Natural Data Mining Techniques

J. N. Kok and W.A. Kosters
Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Email: {joost,kosters}@liacs.nl

1 Data Mining

The field of data analysis, or data mining as it is called nowadays, is concerned with extracting knowledge from data. It is a new, emerging discipline, having its roots in several other fields, cf. [9]. Many ideas have originated from Artificial Intelligence, see for instance [18]. The main task in data mining is to find hidden relationships among data attributes and to formulate these relationships in a general understandable manner. This general knowledge can then be applied in newly encountered situations.

Our treatment of the subject follows the line of [1], see also [11]. We shall try to focus on those areas of data mining that have interesting theoretical aspects; in particular we do not deal with privacy issues and problems with missing or noisy data, and practical applications are only briefly mentioned. Statistical methods are also not covered; we will try to deal with data mining from the background of computer science. However, it would be very interesting to elaborate upon the connection between statistical methods and the methods mentioned in the sequel.

Data mining is seen as the key element in the so-called knowledge discovery in databases (KDD). KDD is not a new technique. It is an interdisciplinary area in which machine learning, statistics, database technology, expert systems and data visualization come together. The KDD process consists of six stages: data selection, cleaning of data, enrichment of data (for example using external data bases), coding, data mining and reporting. In data selection the data most relevant to the problem at hand is selected, leaving out redundant data. In the cleaning process, the obvious flaws in the data are corrected. In the coding process, data is coded such that it can be used in data mining algorithms. In the data mining process, the actual inference takes place. In the reporting process the results are of course reported, usually in a visually attractive way.

KDD is closely connected to an other important development which is called data warehousing. A data warehouse is a central storage facility in which operational data is stored. These warehouses usually contain enormous amounts of data. This gives a new view on information management. The combination of KDD and data warehousing allows companies to use the knowledge hidden in the data as a production factor, instead of just support for their operations. A usual problem in the field of data mining is that the combination of databases (often called "data fusion") is not straightforward; a typical example is a situation where individual customer data have to be related to statistical data. In such a situation traditional database techniques, e.g., SQL queries, are clearly not sufficient. Data mining and query tools are complementary instruments. Data mining does not replace query tools. Data mining looks for hidden and unexpected information, for example "what is an optimal segmentation of the customers" (i.e., find customer profiles). Sometimes the questions are rather vague: what are the trends in customer behavior?

Often data mining techniques are integrated in software packages ("embedded data mining"). As an example, one can envisage a sales-system in which automatically profiles of customers are derived (e.g., [13]) or a financial administration in which certain kinds of frauds are detected. It is also clear that many Internet applications will contain data mining algorithms, and in these situations response times should be as short as possible.

Why has data mining become so popular? All major organizations have created large data bases about customers, competitors and products. The increased amount of computing power, computer memory and connectivity (for example through the Internet) makes it easier to connect different databases and to manipulate them. Many new techniques have been proposed to deal with the information in an adaptive way (like inductive logic programming, neural networks and genetic algorithms). With client-server technology many more users have access to data bases. For example marketing managers now also want to use the opportunities.

As a simplified example of a data mining application, we try to relate the gender of a person to other attributes based on the following data:

case-id	shoe-size	eye-color	...	height	gender
case_1	42	*blue*	...	182	*M*
case_2	43	*brown*	...	179	*M*
...	...	...	...	...	.
case_N	36	*blue*	...	169	*F*

Figure 1: Example: A very simple database

Data analysis can result in the conclusion that "eye-color" is irrelevant with respect to a persons gender, but the following general rule might hold (at least in the presently given data set):

IF shoe-size is small AND height is low THEN gender is F

In other words, and the other way around, women are short and have small feet. This piece of knowledge can be applied for prediction, i.e., if a person is described in terms of those attributes that are used in the above table, but his/her gender is not known, our rule will be able to give a well-educated guess.

