Discovering knowledge in commercial databases using modern heuristic techniques*

B. de la Iglesia, J.C.W. Debuse and V.J. Rayward-Smith
University of East Anglia
Norwich
NR4 7TJ
UK
{*bli,jcwd,vjrs*}*@sys.uea.ac.uk*

Abstract

In this paper we describe our experiences of using simulated annealing and genetic algorithms to perform data mining for a large financial service sector company. We first explore the requirements that data mining systems must meet to be useful in most real commercial environments. We then look at some of the available data mining techniques, including our own heuristic techniques, and how they perform with respect to those requirements. The results of applying the techniques to two commercial databases are also shown and analysed.

Introduction

Interest by industry in what is known as *Data mining* or *Knowledge Discovery in Databases* has increased dramatically, as companies see more opportunities for exploiting the vast volumes of data they already hold for other purposes. The information extracted from these massive databases can bring very large financial rewards as it is translated into better customer targeting, improved fraud detection, etc.

This interest has been matched by increased research efforts by various research communities including those working in database systems, artificial intelligence, statistics and data visualisation. Each of these communities has an important role to play in the development of data mining as an integrated approach to data analysis. However, it is important to note that data mining techniques must include an element of automated inductive learning to qualify as such, and practical techniques should be developed in the light of lessons learnt from their application to real, commercial databases. Furthermore, some techniques may yield results that can be considered as valid data mining solutions, but do not meet all the requirements of the commercial environment. In this paper we examine those requirements, test the current data mining techniques against them, and present our results and conclusions.

*This research was sponsored by the Teaching Company Scheme under grant no. GR/K 39875

Data mining: the complete process

The process of data mining consists of various phases, each encompassing a series of related activities (Limb & Meggs 1994).

- Phase I: The data is prepared, cleansed and studied. Statistical techniques, visualisation and preprocessing can be used in this phase.
- Phase II: We try to construct a model of the data or to extract some interesting patterns that can be applied to characterise new data. The process is one of inductive learning and can take the form of supervised or unsupervised learning. Note that we do not need to construct a total model that characterises each class and each instance of the data; finding any knowledge about a class (or even part of a class) that characterises it may represent clear gains from a commercial point of view.
- Phase III: The usefulness of the characterisation previously built is measured. This might be achieved by using statistical tests or visualisation of the results.

There is a danger of concentrating all the research effort on the second phase, in the belief that this is the most important part of the process. However, it must be understood that each of these phases is important in the overall data mining process, and must be performed to some extent.

The requirements of the commercial environment

We have been performing data mining case studies for a large financial company for some time. The company was already using statistical techniques to analyse their data, which means they had already realised the importance of it. They had a steady supply of good quality data. Our brief was to study the data mining techniques available and their suitability to the company's problems and environment.

When solving business problems, the techniques used to solve those problems and the solutions to the problems should exhibit certain key features that are desirable or necessary, as discussed in (Goonatilake & Treleaven 1995). For our specific problem and environment, the relevant features in order of priority are:

• Automated learning: The need for human intervention should be kept to a minimum. This will ensure that expense associated with human experts is avoided, and also the influence of their often 'subjective' views will not affect the learning process.

• Explanation: The knowledge extracted must be in a form that can be understood and analysed by humans. There are two main reasons for this. First, any decisions in this industry may involve very large amounts of money, and management are not enthusiastic about embracing ideas they cannot understand or analyse for themselves. Second, explanations must often be given to customers when a decision that affects them is reached. This is sometimes a legal requirement.

• Flexibility: This refers to the ability of the technique employed to cope with imprecise, noisy or incomplete information.

• Control: The techniques should allow us to steer the search for knowledge to specific areas, or to conduct a general search. In other words, control of the process must be retained by the user. This will allow us to search for rules describing exceptions. In these cases, the technique is required to produce a rule to describe a minority of records in the database. There are many applications for a technique that can find patterns for exceptions in a database, as this is quite a common occurrence.

