Introduction to Data Mining and Knowledge Discovery

Third Edition

by
Two Crows Corporation

Related Readings

Data Mining '99: Technology Report, Two Crows Corporation, 1999
M. Berry and G. Linoff, *Data Mining Techniques,* John Wiley, 1997
William S. Cleveland, *The Elements of Graphing Data, revised,* Hobart Press, 1994
Howard Wainer, *Visual Revelations,* Copernicus, 1997
R. Kennedy, Lee, Reed, and Van Roy, *Solving Pattern Recognition Problems,* Prentice-Hall, 1998
U. Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* MIT Press, 1996
Dorian Pyle, *Data Preparation for Data Mining,* Morgan Kaufmann, 1999
C. Westphal and T. Blaxton, *Data Mining Solutions,* John Wiley, 1998
Vasant Dhar and Roger Stein, *Seven Methods for Transforming Corporate Data into Business Intelligence,* Prentice Hall 1997
Brieman, Freidman, Olshen, and Stone, *Classification and Regression Trees,* Wadsworth, 1984
J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, 1992

Introduction to Data Mining and Knowledge Discovery, Third Edition
ISBN: 1-892095-02-5

TABLE OF CONTENTS

Introduction to Data Mining and Knowledge Discovery

Introduction

Data mining: In brief

Databases today can range in size into the terabytes — more than 1,000,000,000,000 bytes of data. Within these masses of data lies hidden information of strategic importance. But when there are so many trees, how do you draw meaningful conclusions about the forest?

The newest answer is data mining, which is being used both to increase revenues and to reduce costs. The potential returns are enormous. Innovative organizations worldwide are already using data mining to locate and appeal to higher-value customers, to reconfigure their product offerings to increase sales, and to minimize losses due to error or fraud.

Data mining is a process that uses a variety of data analysis tools to discover patterns and relationships in data that may be used to make valid predictions.

The first and simplest analytical step in data mining is to **describe** the data — summarize its statistical attributes (such as means and standard deviations), visually review it using charts and graphs, and look for potentially meaningful links among variables (such as values that often occur together). As emphasized in the section on The Data Mining Process, collecting, exploring and selecting the right data are critically important.

But data description alone cannot provide an action plan. You must **build a predictive model** based on patterns determined from known results, then **test** that model on results outside the original sample. A good model should never be confused with reality (you know a road map isn't a perfect representation of the actual road), but it can be a useful guide to understanding your business.

The final step is to empirically **verify** the model. For example, from a database of customers who have already responded to a particular offer, you've built a model predicting which prospects are likeliest to respond to the same offer. Can you rely on this prediction? Send a mailing to a portion of the new list and see what results you get.

Data mining: What it can't do

Data mining is a tool, not a magic wand. It won't sit in your database watching what happens and send you e-mail to get your attention when it sees an interesting pattern. It doesn't eliminate the need to know your business, to understand your data, or to understand analytical methods. Data mining assists business analysts with finding patterns and relationships in the data — it does not tell you the value of the patterns to the organization. Furthermore, the patterns uncovered by data mining must be verified in the real world.

Remember that the predictive relationships found via data mining are not necessarily *causes* of an action or behavior. For example, data mining might determine that males with incomes between $50,000 and $65,000 who subscribe to certain magazines are likely purchasers of a product you want to sell. While you can take advantage of this pattern, say by aiming your marketing at people who fit the pattern, you should not assume that any of these factors *cause* them to buy your product.

To ensure meaningful results, it's vital that you understand your data. The quality of your output will often be sensitive to outliers (data values that are very different from the typical values in your database), irrelevant columns or columns that vary together (such as age and date of birth), the way you encode your data, and the data you leave in and the data you exclude. Algorithms vary in their sensitivity to such data issues, but it is unwise to depend on a data mining product to make all the right decisions on its own.

Data mining will not automatically discover solutions without guidance. Rather than setting the vague goal, "Help improve the response to my direct mail solicitation," you might use data mining to find the characteristics of people who (1) respond to your solicitation, or (2) respond AND make a large purchase. The patterns data mining finds for those two goals may be very different.

Although a good data mining tool shelters you from the intricacies of statistical techniques, it requires you to understand the workings of the tools you choose and the algorithms on which they are based. The choices you make in setting up your data mining tool and the optimizations you choose will affect the accuracy and speed of your models.

Data mining does not replace skilled business analysts or managers, but rather gives them a powerful new tool to improve the job they are doing. Any company that knows its business and its customers is already aware of many important, high-payoff patterns that its employees have observed over the years. What data mining can do is confirm such empirical observations and find new, subtle patterns that yield steady incremental improvement (plus the occasional breakthrough insight).

Data mining and data warehousing

Frequently, the data to be mined is first extracted from an enterprise data warehouse into a data mining database or data mart (Figure 1). There is some real benefit if your data is already part of a data warehouse. As we shall see later on, the problems of cleansing data for a data warehouse and for data mining are very similar. If the data has already been cleansed for a data warehouse, then it most likely will not need further cleaning in order to be mined. Furthermore, you will have already addressed many of the problems of data consolidation and put in place maintenance procedures.

The data mining database may be a logical rather than a physical subset of your data warehouse, provided that the data warehouse DBMS can support the additional resource demands of data mining. If it cannot, then you will be better off with a separate data mining database.

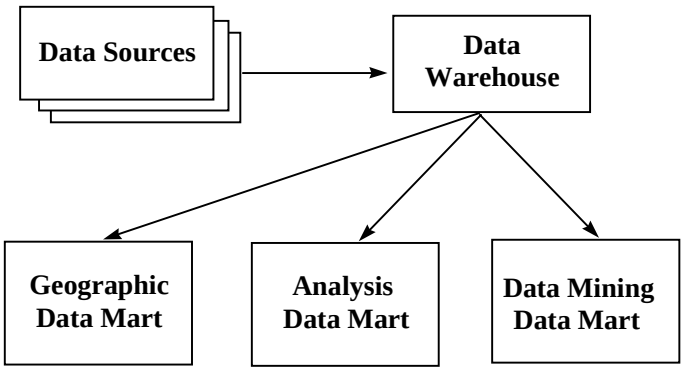

Figure 1. Data mining data mart extracted from a data warehouse.

A data warehouse is not a requirement for data mining. Setting up a large data warehouse that consolidates data from multiple sources, resolves data integrity problems, and loads the data into a query database can be an enormous task, sometimes taking years and costing millions of dollars. You could, however, mine data from one or more operational or transactional databases by simply extracting it into a read-only database (Figure 2). This new database functions as a type of data mart.

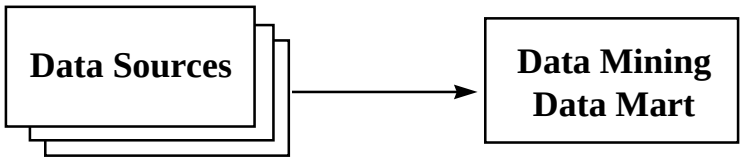

Figure 2. Data mining data mart extracted from operational databases.

Data mining and OLAP

One of the most common questions from data processing professionals is about the difference between data mining and OLAP (On-Line Analytical Processing). As we shall see, they are very different tools that can complement each other.

OLAP is part of the spectrum of decision support tools. Traditional query and report tools describe *what* is in a database. OLAP goes further; it's used to answer *why* certain things are true. The user forms a hypothesis about a relationship and verifies it with a series of queries against the data. For example, an analyst might want to determine the factors that lead to loan defaults. He or she might initially hypothesize that people with low incomes are bad credit risks and analyze the database with OLAP to verify (or disprove) this assumption. If that hypothesis were not borne out by the data, the analyst might then look at high debt as the determinant of risk. If the data did not support this guess either, he or she might then try debt and income together as the best predictor of bad credit risks.

In other words, the OLAP analyst generates a series of hypothetical patterns and relationships and uses queries against the database to verify them or disprove them. OLAP analysis is essentially a deductive process. But what happens when the number of variables being analyzed is in the dozens or even hundreds? It becomes much more difficult and time-consuming to find a good hypothesis (let alone be confident that there is not a better explanation than the one found), and analyze the database with OLAP to verify or disprove it.

Data mining is different from OLAP because rather than verify hypothetical patterns, it uses the data itself to uncover such patterns. It is essentially an inductive process. For example, suppose the analyst who wanted to identify the risk factors for loan default were to use a data mining tool. The data mining tool might discover that people with high debt and low incomes were bad credit risks (as above), but it might go further and also discover a pattern the analyst did not think to try, such as that age is also a determinant of risk.

Here is where data mining and OLAP can complement each other. Before acting on the pattern, the analyst needs to know what the financial implications would be of using the discovered pattern to govern who gets credit. The OLAP tool can allow the analyst to answer those kinds of questions.

Furthermore, OLAP is also complementary in the early stages of the knowledge discovery process because it can help you explore your data, for instance by focusing attention on important variables,

identifying exceptions, or finding interactions. This is important because the better you understand your data, the more effective the knowledge discovery process will be.

Data mining, machine learning and statistics

Data mining takes advantage of advances in the fields of artificial intelligence (AI) and statistics. Both disciplines have been working on problems of pattern recognition and classification. Both communities have made great contributions to the understanding and application of neural nets and decision trees.

Data mining does not replace traditional statistical techniques. Rather, it is an extension of statistical methods that is in part the result of a major change in the statistics community. The development of most statistical techniques was, until recently, based on elegant theory and analytical methods that worked quite well on the modest amounts of data being analyzed. The increased power of computers and their lower cost, coupled with the need to analyze enormous data sets with millions of rows, have allowed the development of new techniques based on a brute-force exploration of possible solutions.

New techniques include relatively recent algorithms like neural nets and decision trees, and new approaches to older algorithms such as discriminant analysis. By virtue of bringing to bear the increased computer power on the huge volumes of available data, these techniques can approximate almost any functional form or interaction on their own. Traditional statistical techniques rely on the modeler to specify the functional form and interactions.

The key point is that data mining is the application of these and other AI and statistical techniques to common business problems in a fashion that makes these techniques available to the skilled knowledge worker as well as the trained statistics professional. Data mining is a tool for increasing the productivity of people trying to build predictive models.

Data mining and hardware/software trends

A key enabler of data mining is the major progress in hardware price and performance. The dramatic 99% drop in the price of computer disk storage in just the last few years has radically changed the economics of collecting and storing massive amounts of data. At $10/megabyte, one terabyte of data costs $10,000,000 to store. At 10¢/megabyte, one terabyte of data costs only $100,000 to store! This doesn’t even include the savings in real estate from greater storage capacities.

