Data Mining and Neural Networks[1]

Iveta Mrázová
Department of Software Engineering, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

František Mráz
Department of Software and Computer Science Education, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract: Especially the emerging technologies of artificial neural networks, fuzzy logic and evolutionary programming provide essential tools for designing intelligent data mining systems. In this article, we present a brief summary of various data mining techniques with the emphasis on techniques based on artificial neural networks. In general, it is relatively complicated to explain and to visualize what and why the neural networks are doing. From this perspective, we will compare several recent models of neural networks which could help us to extract and to interpret a set of clear and simple rules providing insight into how is a particular model working.

1 Introduction

Recently, data base technologies are the well-established means for storing and exploring large volumes of data. Statistics and many techniques from artificial intelligence represent time proven tools for modeling the data and analyzing their mutual dependencies. For many years, these techniques were developed independently one on the other. But due to a number of reasons that occurred in 1990s they were put together in data mining. The main factors were the production and warehousing of large amounts of data, available computing power and a high competitive pressure requiring optimization of decision-making based on automated methods for learning from the past.

Data mining means in principle exploring and analyzing large quantities of data with the aim to discover their mutual relationships. Many data mining techniques started few years or decades ago, mostly in the field of artificial intelligence. But their widespread availability and acceptance is only just beginning. Data mining algorithms typically require multiple passes over large volumes of data and many of them are computationally intensive. Anyway, several trends are increasing the necessity of powerful data mining tools – an increasingly service-based economy, the advent of mass customization and the competitive importance of information.

Data mining is an interactive and iterative process consisting of several steps: selection of data, its pre-processing, transformation, adequate "mining" and interpretation of obtained results. The area of data mining brings together ideas and techniques from a variety of fields – economics, artificial intelligence (AI), databases and statistics. To the tasks well suited for data mining belong classification, estimation, prediction, affinity grouping, clustering and description. The goal of a data mining task might be e.g. to allow a corporation to improve its marketing, sales, and customer support operations through better understanding of its customers. Further, data mining techniques can help e.g. to spot the most valuable customer but also to pick out those that should be e.g. turned down for a loan.

[1] This research was supported by the grant No. 157/1999/A INF/MFF of the GA UK and by the grant No. 201/99/0236 of the GA ČR.

Data mining proceeds following one - or both - basic styles:

a) Hypothesis testing, is a top-down approach that attempts to verify or disprove preconceived ideas concerning relationships present in the data.

b) Knowledge extraction, is a bottom-up approach that starts in the data and tries to "discover" something new that we did not know before. There are known two basic approaches to knowledge extraction – directed and undirected. The directed one is characterized by the presence of a target field whose value is to be predicted. The undirected one attempts to find patterns or similarities among groups of data records without the knowledge of particular target fields.

In the following text, we will concentrate on knowledge extraction. The most common data mining techniques applicable for this purpose are: cluster detection, memory-based reasoning, market basket analysis, decision trees, link analysis, artificial neural networks (ANN), fuzzy logic, support vector machines, and genetic algorithms. First, we will briefly state the tasks which can be solved by means of data mining. Next, we will characterize the above-mentioned data mining techniques. In Section 4, we will focus on the so-called BP-networks already used in data mining. Further, we will discus various directions for using some recent models of ANN which, at least from our point of view, could have better knowledge extraction abilities than standard models of ANNs.

2 Data mining tasks

In practice, the most important data mining tasks include: classification, estimation, prediction, affinity grouping, clustering and description. *Classification* consists of examining the features of a newly presented object and assigning it to one of a predefined set of classes. The classification is characterized by a well-defined definition of a limited number of the classes, and a training set consisting of pre-classified examples. The objective is to build a model that can be applied to unclassified data in order to classify it, i.e. to assign the data to the one or other class. Decision trees and memory-based reasoning are techniques well suited to classification. Link analysis is also useful for classification under certain circumstances.

While classification deals with discrete outputs, *estimation* deals with continuously valued outputs. In practice, estimation is often used to perform a classification task. E.g. instead of classifying objects into two classes, we can estimate the probability that the given object belongs to the first class. Now, the classification task now comes down to establishing a threshold score. Objects with a score greater than or equal to the threshold are classified as belonging to the first class and objects with a lower score are considered to be from the second class. Neural networks are well suited to estimation tasks.