• Adaptation: Business is constantly changing. Therefore, a good data mining system should be able to monitor its performance and revise its knowledge as the environment in which it operates changes.

With this set of desirable properties in mind, we examined some techniques in the context of two case studies. As they did not meet all the above requirements, the development of our own data mining tools using heuristic search techniques became necessary.

Choosing suitable techniques according to the requirements of the commercial environment

There are many techniques that can be used for data analysis, and they are increasingly being referred to as data mining techniques. The first division that can be found in the literature (Holsheimer & Siebes 1994) is that of supervised learning (classification) and unsupervised learning (clustering). In this paper we restrict the study to classification techniques. However, as previously discussed, we do not need to create a complete classifier; we are interested in any knowledge that can be used to characterise new data, bearing in mind that the techniques used to extract it should exhibit the key features discussed. In fact, for many problems we are interested in describing specific patterns that apply to a particular class only (for example, describing exceptions).

Under the generic name of classification techniques we find in the literature: neural networks, tree induction, rule induction, statistical classification and modern heuristic techniques. Not all of those are suitable to our comparative study.

Using the set of key features as an initial selector, we can eliminate neural networks from our list of prospective candidate techniques. Neural networks are regarded at present as 'black box' techniques, because they do not provide *explanations* of the knowledge in a form suitable for verification or interpretation by humans, although increasing research efforts are being made in this area.

Tree induction

Tree induction techniques produce decision trees, covering the whole data set, as a result of their classification process. There are two approaches to decision tree construction. The first approach was developed by the machine learning community and includes algorithms such as C4.5 (Quinlan 1993). The second approach is based on statistical theory and includes CART (Breiman *et al.* 1984) and XAID/CHAID methods (Kass 1975). We will examine these tree induction methods separately.

Machine learning methods The trees produced by the most highly developed machine learning algorithms tend to be extremely large for any real commercial database. As Quinlan puts it in (Quinlan 1987),

> Although decision trees generated by these methods are accurate and efficient, they often suffer the disadvantage of excessive complexity and are therefore incomprehensible to experts. It is questionable whether opaque structures of this kind can be described as knowledge, no matter how well they function.

These types of decision trees can therefore be described as black box techniques as well, and as such are not suitable for our problems and environment. The interpretation and validation of these trees by humans is either impossible, depriving the techniques of the 'explanation' feature, or achievable only by a great deal of human intervention, depriving the technique of the 'automated learning' feature.

A further problem of these decision trees is 'overfitting' and 'overspecialising'. This can be observed when a technique produces a model that has a zero error in classifying the training set of data, but much higher error when classifying new cases. Techniques to avoid overfitting exist. Pruning (Quinlan 1993),for example, grows the whole expanded tree and then examine subtrees in a bottom-up fashion, cutting off any branches that are weak. Pruning techniques attempt to produce simplified trees that do not overfit the data, but the trees obtained are often still very large indeed.

Statistical methods The statistical approach to tree induction is implemented in the package KnowledgeSeeker (Biggs, de Ville, & Suen 1991), based upon

the CHAID/XAID models. CHAID looks at a categorical variables, and XAID looks at numerical variables.

The difference between the tree building approach used in KnowledgeSeeker, and the machine learning tree building approaches is that the former uses the statistical significance hypothesis testing framework to construct the tree. At each step of growing the tree this significance testing is used first to determine the best attribute to branch on, if any, and also to establish the best way to cluster the values of the partitioning attribute. This means that the branches of the tree, for numerical attributes represent ranges of values, and for categorical variables represent groups of values, that are clustered together according to tests of statistical significance of the various possible groupings. This leads to more compact trees than the binary partitions of numerical attributes and the individual partitions of categorical attributes performed by C4.5, but it is still necessary to analyse them to extract the desired or relevant knowledge.