Marketing is based on educated guesses. For instance, before launching a direct mail campaign for a new product a selection of promising prospects should be made. Typically, historical data on former campaigns are available and the expected response for the new campaign is guessed using these data. In particular, a general rule can be created that relates profile attributes (age, financial situation etcetera) to observed response behavior (did or did not respond). This rule can then be applied to predict response behavior in the new campaign. It is clear that the quality of the prediction rule is of utmost importance.

There are several techniques that can extract knowledge from data in the form of general rules. Although these techniques can differ substantially, they all work along the following lines:

1. Collect historical data that contain profile attributes and also attributes to be predicted.

2. Divide historical data into two parts: training data and validation data.

3. Apply the given technique to the training data in order to generate a general rule that relates profile attributes to attributes to be predicted.

4. Apply the generated rule to the validation data and compare the rule's predictions to the actual values of the attributes to be predicted.

Due to possible inconsistencies and incompleteness of the data and the complex character of the relationships between attributes, rules will typically be only partially correct, even on the training data. A rule's performance on the validation data can be even lower, since these data are "new" to the rule. Nevertheless, it is exactly the performance on the validation data that tells the quality of a rule: if it generalizes to unknown data, it can be applied in new situations.

Within the data mining techniques there is a group of relatively simple general applicable pattern recognition techniques including induction of decision trees, neural networks and genetic algorithms. Different data mining techniques are typically compared according to a number of features:

1. Efficiency. The amount of computational effort needed to obtain a good generalization of the data, i.e., a general rule.

2. Effectivity. The quality of the general rule in terms of its performance on unknown data.

3. Interpretability of the outcome. Techniques differ in their "language". The final outcome (the rule) has a specific form, which might or might not be readable to humans. Technically this makes little difference, but in practice human approval of a rule (based on experience, intuition, common sense etcetera) is desirable for

application in a business environment. In the light of Occam's razor, simplicity of the rule is also desirable. Visualization of the outcome can be a problem in its own.

2 Techniques Used in Data Mining

In this section we briefly describe some of the techniques that are frequently used in data mining.

2.1 Association Rules

Discovery of association rules (see [2]) is an interesting subfield of database mining. Association rules simply try to find associations or correlations between various attributes in a data base. One might think of items that are purchased together in a supermarket. This may lead to association rules like "beer $\Rightarrow$ chips (93%)". This rule indicates that, if beer is bought, chips are bought with a 93% certainty. In this setting 93% is called the *confidence* of the rule. Besides the confidence the so-called *support* is important: this is the number of people satisfying the rule. It is easy to find rules with high confidence, but with poor support. It is one of the tricks of the trade to discover rules with both sufficiently high confidence and support.

The use of association rules is obvious: they can be used for promotions, store layout, advertising, and so on. Besides supermarkets, there are numerous other applications of association rules. Finding association rules is not as trivial as it may seem, because the number of possible combinations of attributes grows exponentially with the number of fields in the database, whereas the number of customers or transactions may be very large.

In [2] the Apriori algorithm is presented, which is the basis for many algorithms in this area. This algorithm proceeds as follows. Suppose we have a lexicographically ordered set of all "frequent k-itemsets". A frequent k-itemset is an ordered set of k products with support higher than or equal to some minimum support, the so-called support threshold. The fact that every k element subset of a frequent $(k+1)$-itemset is a frequent k-itemset, justifies the following argument. If we want to find all frequent $(k+1)$-itemsets, it is sufficient to look at pairs of two frequent k-itemsets that only differ in the last product; the union of these two sets is a candidate, which can be pruned if not all its k element subsets are frequent. Now the candidate is passed through the database, thereby computing its support, and either accepted or refused. Notice that the lexicographic ordering of the frequent k-itemsets facilitates the generation of candidates; in particular, every candidate is generated only once.

In the beginning much attention was given to details concerning the data structures that supported the algorithm, in particular in order to minimize the number of memory accesses. Later—when larger and faster memories became available—attention shifted to reduction of database passes and to generalized problems, e.g., finding association rules in taxonomies ([19]), detection of economically interesting features ([8]) and clustering ([12]). This lead to rules like "beer $\Rightarrow$ food" or "the best 100-product shop is: ..." or "the second most significant group consists of people who ...".

