Introduction to Data Mining and Knowledge Discovery

2nd edition

by
Two Crows Corporation

TWO CROWS

Introduction to Data Mining and Knowledge Discovery, 2nd edition
ISBN: 1-892095-00-9

TABLE OF CONTENTS

Introduction to Data Mining and Knowledge Discovery

Introduction

Data mining: What it can do

Most organizations have accumulated a great deal of data, but what they really want is *information.* What can they learn from the data about how to satisfy their best customers, how to allocate their resources most efficiently, and how to minimize losses? When there are millions of trees, how can one draw meaningful conclusions about the forest? The newest, hottest technology to address these concerns is data mining. Data mining uses sophisticated statistical analysis and modeling techniques to uncover patterns and relationships hidden in organizational databases — patterns that ordinary methods might miss.

Data mining finds these patterns and relationships by building *models.* Models, like maps, are abstract representations of reality. A map may model the route from the airport to your home, but it won't show you an accident that is slowing traffic, or construction that requires a detour. While you should never confuse your model with reality, a good model is a useful guide to understanding your business and suggests actions you can take to help you succeed.

There are two main kinds of models in data mining. The first, *predictive models,* use data with known results to develop a model that can be used to explicitly predict values for different data. For example, a model can be built using the payment history of people to whom you have given loans to help identify people who are likely to default on loans.

The second kind, *descriptive models,* describe patterns in existing data which may be used to guide decisions. The fundamental difference between these two types of models is that predictive models make an explicit prediction (such as the profitability of a customer or the likelihood of a loan default) whereas descriptive models are used to help construct a predictive model or to make an implicit prediction when they form the basis for an action or decision.

Of course, 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 not only confirm these empirical observations but find new, subtle patterns as well. This new knowledge can produce high returns by yielding steady incremental improvement. First, it gives the talented user a small advantage each year, on each project, with each customer. Compounded over a period of time, this advantage opens a large performance gap compared to those who aren't making good use of data mining. (For example, a catalog retailer that can better target its mailings can greatly increase profits by reducing the expense of mailings while simultaneously increasing the value of orders.) Second, data mining occasionally does uncover one of those rare "breakthrough" facts that can lead to radical improvements in a business.

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 when it sees an interesting pattern, send you e-mail to get your attention. 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.

It is important to remember that the descriptive or 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 targeting your marketing to people who fit the pattern, you should not assume that any of these factors cause them to buy your product.

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. For example, 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.

You also need to understand your data. For example, the quality of your results 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 problems, but it is unwise to depend on a data mining product to make all the right decisions on its own.

It is also misleading to say that data mining will find answers to questions you don't ask. While you may not have to supply it a hypothesis ("Are Californians between 18 and 25 years old likely to respond to my mailing?"), you still have to tell a data mining tool what kind of pattern you are looking for. Rather than make the vague request, "Help improve the response to my direct mail solicitation," you might use data mining to find the characteristics either of people who do respond to your solicitation, or of those who respond and make a large purchase. The patterns data mining finds for those two goals may be very different.

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.

Data mining and data warehousing

The data to be mined frequently is first extracted from an enterprise data warehouse into a 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 cleaning up data for a data warehouse and for data mining are very similar. If the data has already been cleaned up 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.

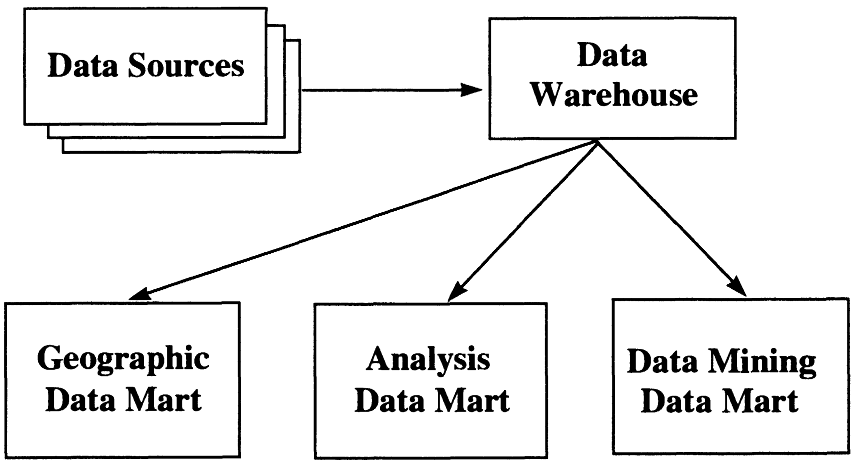

Figure 1. Data mining data mart extracted from a data warehouse.

However, 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.

