[†] **and Hiroshi Tenmoto**[††]

[†] **Department of Information Engineering, Faculty of Engineering Hokkaido University, Japan**
[††]**Deparment of Information Engineering Kushiro National College of Technology, Japan**

Abstract

Proposed is a histogram approach for feature selection and classification. The axes are divided into equally-spaced intervals, while the division numbers differ among axes. The main difference from the similar approaches is that feature selection is embedded in the model selection criterion. As a result, this criterion brings feature selection for a small number of training samples and convergence to the optimal Bayes error for a large number of training samples.

Keywords: Soft feature selection, Histogram, MDL, Convergence, Bayes error

1 Formulation

Let us consider to make a classifier from n training samples in m-dimensional Euclidean space $U = R^m$. Here, a sample is given by $x = (x_1, x_2, \ldots, x_m) \in R^m$, and a training sample sequence z^n is denoted by $z^n = (x,y)^n = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\} \in (U \times Y)^n$ with the class set $Y = \{1,2,\cdots,c\}$.

According to the MDL principle [1], we measure the cost of sending the class-label sequence y^n under the assumption that a receiver knows x^n, c and m. Let $L(\boldsymbol{\phi}|x^n)$ be the bit length needed to send the classifier $\boldsymbol{\phi}$ (the classification rule). In addition, let $L(S|\boldsymbol{\phi},x^n)$ the cost of sending the class-label sequence information when $\boldsymbol{\phi}$ is given. Then, the total cost is written as

$$L(y^n,\boldsymbol{\phi}|x^n) = L(y^n|\boldsymbol{\phi},x^n) + L(\boldsymbol{\phi}|x^n).$$

In our case, $L(\boldsymbol{\phi}|x^n)$ is the bit length to send information of the histogram, that is, the division information of each axis, and $L(S|\boldsymbol{\phi},x^n)$ is the sum of the bit length of the class-label sequences in cells forming the histogram. We assume that $x_1, x_2, \ldots, x_n$ as well as cells can be ordered in some way, e.g., a dictionary order with numerical order.

1.1 MDL coding

What we think of as a classifier is a histogram. We divide ith axis into 2^{d_i} equally-spaced intervals. The ends of each axis are determined by the minimum and maximum values over training samples. Thus, a partition is expressed by m-tuple $\mathbf{d} = (d_1, d_2, \ldots, d_m)$. By d we denote the sum of division indexes as $d = \sum_{i=1}^m d_i$. Then there are $\Pi_{i=1}^m 2^{d_i} = 2^d$ cells.

We want to find the optimal division $\mathbf{d}$ in some sense. In our case, we use MDL criterion for this goal. In the MDL criterion, a shorter length means a better partition for classification.

Through evaluation of the code lengths of individual parts, the total bit length is give by

$$\begin{aligned}
& L(y^n,\boldsymbol{\phi}|x^n) \\
= & L(y^n|\boldsymbol{\phi},x^n) + L(\boldsymbol{\phi}|x^n) \\
= & \log_2 R + \log_2 \binom{R}{R^M} + R^P \log_2 c \\
+ & \sum_{r=n1}^{R^M} \left\{ n^r H(\frac{n_1^r}{n^r}, \frac{n_2^r}{n^r}, \ldots, \frac{n_c^r}{n^r}) + \frac{c-1}{2}\log n^r \right\} \\
+ & \log_2 m + \log \binom{m}{m-m^+} \\
+ & \log_2^* d + dH(\frac{d_1}{d}, \frac{d_2}{d}, \cdots, \frac{d_m}{d}) - \frac{m-1}{2}\log_2 d \\
+ & \log_2 d + \sum_{i=1}^{m-2} \log_2^+ d_i \\
\simeq & \left(RH(\frac{R^P}{R}, \frac{R^M}{R}) + \frac{1}{2}\log_2 R + R^P \log_2 c \right) \\
+ & \left(n\sum_{r=1}^{R^M} \frac{n^r}{n} H(\frac{n_1^r}{n^r}, \frac{n_2^r}{n^r}, \ldots, \frac{n_c^r}{n^r}) + \frac{c-1}{2}\sum_{r=1}^{R^M} \log n^r \right) \\
+ & \left(\log_2 m + mH(\frac{m^+}{m}, \frac{m-m^+}{m}) \right)
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \left(\log_2^* d + dH(\frac{d_1}{d}, \frac{d_2}{d}, \cdots, \frac{d_m}{d}) + \frac{m-1}{2}\log_2 d\right) \\
&= \quad I + II + III + IV \qquad (1)
\end{aligned}
$$

Here, n_i^r is the number of samples of class i in cell r, R^P is the number of class-pure cells, and R^M is the number of class-mixture cells.