The drop in the cost of computer processing has been equally dramatic. Each generation of chips greatly increases the power of the CPU, while allowing further drops on the cost curve. This is also reflected in the price of RAM (random access memory), where the cost of a megabyte has dropped from hundreds of dollars to around a dollar in just a few years. PCs routinely have 64 megabytes or more of RAM, and workstations may have 256 megabytes or more, while servers with gigabytes of main memory are not a rarity.

While the power of the individual CPU has greatly increased, the real advances in scalability stem from parallel computer architectures. Virtually all servers today support multiple CPUs using symmetric multi-processing, and clusters of these SMP servers can be created that allow hundreds of CPUs to work on finding patterns in the data.

Advances in database management systems to take advantage of this hardware parallelism also benefit data mining. If you have a large or complex data mining problem requiring a great deal of access to an existing database, native DBMS access provides the best possible performance.

The result of these trends is that many of the performance barriers to finding patterns in large amounts of data are being eliminated.

Data mining applications

Data mining is increasingly popular because of the substantial contribution it can make. It can be used to control costs as well as contribute to revenue increases.

Many organizations are using data mining to help manage all phases of the customer life cycle, including acquiring new customers, increasing revenue from existing customers, and retaining good customers. By determining characteristics of good customers (profiling), a company can target prospects with similar characteristics. By profiling customers who have bought a particular product it can focus attention on similar customers who have not bought that product (cross-selling). By profiling customers who have left, a company can act to retain customers who are at risk for leaving (reducing churn or attrition), because it is usually far less expensive to retain a customer than acquire a new one.

Data mining offers value across a broad spectrum of industries. Telecommunications and credit card companies are two of the leaders in applying data mining to detect fraudulent use of their services. Insurance companies and stock exchanges are also interested in applying this technology to reduce fraud. Medical applications are another fruitful area: data mining can be used to predict the effectiveness of surgical procedures, medical tests or medications. Companies active in the financial markets use data mining to determine market and industry characteristics as well as to predict individual company and stock performance. Retailers are making more use of data mining to decide which products to stock in particular stores (and even how to place them within a store), as well as to assess the effectiveness of promotions and coupons. Pharmaceutical firms are mining large databases of chemical compounds and of genetic material to discover substances that might be candidates for development as agents for the treatments of disease.

Successful data mining

There are two keys to success in data mining. First is coming up with a precise formulation of the problem you are trying to solve. A focused statement usually results in the best payoff. The second key is using the right data. After choosing from the data available to you, or perhaps buying external data, you may need to transform and combine it in significant ways.

The more the model builder can “play” with the data, build models, evaluate results, and work with the data some more (in a given unit of time), the better the resulting model will be. Consequently, the degree to which a data mining tool supports this interactive data exploration is more important than the algorithms it uses.

Ideally, the data exploration tools (graphics/visualization, query/OLAP) are well-integrated with the analytics or algorithms that build the models.

Summaries and visualization

Before you can build good predictive models, you must understand your data. Start by gathering a variety of numerical summaries (including descriptive statistics such as averages, standard deviations and so forth) and looking at the distribution of the data. You may want to produce cross tabulations (pivot tables) for multi-dimensional data.

Data can be *continuous*, having any numerical value (e.g., quantity sold) or *categorical*, fitting into discrete classes (e.g., red, blue, green). Categorical data can be further defined as either *ordinal*, having a meaningful order (e.g., high/medium/low), or *nominal*, that is unordered (e.g., postal codes).

Graphing and visualization tools are a vital aid in data preparation and their importance to effective data analysis cannot be overemphasized. Data visualization most often provides the *Aha!* leading to new insights and success. Some of the common and very useful graphical displays of data are histograms or box plots that display distributions of values. You may also want to look at scatter plots in two or three dimensions of different pairs of variables. The ability to add a third, overlay variable greatly increases the usefulness of some types of graphs.

Visualization works because it exploits the broader information bandwidth of graphics as opposed to text or numbers. It allows people to see the forest and zoom in on the trees. Patterns, relationships, exceptional values and missing values are often easier to perceive when shown graphically, rather than as lists of numbers and text.

The problem in using visualization stems from the fact that models have many dimensions or variables, but we are restricted to showing these dimensions on a two-dimensional computer screen or paper. For example, we may wish to view the relationship between credit risk and age, sex, marital status, own-or-rent, years in job, etc. Consequently, visualization tools must use clever representations to collapse *n* dimensions into two. Increasingly powerful and sophisticated data visualization tools are being developed, but they often require people to train their eyes through practice in order to understand the information being conveyed. Users who are color-blind or who are not spatially oriented may also have problems with visualization tools.

Clustering

Clustering divides a database into different groups. The goal of clustering is to find groups that are very different from each other, and whose members are very similar to each other. Unlike classification (see Predictive Data Mining, below), you don't know what the clusters will be when you start, or by which attributes the data will be clustered. Consequently, someone who is knowledgeable in the business must interpret the clusters. Often it is necessary to modify the clustering by excluding variables that have been employed to group instances, because upon examination the user identifies them as irrelevant or not meaningful. After you have found clusters that reasonably segment your database, these clusters may then be used to classify new data. Some of the common algorithms used to perform clustering include Kohonen feature maps and K-means.

Don't confuse clustering with segmentation. Segmentation refers to the general problem of identifying groups that have common characteristics. Clustering is a way to segment data into groups that are not previously defined, whereas classification is a way to segment data by assigning it to groups that are already defined.

Link analysis

Link analysis is a descriptive approach to exploring data that can help identify relationships among values in a database. The two most common approaches to link analysis are *association discovery* and *sequence discovery.* Association discovery finds rules about items that appear together in an event such as a purchase transaction. Market-basket analysis is a well-known example of association discovery. Sequence discovery is very similar, in that a sequence is an association related over time.

Associations are written as A ⇒ B, where A is called the antecedent or left-hand side (LHS), and B is called the consequent or right-hand side (RHS). For example, in the association rule "If people buy a hammer then they buy nails," the antecedent is "buy a hammer" and the consequent is "buy nails."

It's easy to determine the proportion of transactions that contain a particular item or item set: simply count them. The frequency with which a particular association (e.g., the item set "hammers and nails") appears in the database is called its *support* or *prevalence.* If, say, 15 transactions out of 1,000 consist of "hammer and nails," the support for this association would be 1.5%. A low level of support (say, one transaction out of a million) may indicate that the particular association isn't very important — or it may indicated the presence of bad data (e.g., "male and pregnant").

To discover meaningful rules, however, we must also look at the *relative* frequency of occurrence of the items and their combinations. Given the occurrence of item A (the antecedent), how often does item B (the consequent) occur? That is, what is the conditional predictability of B, given A? Using the above example, this would mean asking "When people buy a hammer, how often do they also buy nails?" Another term for this conditional predictability is *confidence.* Confidence is calculated as a ratio: (frequency of A and B)/(frequency of A).

Let's specify our hypothetical database in more detail to illustrate these concepts:

Total hardware-store transactions: 1,000
Number which include "hammer": 50
Number which include "nails": 80
Number which include "lumber": 20
Number which include "hammer" *and* "nails": 15
Number which include "nails" *and* "lumber": 10
Number which include "hammer" *and* "lumber": 10
Number which include "hammer," "nails" *and* "lumber": 5

We can now calculate:

Support for "hammer and nails" = 1.5% (15/1,000)
Support for "hammer, nails and lumber" = 0.5% (5/1,000)
Confidence of "hammer ⇒ nails" = 30% (15/50)
Confidence of "nails ⇒ hammer" = 19% (15/80)
Confidence of "hammer and nails ⇒ lumber" = 33% (5/15)
Confidence of "lumber ⇒ hammer and nails" = 25% (5/20)

Thus we can see that the likelihood that a hammer buyer will also purchase nails (30%) is greater than the likelihood that someone buying nails will also purchase a hammer (19%). The prevalence of this hammer-and-nails association (i.e., the support is 1.5%) is high enough to suggest a meaningful rule.

Lift is another measure of the power of an association. The greater the lift, the greater the influence that the occurrence of A has on the likelihood that B will occur. Lift is calculated as the ratio (confidence of A ⇒ B)/ (frequency of B). From our example:

Lift of "hammer ⇒ nails": 3.75 (30%/8%)
Lift of "hammer and nails ⇒ lumber": 16.5 (33%/2%)

Association algorithms find these rules by doing the equivalent of sorting the data while counting occurrences so that they can calculate confidence and support. The efficiency with which they can do this is one of the differentiators among algorithms. This is especially important because of the combinatorial explosion that results in enormous numbers of rules, even for market baskets in the express lane. Some algorithms will create a database of rules, confidence factors, and support that can be queried (for example, "Show me all associations in which ice cream is the consequent, that have a confidence factor of over 80% and a support of 2% or more").

Another common attribute of association rule generators is the ability to specify an item hierarchy. In our example we have looked at all nails and hammers, not individual types. It is important to choose a proper level of aggregation or you'll be unlikely to find associations of interest. An item hierarchy allows you to control the level of aggregation and experiment with different levels.

Remember that association or sequence rules are not really rules, but rather descriptions of relationships in a particular database. There is no formal testing of models on other data to increase the predictive power of these rules. Rather there is an implicit assumption that the past behavior will continue in the future.

It is often difficult to decide what to do with association rules you've discovered. In store planning, for example, putting associated items physically close together may reduce the total value of market baskets — customers may buy less overall because they no longer pick up unplanned items while walking through the store in search of the desired items. Insight, analysis and experimentation are usually required to achieve any benefit from association rules.

Graphical methods may also be very useful in seeing the structure of links. In Figure 3 each of the circles represents a value or an event. The lines connecting them show a link. The thicker lines represent stronger or more frequent linkages, thus emphasizing potentially more important relationships such as associations. For instance, looking at an insurance database to detect potential fraud might reveal that a particular doctor and lawyer work together on an unusually large number of cases.

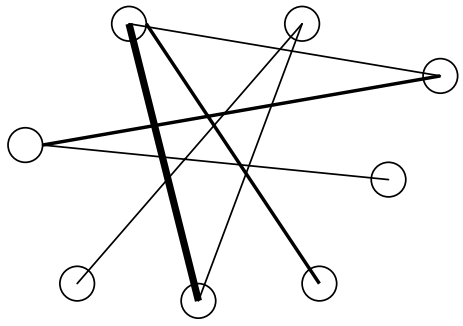

Figure 3. Linkage diagram

A hierarchy of choices

The goal of data mining is to produce new knowledge that the user can act upon. It does this by building a model of the real world based on data collected from a variety of sources which may include corporate transactions, customer histories and demographic information, process control data, and relevant external databases such as credit bureau information or weather data. The result of the model building is a description of patterns and relationships in the data that can be confidently used for prediction.