2.2 K-Nearest-Neighbor

K-Nearest-Neighbor is a technique that is well-suited for classification. It works as follows. Suppose we have a large database of classified subjects, e.g., credit-card users that have been classified as being fraudulent or non-fraudulent. This classification was made by observation. Each subject is described by a number of independent attributes, e.g., zip-code, age, profession, and so on. If a new subject comes in that is not yet classified, we look for the classified subjects in the database that are very similar to the new subject. Then we can make a classification prediction by assigning the new subject to the class that occurs most frequent in its "neighborhood", consisting of the K nearest subjects. In the example: if we want to predict whether a person is fraudulent or not, we look at the persons most similar to him/her with respect to zip-code, age, profession, and so on. If "fraudulent" occurs often among those "neighbors", we classify the new person as "fraudulent", otherwise as "non-fraudulent". It is of course a difficult task to develop a suited distance function. More information can be found in [7].

2.3 Decision Trees

Decision trees can also be used for classification. They make a hierarchic division of the input space in order to explain the observed classification. This results in a number of if-then-else rules that can be organized in a tree structure. Generally, the resulting classification rules are relatively easy to understand.

The best known algorithm is Quinlan's C4.5 (its predecessor ID3 is also famous), see [17]. The algorithm constructs a decision tree by repeatedly adding the best nodes using ideas from information theory (such as entropy). The most discriminating nodes (questions) are privileged.

2.4 Neural Networks

During the last decade, neural networks have gained a lot of popularity. Although the use of neural networks is new to a lot of application areas, the first neural network was already developed in 1958 by Frank Rosenblatt. This network was called the *perceptron*. After a period of silence, neural networks were made popular again by David Rumelhart and James McLelland in the mid 80's.

Generally speaking we can say that neural networks are autonomous parallel distributed information processing systems inspired by the human brain. They consist of interconnected simple processing elements called *neurons*, communicating with one another via weighted connections. These neurons are schematic and simplified versions of biological neurons. A single neuron is depicted in Figure 2.

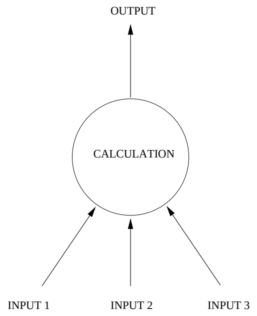

Figure 2: A single neuron. The arrows represent connections.

Although a single neuron can only perform a very simple task (like adding all the inputs and outputting a 1 if a certain threshold is exceeded, a 0 if it is not), a whole network of such neurons is able to perform complex tasks.

Many different types of neural networks exist. There are types that can be used to find hidden structure in data, types that can be used for forecasting, clustering, etcetera. As a non-technical introduction to neural computation we recommend [6]. Excellent technical introductions are [7] and [10].

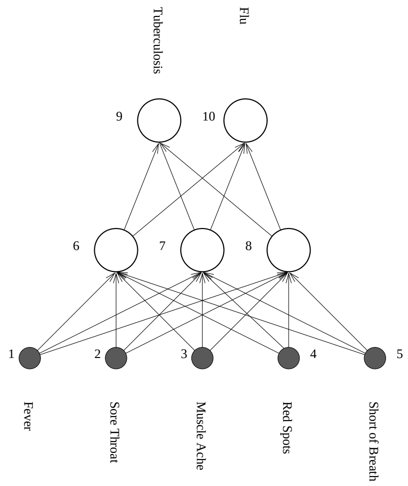

Figure 3: A typical feedforward neural network.

In Figure 3, a hypothetical and simplified example of a neural network application in a medical diagnosis system is given. This figure depicts a two layer feedforward neural network (a certain type of network). The input neurons are represented by small grey circles. The hidden and output neurons are represented by white circles. The arrows that connect the units in the different layers represent connections among the units. The direction of an arrow is the direction in which a signal is propagated through the network. Usually the backpropagation algorithm is used to update the weights of the connections during the training phase: the differences between expected and real outputs are propagated back from output to input layer, meanwhile pushing weights in the preferred directions.