The nature of *prediction* is the same as in the case of classification or estimation except that the records are classified according to some predicted future behavior or estimated future value. In a prediction task, the only way to check the accuracy of the classification is to wait and see. Anyone of the techniques used for classification and estimation can be adapted for use in prediction by using historical data for training examples such that the value of the variable to be predicted will be already known. When this model is applied to current inputs, the result is a prediction of future behavior.

Affinity grouping helps to determine which things go together. A prototypical example is to determine what things go together in a shopping cart in a supermarket. For this reason, the techniques used for affinity grouping are often called market basket analysis. *Clustering* is the task of segmenting a heterogeneous population of data into a number of more

homogeneous subgroups or clusters. Compared with classification, clustering does not rely on predefined classes – there are no predefined classes and no examples with predicted or estimated future values. The data records are grouped together on the basis of self-similarity. It is up to the user to determine what meaning, if any, to attach to the resulting clusters. Clustering is often performed as a kind of prelude to some other form of data mining or modeling.

The purpose of ***description*** is to describe what is going on in a complicated database. Its main goal is to increase the understanding of the people, products, or processes that produced the data in the processed database. An adequate description of a behavior often suggests also its explanation. Some of the data mining tools, e.g. the market basket analysis, are purely descriptive. Others, like neural networks in general, provide very little with regard to nothing description.

3 Data mining techniques

There can be found many approaches in the literature to solve the above-sketched data mining tasks. In this section, we provide a brief overview of the main data mining techniques. Most of them are treated in [5], for the rest of them we give explicit references. Cluster Detection is inherently an undirected data mining technique. The goal of cluster detection consists in finding previously unknown similarities in the data with techniques including geometric methods, statistical methods, and neural networks. Cluster detection provides a very good starting point for any further analysis of the data. Market Basket Analysis is a form of clustering used for finding groups of items that tend to occur together in a transaction (or market basket). The resulting information can be used e.g. for planning store layouts, limiting specials to one of the products in a set that tend to occur together.

Memory-Based Reasoning (MBR) is a directed data mining technique that uses known instances as model to make predictions about unknown instances. MBR looks for the nearest neighbors in the given data set and combines their values to assign the classification or prediction values to unknown patterns. The distance to the neighbors gives a measure of correctness of the results. A major advantage of MBR represents its ability to learn new classifications merely by introducing their new instances into the database. For instance, if we want to determine whether a new customer warrants to pay off installments, we would find similar customers – neighbors – in the data set and make the „give-a-credit“, „investigate-further“ or „refuse-immediately“ decision based on the status of the neighbors.

A powerful model used for classification represent also the so-called Decision Trees. They divide the patterns in the training set into pairwise disjoint subsets, each of which is described by a simple rule on one or more pattern features. This allows evaluating the results, identifying their key characteristics. Link Analysis is an application of graph theory constructs to data mining. It follows relationships between the particular data patterns. Relationships between customers are becoming increasingly important, especially as marketing groups focus more on customers, households, and economic marketing units instead of specific accounts. The few tools available focus on visualizing the links, rather than analyzing the patterns.

Artificial Neural Networks ([3], [8]) represent probably the most common data mining technique, perhaps a synonym with data mining to some people from the area of economics. These challenging models being originally inspired by their biological counterparts are still exploited for mathematical description of brain functioning, and, at a higher level, for modeling of mental processes in cognitive computing. On the other hand of this research

effort there are engineers who adopt these neural models for solving important practical tasks in AI-applications where standard analytical solutions are not always adequate and/or encounter difficulties.

Neural networks learn in general from a training set, generalizing the knowledge incorporated somehow inside it in order to classify or predict future data. Neural networks can also be applied to undirected data mining (in the form of self-organizing feature maps and related structures) and time-series prediction. To the most often used neural network paradigms belong the so-called feed-forward neural networks of the back-propagation type (BP-networks; [14]) based on supervised training and the so-called self-organizing feature maps (SOFMs; [9]) using self-organization for their adjustment.

Neural networks have two major drawbacks. The first refers to poor understanding of models they produce. But in many applications an interpretation of the network function is necessary or at least desirable. In such a case, methods of Fuzzy Logic can be used. Fuzzy logic ([4]) can be conceptualized as a generalization of classical logic. Modern fuzzy logic was developed by Lofti Zadeh in the mid-1960s to model those problems in which imprecise data must be used or in which the rules of inference are formulated in a very general way making use of diffuse categories. The second drawback of neural networks consists in their sometimes high sensitivity to the form of incoming data. Different data transformations can lead to different results; therefore, setting up the data is a significant part of their successful application.

