DATA MINING CLASSIFICATION

FABRICIO VOZNIKA
LEONARDO VIANA

INTRODUCTION

Nowadays there is huge amount of data being collected and stored in databases everywhere across the globe. The tendency is to keep increasing year after year. It is not hard to find databases with Terabytes of data in enterprises and research facilities. That is over 1,099,511,627,776 bytes of data. There is invaluable information and knowledge "hidden" in such databases; and without automatic methods for extracting this information it is practically impossible to mine for them. Throughout the years many algorithms were created to extract what is called nuggets of knowledge from large sets of data. There are several different methodologies to approach this problem: classification, association rule, clustering, etc. This paper will focus on classification which is described in more details in the next section.

PROBLEM DESCRIPTION

Classification consists of predicting a certain outcome based on a given input. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and the respective outcome, usually called goal or prediction attribute. The algorithm tries to discover relationships between the attributes that would make it possible to predict the outcome. Next the algorithm is given a data set not seen before, called prediction set, which contains the same set of attributes, except for the prediction attribute – not yet known. The algorithm analyses the input and produces a prediction. The prediction accuracy defines how "good" the algorithm is. For example, in a medical database the training set would have relevant patient information recorded previously, where the prediction attribute is whether or not the patient had a heart problem. Table 1 below illustrates the training and prediction sets of such database.

Training set

Age	Heart rate	Blood pressure	Heart problem
65	78	150/70	Yes
37	83	112/76	No
71	67	108/65	No

Prediction set

Age	Heart rate	Blood pressure	Heart problem
43	98	147/89	?
65	58	106/63	?
84	77	150/65	?

TABLE 1 – TRAINING AND PREDICTION SETS FOR MEDICAL DATABASE

Among several types of knowledge representation present in the literature, classification normally uses prediction rules to express knowledge. Prediction rules are expressed in the form of IF-THEN rules, where the antecedent (IF part) consists of a conjunction of conditions and the rule consequent (THEN part) predicts a certain predictions attribute value for an item that satisfies the antecedent. Using the example above, a rule predicting the first row in the training set may be represented as following:

```
IF (Age=65 AND Heart rate>70) OR (Age>60 AND Blood pressure>140/70)
THEN Heart problem=yes
```

In most cases the prediction rule is immensely larger than the example above. Conjunction has a nice property for classification; each condition separated by OR's defines smaller rules that captures relations between attributes. Satisfying any of these smaller rules means that the consequent is the prediction. Each smaller rule is formed with AND's which facilitates narrowing down relations between attributes.

How well predictions are done is measured in percentage of predictions hit against the total number of predictions. A decent rule ought to have a hit rate greater than the occurrence of the prediction attribute. In other words, if the algorithm is trying to predict rain in Seattle and it rains 80% of the time, the algorithm could easily have a hit rate of 80% by just predicting rain all the time. Therefore, 80% is the base prediction rate that any algorithm should achieve in this case. The optimal solution is a rule with 100% prediction hit rate, which is very hard, when not impossible, to achieve. Therefore, except for some very specific problems, classification by definition can only be solved by approximation algorithms.

APPROXIMATION ALGORITHMS

STATISTICAL ALGORITHMS: ID3 AND C4.5

The ID3 algorithm was originally developed by J. Ross Quinlan at the University of Sydney, and he first presented it in the 1975 book "Machine Learning". The ID3 algorithm induces classification models, or decision trees, from data. It is a supervised learning algorithm that is trained by examples for different classes. After being trained, the algorithm should be able to predict the class of a new item.

ID3 identifies attributes that differentiate one class from another. All attributes must be known in advance, and must also be either continuous or selected from a set of known values. For instance, temperature (continuous), and country of citizenship (set of known values) are valid attributes. To determine which attributes are the most important, ID3 uses the statistical property of entropy. Entropy measures the amount of information in an attribute. This is how the decision tree, which will be used in testing future cases, is built.