To avoid confusing the different aspects of data mining, it helps to envision a hierarchy of the choices and decisions you need to make before you start:

- Business goal
- Type of prediction
- Model type
- Algorithm
- Product

At the highest level is the **business goal:** what is the ultimate purpose of mining this data? For example, seeking patterns in your data to help you retain good customers, you might build one model to predict customer profitability and a second model to identify customers likely to leave (attrition). Your knowledge of your organization's needs and objectives will guide you in formulating the goal of your models.

The next step is deciding on the **type of prediction** that's most appropriate: (1) *classification*: predicting into what category or class a case falls, or (2) *regression*: predicting what number value a variable will have (if it's a variable that varies with time, it's called *time series* prediction). In the example above, you might use regression to forecast the amount of profitability, and classification to predict which customers might leave. These are discussed in more detail below.

Now you can choose the **model type:** a neural net to perform the regression, perhaps, and a decision tree for the classification. There are also traditional statistical models to choose from such as logistic regression, discriminant analysis, or general linear models. The most important model types for data mining are described in the next section, on DATA MINING MODELS AND ALGORITHMS.

Many **algorithms** are available to build your models. You might build the neural net using backpropagation or radial basis functions. For the decision tree, you might choose among CART, C5.0, Quest, or CHAID. Some of these algorithms are also discussed in DATA MINING MODELS AND ALGORITHMS, below.

When selecting a data mining **product,** be aware that they generally have different implementations of a particular algorithm even when they identify it with the same name. These implementation differences can affect operational characteristics such as memory usage and data storage, as well as performance characteristics such as speed and accuracy. Other key considerations to keep in mind are covered later in the section on SELECTING DATA MINING PRODUCTS.

Many business goals are best met by building multiple model types using a variety of algorithms. You may not be able to determine which model type is best until you've tried several approaches.

Some terminology

In predictive models, the values or classes we are predicting are called the *response, dependent* or *target variables.* The values used to make the prediction are called the *predictor* or *independent variables*.

Predictive models are built, or *trained,* using data for which the value of the response variable is already known. This kind of training is sometimes referred to as *supervised learning,* because calculated or estimated values are compared with the known results. (By contrast, descriptive techniques such as clustering, described in the previous section, are sometimes referred to as *unsupervised learning* because there is no already-known result to guide the algorithms.)

Classification

Classification problems aim to identify the characteristics that indicate the group to which each case belongs. This pattern can be used both to understand the existing data and to predict how new instances will behave. For example, you may want to predict whether individuals can be classified as likely to respond to a direct mail solicitation, vulnerable to switching over to a competing long-distance phone service, or a good candidate for a surgical procedure.

Data mining creates classification models by examining already classified data (cases) and inductively finding a predictive pattern. These existing cases may come from an historical database, such as people who have already undergone a particular medical treatment or moved to a new long-distance service. They may come from an experiment in which a sample of the entire database is tested in the real world and the results used to create a classifier. For example, a sample of a mailing list would be sent an offer, and the results of the mailing used to develop a classification model to be applied to the entire database. Sometimes an expert classifies a sample of the database, and this classification is then used to create the model which will be applied to the entire database.

Regression

Regression uses existing values to forecast what other values will be. In the simplest case, regression uses standard statistical techniques such as linear regression. Unfortunately, many real-world problems are not simply linear projections of previous values. For instance, sales volumes, stock prices, and product failure rates are all very difficult to predict because they may depend on complex interactions of multiple predictor variables. Therefore, more complex techniques (e.g., logistic regression, decision trees, or neural nets) may be necessary to forecast future values.

The same model types can often be used for both regression and classification. For example, the CART (Classification And Regression Trees) decision tree algorithm can be used to build both classification trees (to classify categorical response variables) and regression trees (to forecast continuous response variables). Neural nets too can create both classification and regression models.

Time series

Time series forecasting predicts unknownfuture values based on a time-varying series of predictors. Like regression, it uses known results to guide its predictions. Models must take into account the distinctive properties of time, especially the hierarchy of periods (including such varied definitions as the five- or seven-day work week, the thirteen-“month” year, etc.), seasonality, calendar effects such as holidays, date arithmetic, and special considerations such as how much of the past is relevant.

Now let's examine some of the types of models and algorithms used to mine data. Most products use variations of algorithms that have been published in computer science or statistics journals, with their specific implementations customized to meet the individual vendor's goal. For example, many vendors sell versions of the CART or CHAID decision trees with enhancements to work on parallel computers. Some vendors have proprietary algorithms which, while not extensions or enhancements of any published approach, may work quite well.

Most of the models and algorithms discussed in this section can be thought of as generalizations of the standard workhorse of modeling, the linear regression model. Much effort has been expended in the statistics, computer science, artificial intelligence and engineering communities to overcome the limitations of this basic model. The common characteristic of many of the newer technologies we will consider is that the pattern-finding mechanism is data-driven rather than user-driven. That is, the relationships are found inductively by the software itself based on the existing data rather than requiring the modeler to specify the functional form and interactions.

Perhaps the most important thing to remember is that no one model or algorithm can or should be used exclusively. For any given problem, the nature of the data itself will affect the choice of models and algorithms you choose. There is no "best" model or algorithm. Consequently, you will need a variety of tools and technologies in order to find the best possible model.

Neural networks

Neural networks are of particular interest because they offer a means of efficiently modeling large and complex problems in which there may be hundreds of predictor variables that have many interactions.(Actual biological neural networks are incomparably more complex.) Neural nets may be used in classification problems (where the output is a categorical variable) or for regressions (where the output variable is continuous).

A neural network (Figure 4) starts with an *input layer*, where each node corresponds to a predictor variable. These input nodes are connected to a number of nodes in a *hidden layer*. Each input node is connected to every node in the hidden layer. The nodes in the hidden layer may be connected to nodes in another hidden layer, or to an *output layer*. The output layer consists of one or more response variables.

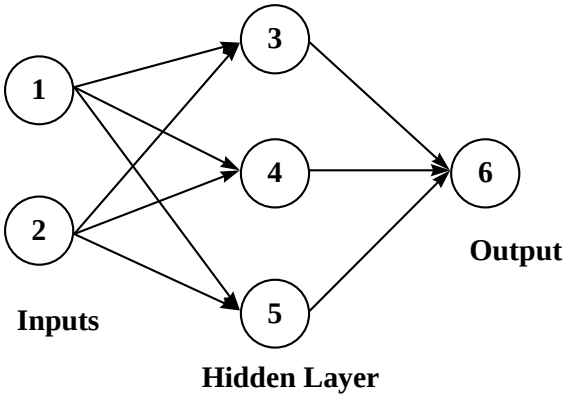

Figure 4. A neural network with one hidden layer.

After the input layer, each node takes in a set of inputs, multiplies them by a connection weight W_{xy} (e.g., the weight from node 1 to 3 is W_{13} — see Figure 5), adds them together, applies a function (called the activation or squashing function) to them, and passes the output to the node(s) in the next layer. For example, the value passed from node 4 to node 6 is:

Activation function applied to ([W_{14} * value of node 1] + [W_{24} * value of node 2])

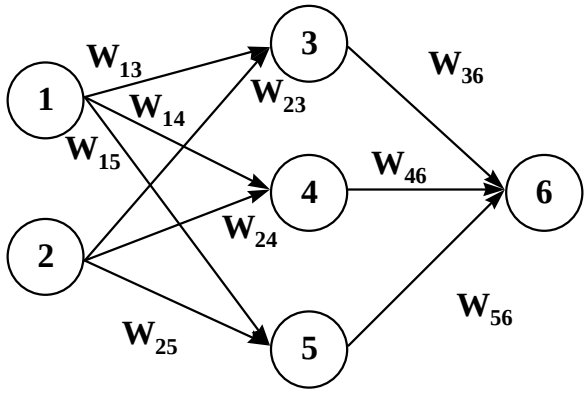

Figure 5. W_{xy} is the weight from node x to node y.

Each node may be viewed as a predictor variable (nodes 1 and 2 in this example) or as a combination of predictor variables (nodes 3 through 6). Node 6 is a non-linear combination of the values of nodes 1 and 2, because of the activation function on the summed values at the hidden nodes. In fact, if there is a linear activation function but no hidden layer, neural nets are equivalent to a linear regression; and with certain non-linear activation functions, neural nets are equivalent to logistic regression.

The connection weights (W's) are the unknown parameters which are estimated by a training method. Originally, the most common training method was backpropagation; newer methods include conjugate gradient, quasi-Newton, Levenberg-Marquardt, and genetic algorithms. Each training method has a set of parameters that control various aspects of training such as avoiding local optima or adjusting the speed of conversion.

The *architecture* (or topology) of a neural network is the number of nodes and hidden layers, and how they are connected. In designing a neural network, either the user or the software must choose the number of hidden nodes and hidden layers, the activation function, and limits on the weights. While there are some general guidelines, you may have to experiment with these parameters.

One of the most common types of neural network is the *feed-forward backpropagation* network. For simplicity of discussion, we will assume a single hidden layer.

Backpropagation training is simply a version of gradient descent, a type of algorithm that tries to reduce a target value (error, in the case of neural nets) at each step. The algorithm proceeds as follows.

Feed forward: The value of the output node is calculated based on the input node values and a set of initial weights. The values from the input nodes are combined in the hidden layers, and the values of those nodes are combined to calculate the output value.

Backpropagation: The error in the output is computed by finding the difference between the calculated output and the desired output (i.e., the actual values found in the training set). Next, the error from the output is assigned to the hidden layer nodes proportionally to their weights. This permits an error to be computed for every output node and hidden node in the network. Finally, the error at each of the hidden and output nodes is used by the algorithm to adjust the weight coming into that node to reduce the error.

This process is repeated for each row in the training set. Each pass through all rows in the training set is called an *epoch*. The training set will be used repeatedly, until the error no longer decreases. At that point the neural net is considered to be trained to find the pattern in the test set.

Because so many parameters may exist in the hidden layers, a neural net with enough hidden nodes will always eventually fit the training set if left to run long enough. But how well it will do on other data? To avoid an overfitted neural network which will only work well on the training data, you must know when to stop training. Some implementations will evaluate the neural net against the test data periodically during training. As long as the error rate on the test set is decreasing, training will continue. If the error rate on the test data goes up, even though the error rate on the training data is still decreasing, then the neural net may be overfitting the data.

The graph in Figure 6 illustrates how the test data set helps us avoid overfitting. You can see how the error rate decreases with each pass the neural net makes through the data (small circle markers), but the error rate for the test data (triangle markers) bottoms out and starts increasing. Since the goal of data mining is to make predictions on data other than the training set, you are clearly better off using a neural net that minimizes the error on the test data, not the training data.