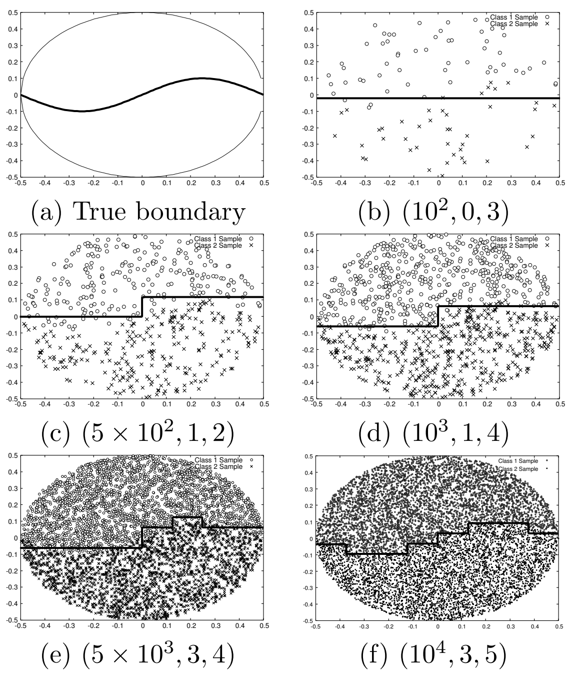
Let us examine how our criterion (1) works. First of all, it is noted that with this criterion our classification approaches to Bayes optimal classifier as n goes to infinity. Next let us examine (1) term by term. The dominant terms are I, II and IV. When the perfect classification on training samples is done by a certain $\mathbf{d}$, II vanishes because of $R^M = 0$ and $R^P = R$. Then, the problem reduces to minimize terms I and IV. Thus, what should be done is firstly to minimize the value of d and then to minimize the entropy of $\{d_i/d\}$. This tendency holds for general cases. That is, this enhances feature selection in which some d_i's are expected to be zero to decrease the entropy. This is the biggest difference from previous similar MDL approaches [2, 3, 4].

2 Experiments

We carried out an experiments on an artificial dataset (Fig. 1 (a)). The results are shown in Figs.1, 2 and 3. We can see that feature selection succeeded for a small sample size and the boundary approaches to the optimal one for a large sample size.

3 Conclusion

We have seen that an MDL-based histogram works in double directions: (soft) feature selection for a limited number of training samples and convergece to the optimal Bayes classifier. Especially, from the simple division numbers, it was confirmed that we could know the degree of importance of each feature even if no removal of the feature was done.



(a) True boundary (b) $(10^2, 0, 3)$ (c) $(5 \times 10^2, 1, 2)$ (d) $(10^3, 1, 4)$ (e) $(5 \times 10^3, 3, 4)$ (f) $(10^4, 3, 5)$

Figure 1. Change of classification boundary. The caption is (n, d_1, d_2).

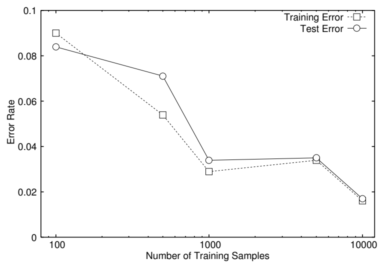
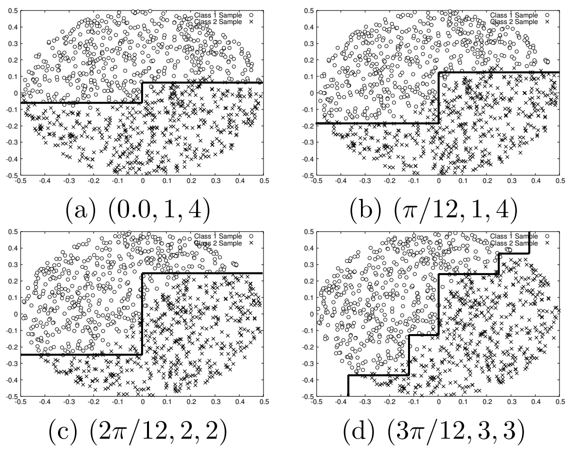


Figure 2. Training and test error.



(a) $(0.0, 1, 4)$ (b) $(\pi/12, 1, 4)$ (c) $(2\pi/12, 2, 2)$ (d) $(3\pi/12, 3, 3)$

Figure 3. Soft feature selection ($n = 1000$). The caption is (θ, d_1, d_2).

References

[1] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15 of *Series in Computer Science*. World Scirntific, 1989.

[2] J. Rissanen and B. Yu. MDL leraning. In D. W. Kueker and C. H. Smith, editors, *Learning and Geometry: Computational Approaches*, pages 3–19. Birkhauser, 1998.

[3] K. Yamanishi. A learning criterion for stochastic rules. In *The Third Workshop on Computational Learning Theory*, pages 67–81, 1990.

[4] H. Tsuchiya, S. Itoh, and T. Mashimoto. An algorithm for designing a pattern classifier by using MDL criterion. *IEICE Trans. Fundamentals*, E79-A(6):910–920, 1996.