Support Vector Machines (SVM) introduced by Vapnik and co-workers are learning systems that use linear discriminant functions in a transformed high dimensional feature space. This paradigm employs the so-called supporting vectors, which are patterns situated most closely to the separating hyperplane. In the few years since its introduction, this method has already outperformed many other systems in a wide variety of applications ([6]). SVM can be applied to similar tasks like neural networks.

Genetic Algorithms ([7]) employ genetics and natural selection to a search for optimal sets of parameters that correspond to a predictive function. By means of selection, crossover, and mutation operators they evolve successive generations of solutions. As the generations evolve, only the most predictive individuals should survive, until the fitness function converges to an optimal solution. Genetic algorithms have also been used to enhance other data mining models, like e.g. MBR and neural networks ([15], [16]).

4 Neural networks in data mining

This section is focused on BP-networks and some of their numerous modifications designed with the aim to improve their characteristics inevitable for data mining. An artificial neural network consists of basic units called neurons. Each neuron has several inputs (with associated weights) that it combines into a single output value. The neurons can be mutually inter-connected such that the outputs of some neurons can be used as inputs of other neurons. In a feed-forward neural network, the neurons are grouped together in a sequence of layers. Each neuron in the first layer (called the input layer) is connected to exactly one input. In the following layers, each neuron receives as inputs the outputs of all the neurons in the preceding layer. The last layer (called the output layer) is connected to the output of the network. All the layers between the input layer and the output layer are called hidden layers.

In the standard BP-network, each neuron calculates its output by multiplying the value on each input by its corresponding weight, summing these up, and applying the sigmoidal transfer function to the computed sum. A neural network can have any number of hidden

layers, but for most practical tasks, one or two hidden layers are sufficient. The wider the layer the greater the capacity of the network to recognize patterns. Anyway, for too large layers, the neural network can recognize patterns-of-one by memorizing each of the training examples (from the training set). Therefore, this problem is sometimes denoted as "overtraining". Obviously, we want the networks to generalize the training set, not to memorize it.

The aim of the standard Back-Propagation training algorithm was to find a set of weights that ensures that for each input vector contained in the training set the actual output vector produced by the network equals the desired output vector. This requirement has the form of the so-called objective function. Actual or desired states of hidden neurons are not specified by the task and the learning procedure has to decide under what circumstances the hidden neurons should be active in order to help achieve the desired input-output behaviour. The Back-Propagation training algorithm involves a forward pass through the network to estimate the error at the output layer, and then a backward pass to determine also the error at the hidden neurons. Then, the synaptic weights are modified with the aim to decrease the error at the output layer and to achieve the desired input-output behaviour.

More details can be found e.g. in [3]. Solving practical tasks, there are often trained many networks with different parameters like learning step, initial weight values, architecture etc. Afterwards, the network with the best wanted properties is selected and actually applied in the solution. The standard BP-training algorithm has a lot of advantages. However, we have also to deal with several limitations:

a) Relatively slow convergence of the standard BP-training algorithm – moreover, sometimes the learning process does not end with a network working satisfactorily even on the training set, or the training results in an "overtrained" network.

b) An almost "unclear" inner network structure with poor generalization – the training algorithm neither specifies explicitly, what hidden neuron outputs are appropriate for processing an input pattern in a proper way, nor specifies the number of hidden neurons. Moreover, it is extremely difficult to „guess“ the real meaning of every particular hidden neuron for a proper network output.

c) Poor robustness of trained networks – intuitively, for an input patterns with only a small deviation from a certain training patterns the network should give approximately the same output. However, this can be a difficult problem especially for those input patterns lying in the border area between pattern clusters.

d) Poor reusability of already trained networks in the solution of other tasks – even „slight“ modifications of the given task (represented e.g. by another dimension of the particular input/output vector) or by a modification of the training set (e.g. by adding and/or removing some training patterns) may often lead to a renewed training process.

The standard BP-training algorithm has relatively good approximating and classification properties and using it we can often form an internal structure that seems to be appropriate for a particular task domain. The outputs of all neurons in a hidden layer computed for a given input pattern is called internal representation of the input pattern. In [10], we proposed a training algorithm for forcing the so-called condensed internal representation. The aim of this algorithm is to group the outputs of the hidden neurons around three possible values – 1, 0 and 0.5. In the AI-terminology, this corresponds to creating rules of an expert system, where active neuron states indicate “yes”, passive states “no” and the so-called silent states stand for those cases where “no decision is possible”.