This neural network can now be used to determine whether somebody suffers from tuberculosis (TB) or the flu by inputting the observed symptoms. The network then outputs a 1 on the tuberculosis neuron if the patient suffers from TB, a 0 on the tuberculosis neuron if the patient does not suffer from TB, a 1 on the flu neuron if the patient suffers from the flu, and a 0 on the flu neuron if the patient does not suffer from the flu. A correct output, however, can only be obtained from the neural network after it has seen a number of examples. In this case these examples are a number of patients with known diagnosis. The neural network is then able to adapt itself to give a correct classification on new patients. Neural networks are in general able to adapt themselves to numerous classification and regression tasks.

2.5 Evolutionary Computation

Evolutionary computation is based on the principle of Darwinian evolution. In particular, a population of individuals is evolved by reproduction and selection. Reproduction takes place by means of recombination, where a new individual is created by mixing the features of two existing individuals, and mutation, where a new individual is created by slightly modifying one existing individual. Applying reproduction increases the diversity of the population. Selection is to reduce the population diversity by eliminating certain individuals. To have this mechanism work, it is required that a quality measure, called *fitness*, of the individuals is given. If reproduction is applied to the best individuals (mating of the fittest principle) and selection eliminates the worst individuals (survival of the fittest principle), then on the long run the population will consist of individuals having high fitness values—the population is evolving. An overview of the field can be found in [3].

It is easy to see how this mechanism can be applied to problem solving. The definition of a given problem always specifies the set of possible solutions, e.g., all possible prediction rules. Clearly, possible solutions differ in quality, e.g., some rules perform better on the data set than others. If a possible solution is viewed as an individual, then we can define its fitness as its quality with respect to the criteria in the given problem, e.g., the fitness of a prediction rule can be the percentage of correct predictions on the given data—the higher the better. The only thing one has to do in order to solve the given problem in an evolutionary manner is to define how individuals are reproduced (recombined and mutated) and specify how selection works. Typically, both reproduction and selection are probabilistic mechanisms, meaning that they are subjects to random effects. For instance, selection of surviving individuals is random, but individuals with a higher fitness have a higher probability of being selected to survive in the next generation. There are two remarks to be made on evolutionary problem solving methods. First, that a(n optimal) solution cannot be guaranteed with full certainty because of the probabilistic character of the evolutionary process. Consecutive generations, however, contain better and better individuals and the process can be terminated when a solution of sufficient quality has emerged. Second, that the evolution mechanism is fully problem independent: it can be applied to evolve prediction rules, optimal vehicle routes, or game playing strategies (see [15]). Practical applications in management span a wide range of areas including trade, financial services, marketing, or strategic planning, cf. [5].

2.6 Inductive Logic Programming

The general aim of Inductive Logic Programming (ILP) is to develop theories, techniques and applications of inductive learning from observations and background knowledge in a first order logical framework. ILP systems use background knowledge (or domain knowledge) and data to derive a hypothesis, in the form of a set of rules, from the data. The references [18] and [14] can be used as an introductions to this subject.

Several ILP systems are operational (e.g., FOIL, LINUS, GOLEM that are described in [14]); the language of choice is often PROLOG. They have been used for various data analysis tasks, for example analysis of datasets from molecular biology and the processing of natural language datasets.

2.7 Fuzzy Logic and Rough Sets

As is already clear in the Example in Section 1, notions such as height may be precise, but they sometimes should be categorized in terms like "low", "medium" and "high". We then enter the field of Fuzzy Logic, where set membership is no longer Boolean: it may assume real values between 0 and 1. This happens to be a situation that occurs quite often in data mining applications. In [4] a short overview of the field can be found.

In Rough Set theory (cf. [16]) a similar viewpoint is taken. Elements that certainly do or do not belong to a given set determine approximations to this set. Both Fuzzy Logic and Rough Sets inspired many data mining applications, and can be used alongside other methods.