One of the limitations of ID3 is that it is very sensitive to attributes with a large number of values (e.g. social security numbers). The entropy of such attributes is very low, and they don't help you in performing any type of prediction. The C4.5 algorithm overcomes this problem by using another statistical property known as information gain. Information gain measures how well a given attribute separates the training sets into the output classes. Therefore, the C4.5 algorithm extends

the ID3 algorithm through the use of information gain to reduce the problem of artificially low entropy values for attributes such as social security numbers.

GENETIC PROGRAMMING

Genetic programming (GP) has been vastly used in research in the past 10 years to solve data mining classification problems. The reason genetic programming is so widely used is the fact that prediction rules are very naturally represented in GP. Additionally, GP has proven to produce good results with global search problems like classification. The search space for classification can be described as having several 'peaks', this causes local search algorithms, such as simulated annealing, to perform badly.

GP consists of stochastic search algorithms based on abstractions of the processes of Darwinian evolution. Each candidate solution is represented by an individual in GP. The solution is coded into chromosome like structures that can be mutated and/or combined with some other individual's chromosome. Each individual contains a fitness value, which measures the quality of the individual, in other words how close the candidate solution is from being optimal. Based on the fitness value, individuals are selected to mate. This process creates a new individual by combining two or more chromosomes, this process is called crossover. They are combined with each other in the hope that these new individuals will evolve and become better than their parents. Additionally to matting, chromosomes can be mutated at random.

The running time of GPs is usually controlled by the user. There are many parameters used to determine when the algorithm should stop, and each data set can have very different settings. In all cases, the best individual is stored across generations and is returned when the algorithm stops. The most commonly used parameter is number of generations. Another stop parameters used is minimum expected hit ratio, in which case the algorithm will run until a candidate solution has a hit ratio greater than expected. This however can cause the algorithm to run forever. Combinations of stop conditions can also be used to ensure stoppage.

NEURAL NETWORKS

Neural networks were modeled after the cognitive processes of the brain. They are capable of predicting new observations from existing observations. A neural network consists of interconnected processing elements also called units, nodes, or neurons. The neurons within the network work together, in parallel, to produce an output function. Since the computation is performed by the collective neurons, a neural network can still produce the output function even if some of the individual neurons are malfunctioning (the network is robust and fault tolerant).

In general, each neuron within a neural network has an associated activation number. Also, each connection between neurons has a weight associated with it. These quantities simulate their counterparts in the biological brain: firing rate of a neuron, and strength of a synapse. The activation of a neuron depends on the activation of the other neurons and the weight of the edges that are connected to it. The neurons within a neural network are usually arranged in layers. The number of layers within the neural network, and the number of neurons within each layer normally matches the nature of the investigated phenomenon.

After the size has been determined, the network is usually then subjected to training. Here, the network receives a sample training input with its associated classes. It then applies an iterative process on the input in order to adjust the weights of the network so that its future predictions are optimal. After the training phase, the network is ready to perform predictions in new sets of data.

Neural networks can often produce very accurate predictions. However, one of their greatest criticisms is the fact that they represent a "black-box" approach to research. They do not provide any insight into the underlying nature of the phenomena.

ANT COLONY

Ant Colony algorithms were first introduced in the 1992 PhD thesis of Marco Dorigo. They offer a way of finding good paths within a graph, and they were inspired by the behavior of ants in finding paths from the colony to food.

When traveling from their colony to food sources, ants deposit chemicals called pheromones on the trails. The trail is used so that ants can find their way back to the colony. If other ants find this path they are likely to follow it. This will cause more pheromone to be deposited on the trail, which has the effect of reinforcing it. However, if a trail has not been used for a while, the pheromone starts to evaporate. Short paths have the advantage of being marched over faster and more often, therefore, the pheromone density remains high. So a short path will have more ants traveling on it, increasing the pheromone density even more, and eventually all the ants will follow it.

Ant algorithms mimic this behavior in order to find optimal paths within a graph. Initially, all paths have a random small amount of pheromone deposited on it. An ant departs from the starting node and starts the process of visiting the other nodes in the graph. At each node, the ant decides which node it should visit next. The ant rates the attractiveness of traveling to each node in the graph. A nearby node with a large amount of pheromone deposited in its path will be very attractive. Also, a distant node with little amount of pheromone deposited in its path will be very unattractive. This attractiveness information will be use in order to compute the probability that a given route will be taken.