The main drawback of decision trees is that, even for compact examples, it may not be straightforward to see how each part of the tree, which represents a pattern of knowledge, is performing. For example, if we are trying to explain some exceptions we would have to look at the individual branches of the tree, extract those that have the required outcome and measure their performance. We can do this by interpreting the tree as a set of rules. If we do this, the knowledge extracted will exhibit not only the features automatic learning and explanation, but also the fourth key feature: *control.*

Trees as a set of rules A decision tree represents a set of decision rules, one for every terminal node in the tree. Most tree induction algorithms convert the tree into a set of rules automatically. However, the set of potential rules in this representation is limited. First, because a tree starts at a root node, all rules must begin with the same feature. Most importantly, the rules must be mutually exclusive. No two rules can be satisfied simultaneously. However, techniques for transforming the trees produced by C4.5 into a set of non-mutually exclusive rules exist (Quinlan 1993).

Rule induction

As the name implies, rule induction algorithms generate rules outright. We will use CN2, one of the best known rule induction algorithms, for the comparison.

We will also use the 1-R algorithm, which extracts rules based on a single attribute, and claims to produce rules of comparable quality to those produced by other methods.

Our heuristic techniques output rules. We have used two different meta heuristics: genetic algorithms and simulated annealing. We are also considering the use of another meta heuristic, tabu search, in the future. Hybrid techniques will also be investigated.

Simple rules, whether they are generated from a tree, by a rule induction algorithm, or by a meta-heuristic, represent knowledge in a clear and understandable way. They allow us to *control* the search for knowledge and guide it to particular areas by considering only those rules that cover the desired areas.

Classical statistical classification techniques

It is not our intention to compare our techniques to the statistical classification techniques, as we see both approaches as complementary. In our environment we use linear modelling to create models of the data. This would be comparable to a decision tree, or a set of rules that cover each record in the database. The comparison to rules containing isolated patterns would not be straight forward.

We have selected techniques that produce a comparable output, viz rules. We can therefore establish a uniform base for comparison of the three sets of techniques. Furthermore, automatically extracted simple rules will display three of our most important key features: automated learning, explanation and control.

A framework for comparison

Let us assume we have a flat file, D, of d records and that all fields have known values for each $r \in D$. We will refer to these records as *defined records.* Therefore, D consists of a set of records specifying values for the attributes $A_1, A_2, \ldots, A_n$ over the domains $Dom_1, Dom_2, \ldots, Dom_n$, respectively. We are asked to find a rule of the form $\alpha \Rightarrow \beta$ which appears to hold for some records in this database. For simplicity, α, the precondition of the rule can be interpreted as a conjunction of the form

$$(A_{\delta_1} \in v_{\delta_1}) \wedge (A_{\delta_2} \in v_{\delta_2}) \wedge \ldots \wedge (A_{\delta_s} \in v_{\delta_s})$$

with $v_{\delta_1} \subseteq Dom_1, v_{\delta_2} \subseteq Dom_2 \ldots, v_{\delta_s} \subseteq Dom_s$. We can then construct more complex rules with preconditions in Disjunctive Normal Form (DNF) by taking the disjunction of various rules expressed as above to be one rule. The predictive part of the rule, β, can take the form of $(A_j \in v_j)$, where $v_j \subseteq Dom_j$.

We use $\alpha(r)$ to denote that α is true for record r. Associated with any rule $\alpha \Rightarrow \beta$ are three sets of records,

$$\begin{aligned} A &= \{r \mid \alpha(r)\},\ B = \{r \mid \beta(r)\} \text{ and} \\ C &= \{r \mid \alpha(r) \wedge \beta(r)\} = A \cap B. \end{aligned}$$

We can then define the integer values a, b and c by

$$a = |A|,\ b = |B| \text{ and } c = |C|.$$

Remembering that d denotes the size of the database, one can define various ratios between these values which measure properties of the rule. The two ratios which are of greatest interest for measuring the quality of our rules are as follows:

$App(\alpha \Rightarrow \beta) = \frac{a}{d}$ APPLICABILITY
$Acc(\alpha \Rightarrow \beta) = \frac{c}{a}$ ACCURACY

when the threshold percentage of this value is reached. The temperature value at the point of the last temperature rise is considered to be the initial temperature if no rises have yet been performed.