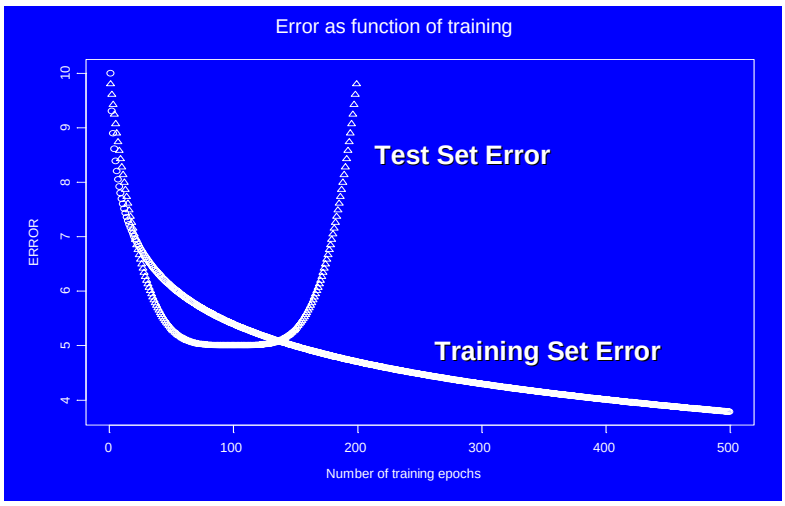

Figure 6. Error rate as a function of the number of epochs in a neural net.
(Screen shot courtesy Dr. Richard D. De Veaux, Williams College)

Neural networks differ in philosophy from many statistical methods in several ways. First, a neural network usually has more parameters than does a typical statistical model. For example, there are thirteen parameters (nine weights and four bias or constant terms) in the neural network shown in Figure 4. Because they are so numerous, and because so many combinations of parameters result in similar predictions, the parameters become uninterpretable and the network serves as a "black box" predictor. In fact, a given result can be associated with several different sets of weights. Consequently, the network weights in general do not aid in understanding the underlying process generating the prediction. However, this is acceptable in many applications. A bank may want to automatically recognize handwritten applications, but does not care about the form of the functional relationship between the pixels and the characters they represent. Some of the many applications where hundreds of variables may be input into models with thousands of parameters (node weights) include modeling of chemical plants, robots and financial markets, and pattern recognition problems such as speech, vision and handwritten character recognition.

One advantage of neural network models is that they can easily be implemented to run on massively parallel computers with each node simultaneously doing its own calculations.

Users must be conscious of several facts about neural networks: First, neural networks are not easily interpreted. There is no explicit rationale given for the decisions or predictions a neural network makes.

Second, they tend to overfit the training data unless very stringent measures, such as weight decay and/or cross validation, are used judiciously. This is due to the very large number of parameters of the neural network which, if allowed to be of sufficient size, will fit *any* data set arbitrarily well when allowed to train to convergence.

Third, neural networks require an extensive amount of training time unless the problem is very small. Once trained, however, they can provide predictions very quickly.

Fourth, they require no less data preparation than any other method, which is to say they require a lot of data preparation. One myth of neural networks is that data of any quality can be used to provide reasonable predictions. The most successful implementations of neural networks (or decision trees, or logistic regression, or any other method) involve very careful data cleansing, selection, preparation and pre-processing. For instance, neural nets require that all variables be numeric. Therefore categorical data such as "state" is usually broken up into multiple dichotomous variables (e.g., "California," "New York") , each with a "1" (yes) or "0" (no) value. The resulting increase in variables is called the *categorical explosion.*

Finally, neural networks tend to work best when the data set is sufficiently large and the signal-to-noise ratio is reasonably high. Because they are so flexible, they will find many false patterns in a low signal-to-noise ratio situation.

Decision trees

Decision trees are a way of representing a series of rules that lead to a class or value. For example, you may wish to classify loan applicants as good or bad credit risks. Figure 7 shows a simple decision tree that solves this problem while illustrating all the basic components of a decision tree: the decision node, branches and leaves.

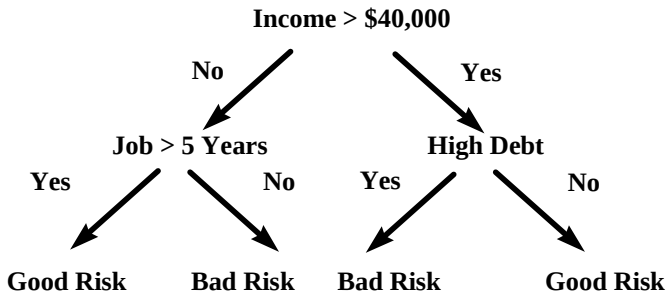

Figure 7. A simple classification tree.

The first component is the top decision node, or root node, which specifies a test to be carried out. The root node in this example is "Income > $40,000." The results of this test cause the tree to split into branches, each representing one of the possible answers. In this case, the test "Income > $40,000" can be answered either "yes" or "no," and so we get two branches.

Depending on the algorithm, each node may have two or more branches. For example, CART generates trees with only two branches at each node. Such a tree is called a binary tree. When more than two branches are allowed it is called a multiway tree.

Each branch will lead either to another decision node or to the bottom of the tree, called a leaf node. By navigating the decision tree you can assign a value or class to a case by deciding which branch to take, starting at the root node and moving to each subsequent node until a leaf node is reached. Each node uses the data from the case to choose the appropriate branch.

Armed with this sample tree and a loan application, a loan officer could determine whether the applicant was a good or bad credit risk. An individual with "Income > $40,000" and "High Debt" would be classified a "Bad Risk," whereas an individual with "Income < $40,000" and "Job > 5 Years" would be classified a "Good Risk."

Decision trees models are commonly used in data mining to examine the data and induce the tree and its rules that will be used to make predictions. A number of different algorithms may be used for building decision trees including CHAID (Chi-squared Automatic Interaction Detection), CART (Classification And Regression Trees), Quest, and C5.0.

Decision trees are grown through an iterative splitting of data into discrete groups, where the goal is to maximize the "distance" between groups at each split.

One of the distinctions between decision tree methods is how they measure this distance. While the details of such measurement are beyond the scope of this introduction, you can think of each split as separating the data into new groups which are as different from each other as possible. This is also sometimes called making the groups purer. Using our simple example where the data had two possible output classes — Good Risk and Bad Risk — it would be preferable if each data split found a criterion resulting in "pure" groups with instances of only one class instead of both classes.

Decision trees which are used to predict categorical variables are called *classification trees* because they place instances in categories or classes. Decision trees used to predict continuous variables are called *regression trees.*

The example we've been using up until now has been very simple. The tree is easy to understand and interpret. However, trees can become very complicated. Imagine the complexity of a decision tree derived from a database of hundreds of attributes and a response variable with a dozen output classes. Such a tree would be extremely difficult to understand, although each path to a leaf is usually understandable. In that sense a decision tree can explain its predictions, which is an important advantage.

However, this clarity can be somewhat misleading. For example, the hard splits of decision trees imply a precision that is rarely reflected in reality. (Why would someone whose salary was $40,001 be a good credit risk whereas someone whose salary was $40,000 not be?) Furthermore, since several trees can often represent the same data with equal accuracy, what interpretation should be placed on the rules?

Decision trees make few passes through the data (no more than one pass for each level of the tree) and they work well with many predictor variables. As a consequence, models can be built very quickly, making them suitable for large data sets.

Trees left to grow without bound take longer to build and become unintelligible, but more importantly they overfit the data. Tree size can be controlled via *stopping rules* that limit growth. One common stopping rule is simply to limit the maximum depth to which a tree may grow. Another stopping rule is to establish a lower limit on the number of records in a node and not do splits below this limit.

An alternative to stopping rules is to prune the tree. The tree is allowed to grow to its full size and then, using either built-in heuristics or user intervention, the tree is pruned back to the smallest size that does not compromise accuracy. For example, a branch or subtree that the user feels is inconsequential because it has very few cases might be removed. CART prunes trees by cross validating them to see if the improvement in accuracy justifies the extra nodes.

A common criticism of decision trees is that they choose a split using a "greedy" algorithm in which the decision on which variable to split doesn't take into account any effect the split might have on future splits. In other words, the split decision is made at the node "in the moment" and it is never revisited. In addition, all splits are made sequentially, so each split is dependent on its predecessor. Thus all future splits are dependent on the first split, which means the final solution could be very different if a different first split is made. The benefit of looking ahead to make the best splits based on two or more levels at one time is unclear. Such attempts to look ahead are in the research stage, but are very computationally intensive and presently unavailable in commercial implementations.

Furthermore, algorithms used for splitting are generally univariate; that is, they consider only one predictor variable at a time. And while this approach is one of the reasons the model builds quickly — it limits the number of possible splitting rules to test — it also makes relationships between predictor variables harder to detect.

Decision trees that are not limited to univariate splits could use multiple predictor variables in a single splitting rule. Such a decision tree could allow linear combinations of variables, also known as oblique trees. A criterion for a split might be "SALARY < (0.35 * MORTGAGE)," for instance. Splitting on logical combinations of variables (such as "SALARY > 35,000 OR MORTGAGE < 150,000") is another kind of multivariate split.

Decision trees handle non-numeric data very well. This ability to accept categorical data minimizes the amount of data transformations and the explosion of predictor variables inherent in neural nets.

Some classification trees were designed for and therefore work best when the predictor variables are also categorical. Continuous predictors can frequently be used even in these cases by converting the continuous variable to a set of ranges (binning). Some decision trees do not support continuous response variables (i.e., will not build regression trees), in which case the response variables in the training set must also be binned to output classes.

Multivariate Adaptive Regression Splines (MARS)

In the mid-1980s one of the inventors of CART, Jerome H. Friedman, developed a method designed to address its shortcomings.

The main disadvantages he wanted to eliminate were:

- Discontinuous predictions (hard splits).
- Dependence of all splits on previous ones.
- Reduced interpretability due to interactions, especially high-order interactions.

To this end he developed the MARS algorithm. The basic idea of MARS is quite simple, while the algorithm itself is rather involved. Very briefly, the CART disadvantages are taken care of by:

- Replacing the discontinuous branching at a node with a continuous transition modeled by a pair of straight lines. At the end of the model-building process, the straight lines at each node are replaced with a very smooth function called a spline.
- Not requiring that new splits be dependent on previous splits.

Unfortunately, this means MARS loses the tree structure of CART and cannot produce rules. On the other hand, MARS automatically finds and lists the most important predictor variables as well as the interactions among predictor variables. MARS also plots the dependence of the response on each predictor. The result is an automatic non-linear step-wise regression tool.