3 Summary

Let us conclude with some remarks on the character of these techniques. As for the applicability to data mining the following should be mentioned. All techniques are directly applicable to machine learning tasks in general, and to data mining problems in particular. As mentioned in Section 1, methods can be compared according to three criteria, efficiency, effectivity and interpretability of the outcomes. As for efficiency, evolutionary and neurocomputation techniques and inductive logic programming may require long run times, ranging from a couple of minutes to a few hours. This however is not necessarily a problem. Namely, the long running times are needed to *find* a solution to a data mining problem, but once a solution is detected, *applying* such a solution in a new situation can be done fast. Concerning the issue of effectivity, we can generally state that the more complicated techniques (like neural networks) give the best results. However, this is situation dependent and one has to take the time/quality tradeoff into account. Concerning interpretability of the results, one can say that the simple techniques are generally the easiest to interpret.

As already mentioned in Section 1, we have not discussed statistical methods such as regression and Multi Dimensional Scaling. In all practical situations it is important to use these well-founded and extensively studied methods, in particular since they are often easy to handle and quite fast. They have as another advantage that they usually also offer reliability bounds on their results.

References

[1] P. Adriaans and D. Zantinge. *Data Mining.* Addison-Wesley, 1996.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[3] T. Bäck, J.N. Kok, J.M. de Graaf, and W.A. Kosters. Theory of genetic algorithms. *Bulletin of the EATCS*, 63:161–192, 1997.

[4] M. Berthold. Fuzzy logic. In M. Berthold and D.J. Hand, editors, *Intelligent Data Analysis, An Introduction*, chapter 8, pages 269–298. Springer-Verlag, 1999.

[5] J. Biethahn and V. Nissen (eds.). *Evolutionary Algorithms in Management Applications.* Springer-Verlag, 1995.

[6] J.P. Bigus. *Data Mining with Neural Networks: Solving Business Problems—From Application Development to Decision Support.* McGraw-Hill, 1996.

[7] C.M. Bishop. *Neural Networks for Pattern Recogition.* Oxford University Press, 1995.

[8] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In S. Chaudhuri and D. Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 254–260. ACM Press, 1999.

[9] D.J. Hand. Intelligent data analysis: Issues and opportunities. In X. Liu, P. Cohen, and M. Berthold, editors, *Advances in Intelligent Data Analysis*, volume 1280 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1997.

[10] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[11] M. Holsheimer and A.P.J.M. Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, Amsterdam, 1994.

[12] W.A. Kosters, E. Marchiori, and A.A.J. Oerlemans. Mining clusters with association rules. In D.J. Hand, J.N. Kok, and M. Berthold, editors, *Advances in Intelligent Data Analysis, IDA-99*, volume 1642 of *Lecture Notes in Computer Science*, pages 39–50. Springer-Verlag, 1999.

[13] W.A. Kosters, H.A. La Poutré, and M.C. van Wezel. Understanding customer choice processes using neural networks. In H.F. Arner Jr., editor, *Proceedings of the First International Conference on the Practical Application of Knowledge Discovery and Data Mining (PADD '97)*, pages 167–178. The Practical Application Company, London, 1997.

[14] N. Lavrac, I. Weber, D. Zupanic, D. Kazakov, O. Stepankova, and S. Dzeroski. ILPNET repositories on WWW: Inductive logic programming systems, datasets and bibliography. *AI Communications*, 9(4):157–206, 1996.

[15] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, third edition, 1996.

[16] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers, 1991.

[17] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[18] S. Russell and P. Norvig. *Artificial Intelligence.* Prentice-Hall, 1995.

[19] R. Srikant and R. Agrawal. Mining generalized association rules. In U. Dayal, P.M.D. Gray, and S. Nishio, editors, *Proceedings of the 21st VLDB Conference*, pages 407–419. Morgan Kaufmann, 1995.

This paper has appeared in the Bulletin of the European Association for Theoretical Computer Science, Volume 71, June 2000, pages 133-142.