C4.5

The C4.5 package generates decision trees, which provide a classification for every object within a database. The package allows the decision tree to be simplified, using a pruning technique which reduces the size of the tree according to a user-defined confidence level.

C4.5 was used to produce decision trees for both databases. As expected, the trees produced for both problems were immense. If they had been used as a set of mutually exclusive rules, most rules would have an unsatisfactorily low level of applicability.

A 5% confidence level value was used for pruning, giving significantly smaller trees. However, even the pruned trees were massive, and not appropriate for assimilation by humans.

A rule with the desired output was extracted from both pruned and unpruned trees, for each database. The best rule from the two extracted was selected for comparison with other approaches. Two rule extraction methods were used; these are described below.

C4.5 allows the automated extraction of production rules from the decision trees which it produces. These have the advantage of being generally much simpler to understand and significantly more compact than the decision tree. However, for the first database, the majority class was predicted, regardless of the value of the input fields. For the second database, the rule produced was of inferior quality to that produced using the following extraction mechanism.

As we have described previously, the company was interested only in rules which predicted a predetermined output value, which we will denote β_d. Let $b(node_x)$ denote the branch of the test at the node $node_x$ which classifies a subset within which there is the largest proportion of records with output value β_d. This branch is used as the α part of a rule $\alpha_x \Rightarrow \beta_d$. We denote the rule produced by this method $\alpha_r \Rightarrow \beta_d$.

Set current node $node_c$ to be the root of the decision tree;
$\alpha_r = \alpha_c$;
Compute the fitness of $\alpha_r \Rightarrow \beta_d$ using the same function and λ values as the GA and SA;
repeat
  Set the $node_c$ to equal the child of $node_c$ which classifies a subset within which there is the largest proportion of records with output value β_d;
  if fitness$(\alpha_r \wedge \alpha_c \Rightarrow \beta_d) >$ fitness$(\alpha_r \Rightarrow \beta_d)$ **then** $\alpha_r = \alpha_r \wedge \alpha_c$
until α_r remains unchanged

The performance of the rules obtained using this method is tabulated in the results section.

KnowledgeSeeker

The software KnowledgeSeeker (KS) was used to construct decision trees. The simplest tree produced using several sets of parameters was chosen to be converted into rules and used in our comparison. Each node of the tree, terminal and non-terminal, was considered as a rule as read from the root of the tree to that node.

CN2

The CN2 (Clark & Niblett 1989) algorithm was applied to the two problem databases, producing an unordered list of rules. Rules with the required output value were then tested using the same λ values as the SA and GA based packages; the best rule was then compared to rules produced using the other approaches.

A Laplacian measure of expected rule accuracy, which favours more general rules, was used to evaluate rules for selection by CN2. This is defined as $(c+1)/(a+k)$, where k is the number of classes in the domain.

Several setting of star size and significance threshold were experimented with for both problems. These parameters allow control over the search process and rule generality respectively.

1R

The 1R (Holte 1993) system, as part of the MLC++ (Kohavi *et al.* 1994) package, was also applied to both databases. This algorithm produces a rule which bases the classification of examples upon the value of a single attribute. The package initially discretises each field within the training data into intervals, each (except the rightmost) containing at most MIN_INST records. We used a MIN_INST value of 6 for our problems, since this was recommended in (Holte 1993) for most datasets. All objects contained within each interval are assigned the majority class for the interval.

Experimental results

The results for all methods are shown in figure 1. Figures in brackets refer to results achieved using the test database; # refers to the number of inequalities used within the rule. Acc. and App. refer to accuracy and applicability respectively.

Conclusions

In this paper we have looked at very simple rules produced by various techniques. These very simple rules represent significant patterns of knowledge extracted from databases, as it can be seen from their applicability/accuracy. We have also shown that these simple rules are in a format that can be used in a commercial environment as they represent understandable knowledge, extracted automatically, using a flexible method that can be easily controlled to look for specific class descriptions.