MARS, like most neural net and decision tree algorithms, has a tendency to overfit the training data. This can be addressed in two ways. First, manual cross validation can be performed and the algorithm tuned to provide good prediction on the test set. Second, there are various tuning parameters in the algorithm itself that can guide internal cross validation.

Rule induction

Rule induction is a method for deriving a set of rules to classify cases. Although decision trees can produce a set of rules, rule induction methods generate a set of independent rules which do not necessarily (and are unlikely to) form a tree. Because the rule inducer is not forcing splits at each level, and can look ahead, it may be able to find different and sometimes better patterns for classification. Unlike trees, the rules generated may not cover all possible situations. Also unlike trees, rules may sometimes conflict in their predictions, in which case it is necessary to choose which rule to follow. One common method to resolve conflicts is to assign a confidence to rules and use the one in which you are most confident. Alternatively, if more than two rules conflict, you may let them vote, perhaps weighting their votes by the confidence you have in each rule.

K-nearest neighbor and memory-based reasoning (MBR)

When trying to solve new problems, people often look at solutions to similar problems that they have previously solved. K-nearest neighbor (k-NN) is a classification technique that uses a version of this same method. It decides in which class to place a new case by examining some number — the "k" in k-nearest neighbor — of the most similar cases or neighbors (Figure 8). It counts the number of cases for each class, and assigns the new case to the same class to which most of its neighbors belong.

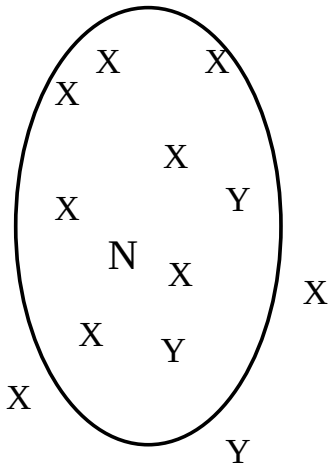

Figure 8. K-nearest neighbor. N is a new case. It would be assigned to the class X because the seven X's within the ellipse outnumber the two Y's.

The first thing you must do to apply k-NN is to find a measure of the distance between attributes in the data and then calculate it. While this is easy for numeric data, categorical variables need special handling. For example, what is the distance between blue and green? You must then have a way of summing the distance measures for the attributes. Once you can calculate the distance between cases, you then select the set of already classified cases to use as the basis for classifying new cases, decide how large a neighborhood in which to do the comparisons, and also decide how to count the neighbors themselves (e.g., you might give more weight to nearer neighbors than farther neighbors).

K-NN puts a large computational load on the computer because the calculation time increases as the factorial of the total number of points. While it's a rapid process to apply a decision tree or neural net to a new case, k-NN requires that a new calculation be made for each new case. To speed up k-NN, frequently all the data is kept in memory. Memory-based reasoning usually refers to a k-NN classifier kept in memory.

K-NN models are very easy to understand when there are few predictor variables. They are also useful for building models that involve non-standard data types, such as text. The only requirement for being able to include a data type is the existence of an appropriate metric.

Logistic regression

Logistic regression is a generalization of linear regression. It is used primarily for predicting binary variables (with values such as yes/no or 0/1) and occasionally multi-class variables. Because the response variable is discrete, it cannot be modeled directly by linear regression. Therefore, rather than predict whether the event itself (the response variable) will occur, we build the model to predict the *logarithm* of the odds of its occurrence. This logarithm is called the log odds or the logit transformation.

The odds ratio:

$$\frac{\text{probability of an event occurring}}{\text{probability of the event not occurring}}$$

has the same interpretation as in the more casual use of odds in games of chance or sporting events. When we say that the odds are 3 to 1 that a particular team will win a soccer game, we mean that the probability of their winning is three times as great as the probability of their losing. So we believe they have a 75% chance of winning and a 25% chance of losing. Similar terminology can be applied to the chances of a particular type of customer (e.g., a customer with a given gender, income, marital status, etc.) replying to a mailing. If we say the odds are 3 to 1 that the customer will respond, we mean that the probability of that type of customer responding is three times as great as the probability of him or her not responding.

Having predicted the log odds, you then take the anti-log of this number to find the odds. Odds of 62% would mean that the case is assigned to the class designated "1" or "yes," for example.

While logistic regression is a very powerful modeling tool, it assumes that the response variable (the log odds, not the event itself) is linear in the coefficients of the predictor variables. Furthermore, the modeler, based on his or her experience with the data and data analysis, must choose the right inputs and specify their functional relationship to the response variable. So, for example, the modeler must choose among income or $(\text{income})^2$ or log (income) as a predictor variable. Additionally the modeler must explicitly add terms for any interactions. It is up to the model builder to search for the right variables, find their correct expression, and account for their possible interactions. Doing this effectively requires a great deal of skill and experience on the part of the analyst.

Neural nets, on the other hand, use their hidden layers to estimate the forms of the non-linear terms and interaction in a semi-automated way. Users need a different set of analytic skills to apply neural nets successfully. For example, the choice of an activation function will affect the speed with which a neural net trains.

Discriminant analysis

Discriminant analysis is the oldest mathematical classification technique, having been first published by R. A. Fisher in 1936 to classify the famous Iris botanical data into three species. It finds hyper-planes (e.g., lines in two dimensions, planes in three, etc.) that separate the classes. The resultant model is very easy to interpret because all the user has to do is determine on which side of the line (or hyper-plane) a point falls. Training is simple and scalable. The technique is very sensitive to patterns in the data. It is used very often in certain disciplines such as medicine, the social sciences, and field biology.

Discriminant analysis is not popular in data mining, however, for three main reasons. First, it assumes that all of the predictor variables are normally distributed (i.e., their histograms look like bell-shaped curves), which may not be the case. Second, unordered categorical predictor variables (e.g., red/blue/green) cannot be used at all. Third, the boundaries that separate the classes are all linear forms (such as lines or planes), but sometimes the data just can't be separated that way.

Recent versions of discriminant analysis address some of these problems by allowing the boundaries to be quadratic as well as linear, which significantly increases the sensitivity in certain cases. There are also techniques that allow the normality assumption to be replaced with an estimate of the real distribution (i.e., replace the theoretical bell-shaped curve with the histograms of the predictor variables). Ordered categorical data can be modeled by forming the histogram from the bins defined by the categorical variables.

Generalized Additive Models (GAM)

There is a class of models extending both linear and logistic regression, known as generalized additive models or GAM. They are called additive because we assume that the model can be written as the sum of possibly non-linear functions, one for each predictor. GAM can be used either for regression or for classification of a binary response. The response variable can be virtually any function of the predictors as long as there are not discontinuous steps. For example, suppose that payment delinquency is a rather complicated function of income where the probability of delinquency initially declines as income increases. It then turns around and starts to increase again for moderate income, finally peaking before coming down again for higher income card-holders. In such a case, a linear model may fail to see any relationship between income and delinquency due to the non-linear behavior. GAM, using computer power in place of theory or knowledge of the functional form, will produce a smooth curve, summarizing the relationship as described above. The most common estimation procedure is backfitting. Instead of estimating large numbers of parameters as neural nets do, GAM goes one step further and estimates a value of the output for each value of the input — one point, one estimate. As with the neural net, GAM generates a curve automatically, choosing the amount of complexity based on the data.

Boosting

If you were to build a model using one sample of data, and then build a new model using the same algorithm but on a different sample, you might get a different result. After validating the two models, you could choose the one that best met your objectives. Even better results might be achieved if you built several models and let them vote, making a prediction based on what the majority recommended. Of course, any interpretability of the prediction would be lost, but the improved results might be worth it.

This is exactly the approach taken by boosting, a technique first published by Freund and Schapire in 1996. Basically, boosting takes multiple random samples from the data and builds a classification model for each. The training set is changed based on the result of the previous models. The final classification is the class assigned most often by the models. The exact algorithms for boosting have evolved from the original, but the underlying idea is the same.

Boosting has become a very popular addition to data mining packages.

Genetic algorithms

Genetic algorithms are not used to find patterns per se, but rather to guide the learning process of data mining algorithms such as neural nets. Essentially, genetic algorithms act as a method for performing a guided search for good models in the solution space.

They are called genetic algorithms because they loosely follow the pattern of biological evolution in which the members of one generation (of models) compete to pass on their characteristics to the next generation (of models), until the best (model) is found. The information to be passed on is contained in "chromosomes," which contain the parameters for building the model.

For example, in building a neural net, genetic algorithms can replace backpropagation as a way to adjust the weights. The chromosome in this case would contain the weights. Alternatively, genetic algorithms might be used to find the best architecture, and the chromosomes would contain the number of hidden layers and the number of nodes in each layer.

While genetic algorithms are an interesting approach to optimizing models, they add a lot of computational overhead.

Process Models

Recognizing that a systematic approach is essential to successful data mining, many vendor and consulting organizations have specified a process model designed to guide the user (especially someone new to building predictive models) through a sequence of steps that will lead to good results. SPSS uses the 5A's — Assess, Access, Analyze, Act and Automate — and SAS uses SEMMA — Sample, Explore, Modify, Model, Assess.

Recently, a consortium of vendors and users consisting of NCR Systems Engineering Copenhagen (Denmark), Daimler-Benz AG (Germany), SPSS/Integral Solutions Ltd. (England) and OHRA Verzekeringen en Bank Groep B.V (The Netherlands) has been developing a specification called CRISP-DM — Cross-Industry Standard Process for Data Mining. CRISP-DM is similar to process models from other companies including the one from Two Crows Corporation. As of September 1999, CRISP-DM is a work in progress. It is a good start in helping people to understand the necessary steps in successful data mining.

The Two Crows Process Model

The Two Crows data mining process model described below is derived from the Two Crows process model discussed in the previous edition of this document, and also takes advantage of some insights from CRISP-DM.

Keep in mind that while the steps appear in a list, the data mining process is not linear — you will inevitably need to loop back to previous steps. For example, what you learn in the "explore data" step may require you to add new data to the data mining database. The initial models you build may provide insights that lead you to create new variables.

The basic steps of data mining for knowledge discovery are:

1. Define business problem
2. Build data mining database
3. Explore data
4. Prepare data for modeling
5. Build model
6. Evaluate model
7. Deploy model and results

Let's go through these steps to better understand the knowledge discovery process.