In order to prevent the network from forming very similar internal representations for different output patterns already during the training process, the new objective function will consist of the standard BP-error function enlarged by the representation error function and an ambiguity criterion ([11]). The internal knowledge representation is forced to form rather in the hidden neurons than exclusively by means of large weight values typical for "overtrained" networks. In this way, forcing the unambiguous condensed internal representation can help to overcome the danger of "overlearning". Such a "transparent" internal representation seems to be very important for the design of the optimal network architecture.

With regard to the reusability of already trained networks, the elimination of unnecessary hidden and input neurons seems to be an important aspect. In many cases, a reduced number of hidden neurons improves generalization capabilities of the trained network at the expense of its accuracy. On the other hand, an insufficient number of hidden neurons may result in an inadequate capability to separate the concerned pattern clusters. There are known many attempts to determine the right number of hidden neurons – almost all based on the empirical experience. However, a satisfactory general rule for the selection of the right number of hidden neurons has not been developed yet.

Some approaches initialize the training process with more hidden neurons than needed. Later on, some neurons may be discovered to be unnecessary and can be removed from the network. Those hidden neurons yielding a redundant information are often characterized by constant output values for all input patterns from a given set or by the same or "opposite" output values than those of another neuron from the same layer – and this for all considered input patterns. A BP-network having no such redundant neurons is said to be reduced. Further, it can be shown that for every BP-network and a set of input patterns there exists a reduced BP-network yielding the same output vector as the original BP-network. Alternative approaches based on incremental approximation start the training process with a small number of hidden neurons (maybe 1). During training, new hidden neurons are added and their weights are updated ([1], [2]).

Solving practical tasks, there are usually trained many networks with different parameters like learning step, initial weights, architecture etc. But training large BP-networks with a complicated structure can be difficult and time-consuming. Moreover, even „slight" modifications of the given task represented for example by another dimension of the particular input/output vector or by a modification of the training set (given e.g. by adding and/or removing some training patterns) may often lead to a renewed training process.

In such a case, although having fast training algorithms, it would be more efficient to extract or to get a kind of a "module library" and to "incorporate" these modules into the solutions of other and possibly even more complex tasks ([11]). Such smaller parts of the entire neural network can be treated easier (e.g. with respect to the desired internal representation or robustness). In this context, especially the recognition of superfluous pattern features and pruning of unnecessary hidden and/or input neurons seems to play an important role in solving practical problems in data mining.

For the standard BP-training algorithm, there are usually no restrictions concerning the number of hidden or even inevitable input neurons needed for a "correct solution" of a given task. Therefore, it is very difficult to "guess" the really meaning of every particular hidden neuron for a proper network output. Good training accuracy can be achieved by forming very complex decision boundaries, which in turn requires a large network size. However, a good training accuracy does not necessarily guarantee a satisfactory robustness and/or generalization capabilities of the trained network. Actually, for a trained network we can find

an "ε-equivalent" network with better robustness ([11]). Outputs of the ε-equivalent network can differ at most by a fixed (arbitrarily small) ε from the outputs of the original network.

5 Conclusions and further research

Another very important issue in data mining is represented by the ability to detect significant (novel) input patterns and to identify their characteristic features. This property could be used both for training and optimizing the structure of the data mining model at hand and for improving its characteristics – generalization abilities, robustness, storage capacity, etc. A suitable means for the detection of significant input patterns and features seems to be the internal knowledge representation. Most probably, the internal representation of significant input patterns will differ „somehow“ from the known and „not too exceptional“ ones. This could help to gain insight into the structure of the processed data, or to detect irregularities and errors in it.

Therefore, our further research will be focused on the mechanisms of creating and visualizing an appropriate internal knowledge representation in modular and hybrid data mining systems. The extracted knowledge will be applied for finding significant input patterns and their characteristics with a stressed attention to selected economical problems. In this framework, we plan to integrate various techniques for designing intelligent data mining systems. The models under consideration include the so-called generalized relief error networks (GREN-networks; [12], [13]) acting like an advisor for other models based on error-correction learning (usually BP-networks). The major advantages of GREN-networks consist in that they do not require „well-balanced“ training sets for training other networks. Further, they are able (to a certain extent) to „simulate“ the evolution of presented input patterns and evaluate to corresponding error.