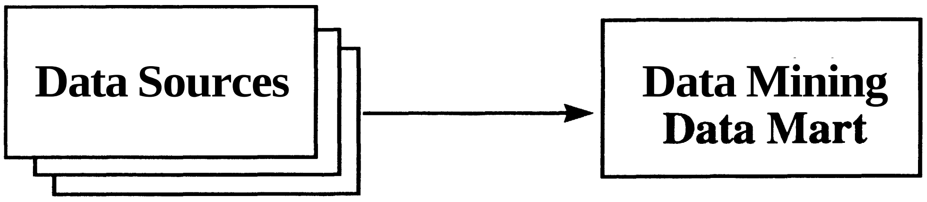

Figure 2. Data mining data mart extracted from operational databases.

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 might initially hypothesize that people with low incomes are bad credit risks and analyze the database with OLAP to verify (or disprove) this assumption. When the supposition was not borne out by the data, he might then look at high debt as the determinant of risk. If the data did not support that hypothesis, he 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. 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) this assumption.

Data mining is different from OLAP because rather than verify hypothetical patterns, it uses the data itself to uncover such patterns. 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 helps you understand 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 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.

Some examples include relatively recent algorithms like neural nets and decision trees, and new approaches to older algorithms like 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. 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 and hardware/software trends

A key enabler of data mining is the major progress in hardware price and performance. The dramatic drop in the price of computer disk storage from tens of dollars per megabyte to tens of cents per megabyte 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 random access memory, where the cost of a megabyte has dropped from hundreds of dollars to tens of dollars in just a few years. PCs routinely have 32 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.

Database management systems have also advanced to take advantage of the hardware parallelism now available. This is important because large or complex data mining problems can require large amounts of database access. If the data from an existing database can be mined, then native DBMS access will provide the best possible performance.

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

Data mining applications

Data mining is spreading throughout many organizations because of the substantial contribution it can make. It can be used to control costs as well as contribute to revenue increases.

For example, many organizations are using data mining to help manage customer relationships. By determining characteristics of customers who are likely to leave for a competitor, a company can act to retain that customer, because it is usually far less expensive to retain a customer than acquire a new one. This is sometimes referred to as reducing chum or attrition.

Database marketing is another area in which data mining is being used intensively. By identifying good candidates for mail offers or catalogs, direct mail marketers can reduce expenses and increase sales. Targeting specific promotions to existing and potential customers can have a similar good effect.

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 which 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

More important than the choice of any algorithm are the skills of the model builder and how well a program supports the model-building process. The differences in the quality of modeling results are more often due to these factors than to the algorithms.

There are two keys to success in data mining. First is coming up with a good formulation of the problem you are trying to solve. A good, focused problem 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.

This interaction is most effective when the following three components are well integrated:

1. Visualization and graphics tools that help the model builder understand the data and interpret the results.
2. Query/OLAP tools that also help the model builder understand the data and interpret the results.
3. Analytics or algorithms that build the models.

TYPES OF MODELS

The goal of data mining is to produce new knowledge that the user can act upon. It does that by building a model of the real world based on data collected from a variety of sources, including corporate transactions, customer histories, customer demographic information, and relevant external databases such as credit bureau information. The result of the model building is a description of patterns and relationships in the data.

A data mining model can generally be categorized as one of six types: classification, regression, time series, clustering, association analysis, and sequence discovery. Classification, regression and time series models are primarily useful for *prediction,* while clustering, association and sequence discovery models are primarily useful for *description* of the behavior that is captured in your database. Classification and regression are the most common types of data mining models built.

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

It helps to envision a sort of hierarchy or taxonomy of data mining solutions. At the highest level is the *business problem.* For example, you may want to find patterns in your data to help you retain good customers, so you build one model to predict customer profitability and a second model to identify customers likely to leave (attrition).

The next level of the hierarchy is the *type of model* you build. You might build a regression model to forecast amount of profit, and a classification model to predict attrition. (Confusingly, some people refer to "regression problems" rather than "regression models." It's preferable to reserve the term "problem" for referring to the overall issue being addressed.)

The next level of the hierarchy is the ***algorithm*** used to build the model. In this example, you might use a particular neural net algorithm (such as a backpropagation, radial basis functions, or Kohonen maps) to perform the regression, and a particular decision tree algorithm (such as CART, C5.0, ID3, or CHAID) to do the classification. There are also traditional statistical algorithms to choose from such as logistic regression, discriminant analysis, or general linear models.

Last is the ***product*** used to construct the model. Different products will generally have different implementations of a particular algorithm. These implementation differences will address operational characteristics such as memory usage and data storage, as well as affecting performance characteristics such as speed and accuracy.