1. **Define the business problem.** First and foremost, the prerequisite to knowledge discovery is understanding your data and your business. Without this understanding, no algorithm, regardless of sophistication, is going to provide you with a result in which you should have confidence. Without this background you will not be able to identify the problems you're trying to solve, prepare the data for mining, or correctly interpret the results. To make the best use of data mining you must make a clear statement of your objectives. It may be that you wish to increase the response to a direct mail campaign. Depending on your specific goal, such as "increasing the

response rate" or "increasing the value of a response," you will build a very different model. An effective statement of the problem will include a way of measuring the results of your knowledge discovery project. It may also include a cost justification.

2. **Build a data mining database.** This step along with the next two constitute the core of the data preparation. Together, they take more time and effort than all the other steps combined. There may be repeated iterations of the data preparation and model building steps as you learn something from the model that suggests you modify the data. *These data preparation steps may take anywhere from 50% to 90% of the time and effort of the entire knowledge discovery process!*

   The data to be mined should be collected in a database. Note that this does not necessarily imply a database management system must be used. Depending on the amount of the data, the complexity of the data, and the uses to which it is to be put, a flat file or even a spreadsheet may be adequate.

   In general, it's not a good idea to use your corporate data warehouse for this. You will be better off creating a separate data mart. Mining the data will make you a very active user of the data warehouse, possibly causing resource allocation problems. You will often be joining many tables together and accessing substantial portions of the warehouse. A single trial model may require many passes through much of the warehouse.

   Almost certainly you will be modifying the data from the data warehouse. In addition you may want to bring in data from outside your company to overlay on the data warehouse data or you may want to add new fields computed from existing fields. You may need to gather additional data through surveys. Other people building different models from the data warehouse (some of whom will use the same data as you) may want to make similar alterations to the warehouse. However, data warehouse administrators do not look kindly on having data changed in what is unquestionably a corporate resource.

   One more reason for a separate database is that the structure of the corporate data warehouse may not easily support the kinds of exploration you need to do to understand this data. This includes queries summarizing the data, multi-dimensional reports (sometimes called pivot tables), and many different kinds of graphs or visualizations.

   Lastly, you may want to store this data in a different DBMS with a different physical design than the one you use for your corporate data warehouse. Increasingly, people are selecting special-purpose DBMSs which support these data mining requirements quite well. If, however, your corporate data warehouse allows you to create logical data marts and if it can handle the resource demands of data mining, then it may also serve as a good data mining database.

   The tasks in building a data mining database are:

   a. Data collection
   b. Data description
   c. Selection
   d. Data quality assessment and data cleansing
   e. Consolidation and integration
   f. Metadata construction
   g. Load the data mining database
   h. Maintain the data mining database

You must remember that these tasks are not performed in strict sequence, but as the need arises. For example, you will start constructing the metadata infrastructure as you collect the data, and modify it continuously. What you learn in consolidation or data quality assessment may change your initial selection decision.

a. Data collection. Identify the sources of the data you will be mining. A data-gathering phase may be necessary because some of the data you need may never have been collected. You may need to acquire external data from public databases (such as census or weather data) or proprietary databases (such as credit bureau data).

A Data Collection Report lists the properties of the different source data sets. Some of the elements in this report should include:

- Source of data (internal application or outside vendor)
- Owner
- Person/organization responsible for maintaining data
- DBA
- Cost (if purchased)
- Storage organization (e.g., Oracle database, VSAM file, etc.)
- Size in tables, rows, records, etc.
- Size in bytes
- Physical storage (CD-ROM, tape, server, etc.)
- Security requirements
- Restrictions on use
- Privacy requirements

Be sure to make note of special security and privacy issues that your data mining database will inherit from the source data. For example, many European data sets are constrained in their use by privacy regulations that are far stricter than those in the United States.

b. Data Description Describe the contents of each file or database table. Some of the properties documented in a Data Description Report are:

- Number of fields/columns
- Number/percentage of records with missing values
- Field names

For each field:

  - Data type
  - Definition
  - Description
  - Source of field
  - Unit of measure
  - Number of unique values
  - List of values
  - Range of values
  - Number/percentage of missing values
  - Collection information (e.g., how, where, conditions)
  - Timeframe (e.g., daily, weekly, monthly)
  - Specific time data (e.g., every Monday or every Tuesday)
  - Primary key/foreign key relationships

c. **Selection.** The next step in preparing the data mining database is to select the subset of data to mine. This is *not* the same as sampling the database or choosing predictor variables. Rather, it is a gross elimination of irrelevant or unneeded data. Other criteria for excluding data may include resource constraints, cost, restrictions on data use, or quality problems.

d. **Data quality assessment and data cleansing.** GIGO (Garbage In, Garbage Out) is quite applicable to data mining, so if you want good models you need to have good data. A data quality assessment identifies characteristics of the data that will affect the model quality. Essentially, you are trying to ensure not only the correctness and consistency of values but also that all the data you have is measuring the same thing in the same way.

   There are a number of types of data quality problems. Single fields may have an incorrect value. For example, recently a man's nine-digit Social Security identification number was accidentally entered as *income* when the government computed his taxes! Even when individual fields have what appear to be correct values, there may be incorrect combinations, such as pregnant males. Sometimes the value for a field is missing. Inconsistencies must be identified and removed when consolidating data from multiple sources.

   Missing data can be a particularly pernicious problem. If you have to throw out every record with a field missing, you may wind up with a very small database or an inaccurate picture of the whole database. The fact that a value is missing may be significant in itself. Perhaps only wealthy customers regularly leave the "income" field blank, for instance. It can be worthwhile to create a new variable to identify missing values, build a model using it, and compare the results with those achieved by substituting for the missing value to see which leads to better predictions.

   Another approach is to calculate a substitute value. Some common strategies for calculating missing values include using the modal value (for nominal variables), the median (for ordinal variables), or the mean (for continuous variables). A less common strategy is to assign a missing value based on the distribution of values for that variable. For example, if a database consisted of 40% females and 60% males, then you might assign a missing gender entry the value of "female" 40% of the time and "male" 60% of the time. Sometimes people build predictive models using data mining techniques to predict missing values. This usually gives a better result than a simple calculation, but is much more time-consuming.

   Recognize that you will not be able to fix all the problems, so you will need to work around them as best as possible. It is far preferable and more cost-effective to put in place procedures and checks to avoid the data quality problems — "an ounce of prevention." Usually, however, you must build the models you need with the data you now have, and avoidance is something you'll work toward for the future.

e. **Integration and consolidation.** The data you need may reside in a single database or in multiple databases. The source databases may be transaction databases used by the operational systems of your company. Other data may be in data warehouses or data marts built for specific purposes. Still other data may reside in a proprietary database belonging to another company such as a credit bureau.

   Data integration and consolidation combines data from different sources into a single mining database and requires reconciling differences in data values from the various sources. Improperly reconciled data is a major source of quality problems. There are often large

differences in the way data are defined and used in different databases. Some inconsistencies may be easy to uncover, such as different addresses for the same customer. Making it more difficult to resolve these problems is that they are often subtle. For example, the same customer may have different names or — worse — multiple customer identification numbers. The same name may be used for different entities (homonyms), or different names may be used for the same entity (synonyms). There are often unit incompatibilities, especially when data sources are consolidated from different countries; for example, U.S. dollars and Canadian dollars cannot be added without conversion.

f. Metadata construction. The information in the Dataset Description and Data Description reports is the basis for the metadata infrastructure. In essence this is a database about the database itself. It provides information that will be used in the creation of the physical database as well as information that will be used by analysts in understanding the data and building the models.

g. Load the data mining database. In most cases the data should be stored in its own database. For large amounts or complex data, this will usually be a DBMS as opposed to a flat file. Having collected, integrated and cleaned the data, it is now necessary to actually load the database itself. Depending on the DBMS and hardware being used, the amount of data, and the complexity of the database design, this may turn out to be a serious undertaking that requires the expertise of information systems professionals.

h. Maintain the data mining database. Once created, a database needs to be cared for. It needs to be backed up periodically; its performance should be monitored; and it may need occasional reorganization to reclaim disk storage or to improve performance. For a large, complex database stored in a DBMS, the maintenance may also require the services of information systems professionals.

3. Explore the data. See the DATA DESCRIPTION FOR DATA MINING section above for a detailed discussion of visualization, link analysis, and other means of exploring the data. The goal is to identify the most important fields in predicting an outcome, and determine which derived values may be useful.

In a data set with hundreds or even thousands of columns, exploring the data can be as time-consuming and labor-intensive as it is illuminating. A good interface and fast computer response are very important in this phase because the very nature of your exploration is changed when you have to wait even 20 minutes for some graphs, let alone a day.

4. Prepare data for modeling. This is the final data preparation step before building models. There are four main parts to this step:

a. Select variables
b. Select rows
c. Construct new variables
d. Transform variables

a. Select variables. Ideally, you would take all the variables you have, feed them to the data mining tool and let it find those which are the best predictors. In practice, this doesn't work very well. One reason is that the time it takes to build a model increases with the number of

variables. Another reason is that blindly including extraneous columns can lead to incorrect models. A very common error, for example, is to use as a predictor variable data that can only be known if you know the value of the response variable. People have actually used date of birth to "predict" age without realizing it.

While in principle some data mining algorithms will automatically ignore irrelevant variables and properly account for related (covariant) columns, in practice it is wise to avoid depending solely on the tool. Often your knowledge of the problem domain can let you make many of these selections correctly. For example, including ID number or Social Security number as predictor variables will at best have no benefit and at worst may reduce the weight of other important variables.

b. Select rows. As in the case of selecting variables, you would like to use all the rows you have to build models. If you have a lot of data, however, this may take too long or require buying a bigger computer than you would like.

Consequently it is often a good idea to *sample* the data when the database is large. This yields no loss of information for most business problems, although sample selection must be done carefully to ensure the sample is truly random. Given a choice of either investigating a few models built on all the data or investigating more models built on a sample, the latter approach will usually help you develop a more accurate and robust model.

You may also want to throw out data that are clearly outliers. While in some cases outliers may contain information important to your model building, often they can be ignored based on your understanding of the problem. For example, they may be the result of incorrectly entered data, or of a one-time occurrence such as a labor strike.

Sometimes you may need to add new records (e.g., for customers who made no purchases).

c. Construct new variables. It is often necessary to construct new predictors derived from the raw data. For example, forecasting credit risk using a debt-to-income *ratio* rather than just debt and income as predictor variables may yield more accurate results that are also easier to understand. Certain variables that have little effect alone may need to be combined with others, using various arithmetic or algebraic operations (e.g., addition, ratios). Some variables that extend over a wide range may be modified to construct a better predictor, such as using the log of income instead of income.

d. Transform variables. The tool you choose may dictate how you represent your data, for instance, the categorical explosion required by neural nets. Variables may also be scaled to fall within a limited range, such as 0 to 1. Many decision trees used for classification require continuous data such as income to be grouped in ranges (bins) such as High, Medium, and Low. The encoding you select can influence the result of your model. For example, the cut-off points for the bins may change the outcome of a model.