The detection of significant input patterns and pattern features already before or during the training process can save computational time and means. The reason for this expectation consists in the same observation like e.g. in SVMs - that the most important training patterns lie close to separating hypersurfaces (on the border of the respective pattern clusters). Anyway, other patterns should be considered for training too – however rather according to the extent of their current significance, instead of simply omitting them. We expect that such a strategy could speed up the training process significantly. Further, we hope that the identification of the most significant pattern features could reduce drastically the number of necessary input neurons and hence lead to a simpler architecture of the chosen model as well.

On the other hand, GREN-networks still lack an illustrative kind of internal representation of „correct“ and „suspicious“ input patterns and their mutual topological relationship in the underlying feature space present e.g. in SOFMs or in Fuzzy ART-Maps. A transparent and visualized representation of significant pattern features, patterns and their mutual relationships among the data set could namely indicate e.g. „suspicious“ positive outliers. Such input patterns lying far from separating hypersurfaces might satisfy the prescribed criteria but at the same time they can be situated far from the respective pattern cluster. This could indicate a „suspicious“ input pattern but also an extremely good customer.

If we would be able to visualize the topological relationship among the data, we could e.g. tailor our loan products for each respective customer. Such a kind of financial engineering would allow to suggest and to adjust the proposed financial products such that they would better suit to the intended group of customers. Moreover, we expect that the application of GREN-networks would allow (at least to a certain extent) to predict the „evolution“ of

considered customers and to „identify“ their most significant features that should or could be improved together with the amount of the suggested change.

References

[1] G. Andrejková: Application of the Approximation Theory by Neural Networks, in: Neural Network World, No. 5, Vol. 10, 2000, pp. 787-795.

[2] G. Andrejková: Incremental Approximation by Layer Neural Networks. Proceedings ISCI'2000, Physica-Verlag, (2000), pp. 15-20.

[3] M. Anthony, P. L. Bartlett: Neural Network Learning: Theoretical Foundations, (1999), Cambridge University Press.

[4] C. H. Chen (Ed.): Fuzzy Logic and Neural Network Handbook, (1996), McGraw-Hill, New York.

[5] M. J. A. Berry, G. Linoff: Data Mining Techniques For Marketing, Sales, and Customer Support, (1997), John Wiley & Sons, New York.

[6] N. Christianin, J. Shawe-Taylor: An Introduction to Support Vector Machines and other kernel-based learning methods, (2000), Cambridge University Press.

[7] Z. Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs, (1999), Springer-Verlag, Berlin.

[8] S. Haykin: Neural Networks: A Comprehensive Foundation, (1999), Prentice Hall, Upper Saddle River, N. J.

[9] T. Kohonen: Self-Organizing Maps, (1995), Springer-Verlag, Berlin.

[10] I.Mrázová: Condensed Knowledge Representation in BP-networks, in: Proc. of the ISE'94, Hamburg, IEE, (5 - 9 Sept. 1994), pp. 118-123.

[11] I. Mrázová: Internal Representation in BP-networks, PhD-Thesis, 1997, ÚIVT AV ČR, Prague.

[12] I. Mrázová: Generalized Relief Error Networks and Patterns with Lower Errors, accepted for publication in the International Journal of Smart Engineering System Design, (January 2001).

[13] I. Mrázová: Controlled Learning of GREN-Networks, accepted for publication in: Proc. of the ANNIE 2001, St. Louis, (4 – 7 Nov. 2001).

[14] D. E. Rumelhart, G. E. Hinton, R. J. Williams: Learning Representations by Back-Propagating Errors, in: Nature, (323) (1986), pp. 533-536.

[15] D. J. Schaffer: Combinations of Genetic Algorithms with Neural Networks or Fuzzy Systems, in: J. M. Zurada, R. J. Marks II, C. J. Robinson (Eds.): Computational Intelligence Imitating Life, (1994), IEEE Press, pp. 371-382.

[16] H. J. Zimmerman: Hybrid Approaches for Fuzzy Data Analysis and Configuration Using Genetic Algorithms and Evolutionary Methods, in: J. M. Zurada, R. J. Marks II, C. J. Robinson (Eds.): Computational Intelligence Imitating Life, (1994), IEEE Press, pp. 364-370.