Data set	Tool	#	Fitness	Acc. (%)	App. (%)
1	SA	2	1195.61 (1133.36)	80.8 (80.4)	52.5 (52.6)
1	GA	6	1098.97 (1016.99)	84 (83.1)	36.6 (37)
1	C4.5	2	1196.06 (1122.66)	80.6 (80.1)	53.8 (53.8)
1	CN2	2	736.81 (756.75)	85.6 (85.9)	21.7 (22.3)
1	KS	2	494.92 (424.55)	76.3 (75.6)	40.3 (40.0)
1	1R	1055	964.92 (233.82)	75.8 (72.1)	86.7 (86.3)
2	SA	8	3675 (3783)	24.2 (24.4)	10.3 (10.3)
2	GA	8	3675 (3783)	24.2 (24.4)	10.3 (10.3)
2	C4.5	6	3210 (2750)	30.3 (26.7)	6.5 (6.5)
2	CN2	4	2682 (2471)	26.9 (25.5)	6.4 (6.3)
2	KS	3	2028 (1880)	23.8 (18.5)	5.8 (7)
2	1R	2	783 (1214)	9.2 (9.6)	41 (41.1)

Figure 1: Experimental results

We have looked at some of the state of the art techniques for data mining and shown that, for standard datasets, the rules obtained using heuristic techniques are of comparable quality to the rules induced using CN2, and to the rules extracted from C4.5 decision trees using our own method. The other techniques produced worse results.

For non-standard databases, such as our second case study, our methods are superior to all the others and do not require the modification of the database ('balancing') so that the exceptions represent a significant proportion.

The conclusion to our paper must be that we find patterns of very high quality because we set out to look for these patterns, rather than produce a model of the data and then search in the model for patterns. With tree induction methods, for example, every rule must contain the partition at the root of the tree, so the search for rules is immediately constrained. Although at the top of the tree, the partition chosen may be the best, there may be stronger patterns that do not contain this partition. The heuristic techniques described can search for rules freely, without committing themselves to including a particular partition. This freedom gives our method a strong advantage: the user can control the search for rules.

The rules produced using heuristics exhibit all but one of the aforementioned desirable features: automation, explanation, control and flexibility. The only feature missing is adaptability, and we plan to extend and enhance our techniques to include this.

References

Biggs, D.; de Ville, B.; and Suen, E. 1991. A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics* 18(1):49–62.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees.* Monterey, CA.: Wadsworth and Brooks.

Clark, P. C., and Niblett, T. N. 1989. The CN2 induction algorithm. *Machine Learning* 3(4).

Goonatilake, S., and Treleaven, P. C., eds. 1995. *Intelligent Systems for Finance and Business.* Chichester, West Sussex: Wiley.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Holsheimer, M., and Siebes, A. 1994. Data mining: The search for knowledge in databases. *Technical report CS-R9406, CWI Amsterdam.*

Holte, R. C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11.

Kass, G. V. 1975. Significance testing in automatic interaction detection. *Applied Statistics* 24(2):178–189.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimisation by simulated annealing. *Science* 220.

Kohavi, R.; John, G.; Long, R.; Manley, D.; and Pfleger, K. 1994. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, 740–743. IEEE Computer Society Press.

Limb, P. R., and Meggs, G. J. 1994. Data mining - tools and techniques. *BT Technol J* 12.

Lundy, M., and Mees, A. 1986. Convergence of an annealing algorithm. *Mathematical programming* 34.

Mann, J. W.; Kapsalis, A.; and Smith, G. D. 1995. The GAmeter toolkit. In Rayward-Smith, V. J., ed., *Applications of Modern Heuristic Methods.* Alfred Waller. chapter 12, 195–209.

Mann, J. W. 1995. X-SAmson v1.0 user manual. *University of East Anglia.*

Quinlan, J. R. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies* 27:221–234.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Rayward-Smith, V. J.; Debuse, J. C. W.; and de la Iglesia, B. 1995. Using a genetic algorithm to data mine in the financial services sector. In Macintosh, A., and Cooper, C., eds., *Applications and Innovations in Expert Systems III.* SGES Publications.