5. Data mining model building. The most important thing to remember about model building is that it is an iterative process. You will need to explore alternative models to find the one that is most useful in solving your business problem. What you learn in searching for a good model may lead you to go back and make some changes to the data you are using or even modify your problem statement.

Once you have decided on the type of prediction you want to make (e.g., classification or regression), you must choose a model type for making the prediction. This could be a decision tree, a neural net, a proprietary method, or that old standby, logistic regression. Your choice of model type will influence what data preparation you must do and how you go about it. For example, a neural net tool may require you to explode your categorical variables. Or the tool may require that the data be in a particular file format, thus requiring you to extract the data into that format. Once the data is ready, you can proceed with training your model.

The process of building predictive models requires a well-defined training and validation protocol in order to insure the most accurate and robust predictions. This kind of protocol is sometimes called supervised learning. The essence of supervised learning is to train (estimate) your model on a portion of the data, then test and validate it on the remainder of the data. A model is built when the cycle of training and testing is completed. Sometimes a third data set, called the validation data set, is needed because the test data may be influencing features of the model, and the validation set acts as an independent measure of the model's accuracy.

Training and testing the data mining model requires the data to be split into at least two groups: one for model training (i.e., estimation of the model parameters) and one for model testing. If you don't use different training and test data, the accuracy of the model will be overestimated. After the model is generated using the training database, it is used to predict the test database, and the resulting accuracy rate is a good estimate of how the model will perform on future databases that are similar to the training and test databases. It does not guarantee that the model is correct. It simply says that if the same technique were used on a succession of databases with similar data to the training and test data, the average accuracy would be close to the one obtained this way.

Simple validation. The most basic testing method is called simple validation. To carry this out, you set aside a percentage of the database as a test database, and do not use it in any way in the model building and estimation. This percentage is typically between 5% and 33%. For all the future calculations to be correct, the division of the data into two groups must be random, so that the training and test data sets both reflect the data being modeled.

After building the model on the main body of the data, the model is used to predict the classes or values of the test database. Dividing the number of incorrect classifications by the total number of instances gives an error rate. Dividing the number of correct classifications by the total number of instances gives an accuracy rate (i.e., accuracy = 1 – error). For a regression model, the goodness of fit or "r-squared" is usually used as an estimate of the accuracy.

In building a single model, even this simple validation may need to be performed dozens of times. For example, when using a neural net, sometimes each training pass through the net is tested against a test database. Training then stops when the accuracy rates on the test database no longer improve with additional iterations.

Cross validation. If you have only a modest amount of data (a few thousand rows) for building the model, you can't afford to set aside a percentage of it for simple validation. Cross validation is a method that lets you use all your data. The data is randomly divided into two equal sets in order to estimate the predictive accuracy of the model. First, a model is built on the first set and used to predict the outcomes in the second set and calculate an error rate. Then a model is built on the second set and used to predict the outcomes in the first set and again calculate an error rate. Finally, a model is built using *all* the data. There are now two independent error estimates which can be averaged to give a better estimate of the true accuracy of the model built on all the data.

Typically, the more general *n-fold cross validation* is used. In this method, the data is randomly divided into *n* disjoint groups. For example, suppose the data is divided into ten groups. The first group is set aside for testing and the other nine are lumped together for model building. The model built on the 90% group is then used to predict the group that was set aside. This process is repeated a total of 10 times as each group in turn is set aside, the model is built on the remaining 90% of the data, and then that model is used to predict the set-aside group. Finally, a model is built using all the data. The mean of the 10 independent error rate predictions is used as the error rate for this last model.

Bootstrapping is another technique for estimating the error of a model; it is primarily used with very small data sets. As in cross validation, the model is built on the entire dataset. Then numerous data sets called bootstrap samples are created by sampling from the original data set. After each case is sampled, it is replaced and a case is selected again until the entire bootstrap sample is created. Note that records may occur more than once in the data sets thus created. A model is built on this data set, and its error rate is calculated. This is called the *resubstitution* error. Many bootstrap samples (sometimes over 1,000) are created. The final error estimate for the model built on the whole data set is calculated by taking the average of the estimates from each of the bootstrap samples.

Based upon the results of your model building, you may want to build another model using the same technique but different parameters, or perhaps try other algorithms or tools. For example, another approach may increase your accuracy. No tool or technique is perfect for all data, and it is difficult if not impossible to be sure before you start which technique will work the best. It is quite common to build numerous models before finding a satisfactory one.

6. Evaluation and interpretation.

a. Model Validation. After building a model, you must evaluate its results and interpret their significance. Remember that the accuracy rate found during testing applies only to the data on which the model was built. In practice, the accuracy may vary if the data to which the model is applied differs in important and unknowable ways from the original data. More importantly, accuracy by itself is not necessarily the right metric for selecting the best model. You need to know more about the type of errors and the costs associated with them.

Confusion matrices. For classification problems, a confusion matrix is a very useful tool for understanding results. A confusion matrix (Figure 9) shows the counts of the actual versus predicted class values. It shows not only how well the model predicts, but also presents the details needed to see exactly where things may have gone wrong. The following table is a sample confusion matrix. The columns show the actual classes, and the rows show the predicted classes. Therefore the diagonal shows all the correct predictions. In the confusion matrix, you can see that our model predicted 38 of the 46 Class B's correctly, but misclassified 8 of them: two as Class A and six as Class C. This is much more informative than simply telling us an overall accuracy rate of 82% (123 correct classifications out of 150 cases).

Prediction	*Actual* **Class A**	**Class B**	**Class C**
Class A	45	2	3
Class B	10	38	2
Class C	4	6	40

Figure 9. Confusion matrix.

In particular, if there are different costs associated with different errors, a model with a lower overall accuracy may be preferable to one with higher accuracy but a greater cost to the organization due to the types of errors it makes. For example, suppose in the above confusion matrix each correct answer had a value of $10 and each incorrect answer for class A had a cost of $5, for class B a cost of $10, and for class C a cost of $20. Then the net value of the matrix would be :

(123 * $10) – (5 * $5) – (12 * $10) – (10 * $20) = $885.

But consider the following confusion matrix (Figure 10). The accuracy has dropped to 79% (118/150). However when we apply the costs from above to this confusion matrix the net value is:

(118 * $10) – (22 * $5) – (7 * $10) – (3 * $20) = $940.

Prediction	*Actual* **Class A**	**Class B**	**Class C**
Class A	40	12	10
Class B	6	38	1
Class C	2	1	40

Figure 10. Another confusion matrix.

Thus, if you wanted to maximize the value of the model, you would be better off choosing the less accurate model that has a higher net value.

The lift (gain) chart (Figure 11) is also a big help in evaluating the usefulness of a model. It shows how responses (e.g., to a direct mail solicitation or a surgical treatment) are changed by applying the model. This change ratio is called the lift. For example, instead of a 10% response rate when a random 10% of the population is treated, the response rate of a scored 10% of the population is over 30%. The lift is 3 in this case.

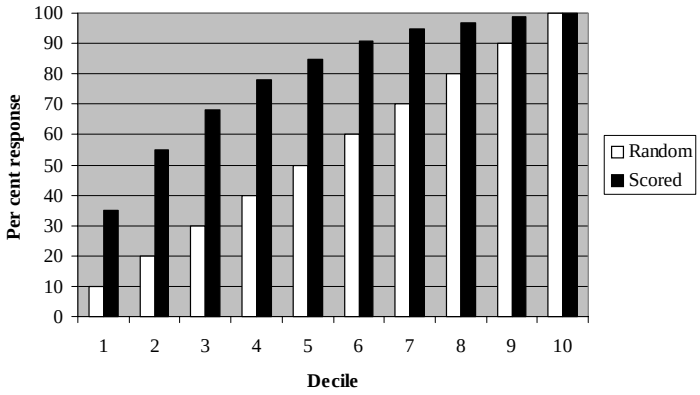

Figure 11. Lift chart.

Another important component of interpretation is to assess the value of the model. Again, a pattern may be interesting, but acting on it may cost more than the revenue or savings it generates. The ROI (Return on Investment) chart in Figure 12 is a good example of how attaching values to a response and costs to a program can provide additional guidance to decision making. (Here, ROI is defined as ratio of profit to cost.) Note that beyond the 8th decile (80%), the ROI of the scored model becomes negative. It is at a maximum at the 2nd decile (20%).

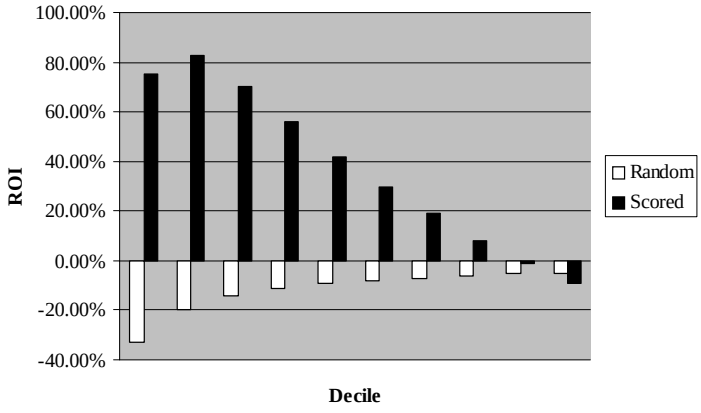

Figure 12. ROI chart.

Alternatively you may want to look at the profitability of a model (profit = revenue minus cost), as shown in the following chart (Figure 13).

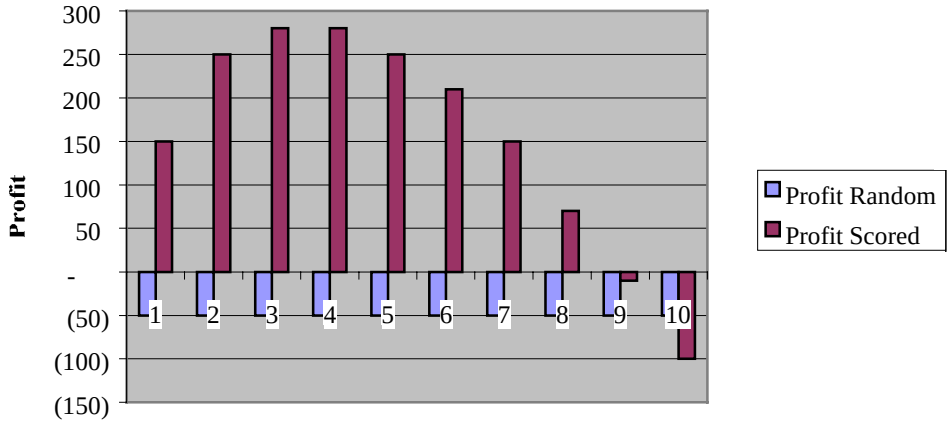

Figure 13. Profit Chart

Note that in the example we've been using, the maximum lift (for the 10 deciles) was achieved at the 1st decile (10%), the maximum ROI at the 2nd decile (20%), and the maximum profit at the 3rd and 4th deciles.