Ant algorithms give good results relatively fast. They can be run continuously in order to adapt to changes in real-time: an advantage as compared to genetic algorithms. For instance, an obstacle such as a traffic jam would quickly lead the ants to discover a different good path. This can be of interest in applications such as network routing and urban transportation.

PAPER DISCUSSION

The classification algorithm described in [Mendes et al. 2001] offers an interesting combination of approaches. It consists of main GP algorithm, where each individual represents an IF-THEN prediction rule, having the rule modeled as a Boolean expression tree. Matting is done by selecting a random subtree on each individual and then swapping them. Tests have shown that the algorithm has a tendency of creating very large rules. This happens because it is possible to create a 100% accurate rule by making a subrule for each row in the training data set and making them match. This indeed is optimal for the training set, but clearly performs badly with new data. To avoid this, the fitness function penalizes large rules. Interesting another cause of rule bloat is the opposite;

rules that do not match any of the rows in the training set. These rules do not affect the individual fitness value in the training set, but it can cause the rule to perform badly with real data. Additionally, it makes rules larger which consume more memory, and takes longer to evaluate. The algorithm deals with this by pruning all non-matching subtrees from the individuals.

Continuous attributes poses a challenge to such algorithms because rules impose hard constraints on attributes, e.g. `(Age>65 AND Heart rate>70)`. However, a patient does not start having heart problem on the day he turns 65, but the probability of having heart problems increase as he gets close to turning 65. The algorithm being discussed solves this problem by using fuzzy logic to describe continuous attributes. This allows for smooth transition between states. The challenge however is to partition the attribute correctly, i.e. when should the chance of heart attack start increasing and how fast is should increase are difficult questions to answers. So there is another genetic algorithm (GA) that evolves individuals representing fuzzy membership functions. In the beginning all continuous attributes are equally partitioned, then after each generation of the main GP, a generation of the GA runs to adjust the membership functions. The goal is to find membership functions that maximize the main GP fitness.

This algorithm showed accuracy of the resulting rule superior to other similar algorithms. This can be attributed in part to fuzzy rules and how it evolves along side with the prediction rules.

CONCLUSION

Data mining offers promising ways to uncover hidden patterns within large amounts of data. These hidden patterns can potentially be used to predict future behavior. The availability of new data mining algorithms, however, should be met with caution. First of all, these techniques are only as good as the data that has been collected. Good data is the first requirement for good data exploration. Assuming good data is available, the next step is to choose the most appropriate technique to mine the data. However, there are tradeoffs to consider when choosing the appropriate data mining technique to be used in a certain application. There are definite differences in the types of problems that are conductive to each technique. The “best” model is often found by trial and error: trying different technologies and algorithms. Often times, the data analyst should compare or even combine available techniques in order to obtain the best possible results.

REFERENCES

Data Mining.
March 2007. March 2007 <http://en.wikipedia.org/wiki/Datamining/>.

Classifying Text with ID3 and C4.5.
June 2001. March 2007 <http://www.ddj.com/184410304/>.

Data Mining Techniques.
2006. March 2007 <http://www.statsoft.com/textbook/stdatmin.html/>.

Ant Colony Algorithm.
January 2007. March 2007 <http://www.egilh.com/blog/archive/2007/01/08/3322.aspx/>.

[Mendes et al. 2001] R.F. Mendes, F.B. Voznika, A.A. Freitas and J.C. Nievola. Discovering fuzzy classification rules with genetic programming and co-evolution. *Principles of Data Mining and Knowledge Discovery (Proceedings of the 5th European Conference, PKDD 2001) – Lecture Notes in Artificial Intelligence 2168,* 314-325, Springer, 2001.

Genetic Programming, John R. Koza, MIT Press, 1998.

Genetic Programming An Introduction, W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, Morgan Kaufmann Publishers, 1998.

Data Mining and Knowledge Discovery with Evolutionary Algorithms, A.A. Freitas, Springer-Verlag, 2002.