Ideally, you can act on the results of a model in a profitable way. But remember, there may be no practical means to take advantage of the knowledge gained.

b. External validation. As pointed out above, no matter how good the accuracy of a model is estimated to be, there is no guarantee that it reflects the real world. A valid model is not necessarily a correct model. One of the main reasons for this problem is that there are always assumptions implicit in the model. For example, the inflation rate may not have been included as a variable in a model that predicts the propensity of an individual to buy, but a jump in inflation from 3% to 17% will certainly affect people's behavior. Also, the data used to build the model may fail to match the real world in some unknown way, leading to an incorrect model.

Therefore it is important to test a model in the real world. If a model is used to select a subset of a mailing list, do a test mailing to verify the model. If a model is used to predict credit risk, try the model on a small set of applicants before full deployment. The higher the risk associated with an incorrect model, the more important it is to construct an experiment to check the model results.

7. **Deploy the model and results.** Once a data mining model is built and validated, it can be used in one of two main ways. The first way is for an analyst to recommend actions based on simply viewing the model and its results. For example, the analyst may look at the clusters the model has identified, the rules that define the model, or the lift and ROI charts that depict the effect of the model.

   The second way is to apply the model to different data sets. The model could be used to flag records based on their classification, or assign a score such as the probability of an action (e.g., responding to a direct mail solicitation). Or the model can select some records from the database and subject these to further analyses with an OLAP tool.

   Often the models are part of a business process such as risk analysis, credit authorization or fraud detection. In these cases the model is incorporated into an application. For instance, a predictive model may be integrated into a mortgage loan application to aid a loan officer in evaluating the applicant. Or a model might be embedded in an application such as an inventory ordering system that automatically generates an order when the forecast inventory levels drop below a threshold.

   The data mining model is often applied to one event or transaction at a time, such as scoring a loan application for risk. The amount of time to process each new transaction, and the rate at which new transactions arrive, will determine whether a parallelized algorithm is needed. Thus, while loan applications can easily be evaluated on modest-sized computers, monitoring credit card transactions or cellular telephone calls for fraud would require a parallel system to deal with the high transaction rate.

   When delivering a complex application, data mining is often only a small, albeit critical, part of the final product. For example, knowledge discovered through data mining may be combined with the knowledge of domain experts and applied to data in the database and incoming transactions. In a fraud detection system, known patterns of fraud may be combined with discovered patterns. When suspected cases of fraud are passed on to fraud investigators for evaluation, the investigators may need to access database records about other claims filed by the claimant as well as other claims in which the same doctors and lawyers were involved.

   Model monitoring. You must, of course, measure how well your model has worked after you use it. However, even when you think you're finished because your model works well, you must continually monitor the performance of the model. Over time, all systems evolve. Salespeople know that purchasing patterns change over time. External variables such as inflation rate may change enough to alter the way people behave. Thus, from time to time the model will have to be retested, retrained and possibly completely rebuilt. Charts of the residual differences between forecasted and observed values are an excellent way to monitor model results. Such charts are easy to use and understand, not computationally intensive, and could be built into the software that implements the model. Thus, the system could monitor itself.

Categories

In evaluating data mining tools you must look at a whole constellation of features, described below. You cannot put data mining tools into simple categories such as “high-end” versus “low-end” because the products are too rich in functionality to divide along just one dimension.

There are three main types of data mining products. First are tools that are analysis aids for OLAP. They help OLAP users identify the most important dimensions and segments on which they should focus attention. Leading tools in this category include Business Objects Business Miner and Cognos Scenario.

The next category includes the “pure” data mining products. These are horizontal tools aimed at data mining analysts concerned with solving a broad range of problems. Leading tools in this category include (in alphabetical order) IBM Intelligent Miner, Oracle Darwin, SAS Enterprise Miner, SGI MineSet, and SPSS Clementine.

The last category is analytic applications which implement specific business processes for which data mining is an integral part. For example, while you can use a horizontal data mining tool as part of the solution of many customer relationship management problems, you can also buy customized packages with the data mining imbedded. However, even packaged solutions require you to build and tune models that match your data. In some cases, the package requires a complete model development phase that can take months.

The following discussion of product selection applies both to horizontal tools and to the data mining component of analytic applications. But no matter how comprehensive the list of capabilities and features you develop for describing a data mining product, nothing substitutes for actual hands-on experience. While feature checklists are an essential part of the purchase decision, they can only rule out products that fall short of your requirements. Actually using a product in a pilot project is necessary to determine if it is the best match for your problem and your organization.

Basic capabilities

Depending on your particular circumstances — system architecture, staff resources, database size, problem complexity — some data mining products will be better suited than others to meet your needs. Evaluating a data mining product involves learning about its capabilities in a number of key areas (below) that may not be addressed in standard marketing materials. The Two Crows publication *Data Mining ’99: Technology Report* contains the responses of 24 vendors to detailed questionnaires that cover these topics in depth for 26 leading products.

System architecture. Is it designed to work on a stand-alone desktop machine or a client-server architecture? But note that the size of the machine on which a product runs is not a reliable indicator of the complexity of problems it can address. Very sophisticated products that can solve complex problems and require skilled users may run on a desktop computer or on a large MPP system in a client-server architecture.

Data preparation. Data preparation is by far the most time-consuming aspect of data mining. Everything a tool can do to ease this process will greatly expedite model development. Some of the functions that a product may provide include:

- Data cleanup, such as handling missing data or identifying integrity violations.
- Data description, such as row and value counts or distribution of values.
- Data transformations, such as adding new columns, performing calculations on existing columns, grouping continuous variables into ranges, or exploding categorical variables into dichotomous variables.
- Data sampling for model building or for the creation of training and validation data sets.
- Selecting predictors from the space of variables, and identifying collinear columns.

Data access. Some data mining tools require data to be extracted from target databases into an internal file format, whereas others will go directly into the native database. A data mining tool will benefit from being able to directly access the data mart DBMS using the native SQL of the database server, in order to maximize performance and take advantage of individual server features such as parallel database access. No single product, however, can support the large variety of database servers, so a gateway must be used for all but the four or five leading DBMSs. The most common gateway supported is Microsoft's ODBC (Open Database Connectivity). In some instances it is useful if the data mining tool can consolidate data from multiple sources.

Algorithms. You must understand the characteristics of the algorithms the data mining product uses so you can determine if they match the characteristics of your problem. In particular, learn how the algorithms treat the data types of your response and predictor variables, how fast they train, and how fast they work on new data.

Another important algorithm feature is sensitivity to noise. Real data has irrelevant columns, rows (cases) that don't conform to the pattern your model finds, and missing or incorrect values. How much of this noise can your model-building tool stand before its accuracy drops? In other words, how sensitive is the algorithm to missing data, and how robust are the patterns it discovers in the face of extraneous and incorrect data? In some instances, simply adding more data may be enough to compensate for noise, but if the additional data itself is very noisy, it may actually reduce accuracy. In fact, a major aspect of data preparation is to reduce the amount of noise in your data that is under your control.

Interfaces to other products. There are many tools that can help you understand your data before you build your model, and help you interpret the results of your model. These include traditional query and reporting tools, graphics and visualization tools, and OLAP tools. Data mining software that provides an easy integration path with other vendors' products provides the user with many additional ways to get the most out of the knowledge discovery process.

Model evaluation and interpretation. Products can help the user understand the results by providing measures (of accuracy, significance, etc.) in useful formats such as confusion matrices and ROI charts, by allowing the user to perform sensitivity analysis on the result, and by presenting the result in alternative ways, such as graphically.

Model deployment. The results of a model may be applied by writing directly to a database or extracting records from it. When you need to apply the model to new cases as they come, it is usually necessary to incorporate the model into a program using an API or code generated by the data mining tool. In either case, one of the key problems in deploying models is to deal with the transformations necessary to make predictions. Many data mining tools leave this as a separate job for the user or programmer.

Scalability. How effective is the tool in dealing with large amounts of data — both rows and columns — and with sophisticated validation techniques? These challenges require the ability to take advantage of powerful hardware. What kinds of parallelism does the tool support? Is there parallel use of a parallel DBMS and are the algorithms themselves parallel? What kind of parallel computers does it support — SMP servers or MPP servers? How well does it scale as the number of processors increases? Does it support parallel data access?

Data mining algorithms written for a uniprocessor machine won't automatically run faster on a parallel machine; they must be rewritten to take advantage of the parallel processors. There are two basic ways of accomplishing this. In the first method, independent pieces of the application are assigned to different processors. The more processors, the more pieces can be executed without reducing throughput. This is called inter-model parallelism. This kind of scale-up is also useful in building multiple independent models. For example, a neural net application could build multiple models using different architectures (e.g., each with a different number of nodes or hidden layers) simultaneously on each processor. But what happens if building each model takes a long time? We then need to break this model into tasks, execute those tasks on separate processors, and recombine them for the answer. This second method is called intra-model parallelism.

User interface. To facilitate model building, some products provide a GUI (graphical user interface) for semi-automatic model building, while others provide a scripting language. Some products also provide data mining APIs which can be used embedded in a programming language like C, Visual Basic, or PowerBuilder. Because of important technical decisions in data preparation and selection and choice of modeling strategies, even a GUI interface that simplifies the model building itself requires expertise to find the most effective models.

Keep in mind that the people who build, deploy, and use the results of the models may be different groups with varying skills. You must evaluate a product's user interface as to suitability for each of these groups.

Summary

Data mining offers great promise in helping organizations uncover patterns hidden in their data that can be used to predict the behavior of customers, products and processes. However, data mining tools need to be guided by users who understand the business, the data, and the general nature of the analytical methods involved. Realistic expectations can yield rewarding results across a wide range of applications, from improving revenues to reducing costs.

Building models is only one step in knowledge discovery. It's vital to properly collect and prepare the data, and to check the models against the real world. The "best" model is often found after building models of several different types, or by trying different technologies or algorithms.

Choosing the right data mining products means finding a tool with good basic capabilities, an interface that matches the skill level of the people who'll be using it, and features relevant to your specific business problems. After you've narrowed down the list of potential solutions, get a hands-on trial of the likeliest ones.