It is important to remember that many business problems are best approached by building multiple types of models. In the above example we needed both regression and classification models. In addition to multiple types of models, you may need to try more than one algorithm for building the model. This is because, depending on the data and the business problem, different algorithms will result in the "best" model. Furthermore, it may be impossible to determine which algorithm is best until you have built models using the different approaches.

Classification, regression and time series models are sometimes referred to as ***supervised learning,*** because they are built using data that is already classified (or has known values). However, for clustering, associations and sequence discovery there is no already-known result to provide the algorithms with guidance. Hence they are sometimes referred to as ***unsupervised learning.***

Classification models require you to identify those characteristics of cases that indicate to which group each case belongs. This pattern can be used both to understand the existing data and to predict how new instances will behave. For example:

- Who is most likely to respond to a direct mail solicitation?
- Who is likeliest to move to another long-distance phone service?
- Who is a good candidate for a surgical procedure?

Data mining creates classification models by examining already classified data (cases) and inductively finding the predictive pattern. These existing cases may come from an historical database, such as people who have already undergone this surgery 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 a mailing, and the results of the mailing used to develop a classification model to be applied to the entire database. In some data mining problems, an expert classifies a sample of the database and this classification is used to create the model which will be applied to the entire database.

Regression uses existing values to forecast what other values will be. In the simplest case, this would use standard statistical techniques such as linear regression. Unfortunately, most real problems are not simply linear projections of previous values. For instance, sales volumes, stock prices, and failure rates are all very difficult to predict because they may depend on complex interactions of multiple predictor variables. Therefore, more complex techniques are necessary to forecast future values.

The same techniques can often be used for both regression and classification. For example, CART (Classification and Regression Trees) builds classification trees that classify categorical dependent variables, and regression trees that forecast continuous dependent variables. Neural nets too can create both classification and regression models.

Time series forecasting, like regression, uses series of existing values to forecast future values. Models must take into account the distinctive properties of time, especially the hierarchy of periods (including the varying definitions of them such 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 to the future.

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, 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.

Associations are items that occur together in a given event or record. Association (and sequencing) tools discover rules of the form: if item A is part of an event, then x % of the time (the confidence factor) item B is part of the event. Here are some hypothetical examples:

- If low-fat cottage cheese and non-fat yogurt are bought, then 85% of the time skim milk is purchased.
- If corn chips are purchased, then 65% of the time cola is purchased, unless there is a promotion, in which case 85% of the time cola is purchased.
- When people buy a hammer they also buy nails 50% of the time.

The use of scanners has allowed retailers to gather the transaction detail that makes association discovery possible, which is why it is sometimes called ***market-basket analysis.***

Each rule has a left-hand side ("buy a hammer") and a right-hand side ("buy nails"). Sometimes the left-hand side is called the antecedent and the right-hand side is called the consequent. In general, both the left-hand side and the right-hand side can contain multiple items, although specific implementations might not support multiple items in the right-hand side or left-hand side. Remember, these terms should not be taken to imply the existence of a causal relationship.

Sequence discovery is closely related to association analysis, except that the related items are spread over time. In order to find these sequences, not only must the details of each transaction be captured, but also the identity of the transactors. Here are some hypothetical examples:

- If surgical procedure X is performed, then 45% of the time infection Y occurs later.
- If stock A rises more than 12% and the NASDAQ index declines, then stock B will rise within two days 68% of the time.
- When people buy a hammer, they also buy nails within the next three months 18% of the time, and within the subsequent three months 12% of the time.

Most products treat sequences as associations in which the events are linked by time. It is possible for sequence discovery to also take explicit advantage of the elapsed time between transactions that make up the event. Notice how much more information there is in the second and third examples above than in the first.

The Knowledge Discovery Process

You can't simply buy a data mining algorithm such as a neural net or a decision tree package, immediately run it against some data, and expect to get a meaningful (let alone useful) result. Knowledge discovery is a process that includes a number of steps that are necessary to ensure the effective application of data mining.

The basic steps of knowledge discovery are:

1. Identify the problem
2. Prepare the data
   a. Collection
   b. Assessment
   c. Consolidation and cleaning
   d. Selection
   e. Transformation
3. Build model
   a. Evaluation and interpretation
   b. External validation
4. Use model
5. Monitor model

First and foremost, the prerequisite to knowledge discovery is understanding your data and your business. Without this understanding, no algorithm, no matter how sophisticated, 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. Let's look at each step of the knowledge discovery process:

1. **Identify the problem.** 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 also include a way of measuring the results of your knowledge discovery project. It may also include a cost-justification.

2. **Prepare the data.** This step is the most time consuming. 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. Together, these data preparation steps may take anywhere from 50% to 85% of the time and effort of the whole knowledge discovery process.

   a. **Collection.** You need to identify the sources of the data you will be mining. In addition, it is quite possible that some of the data you need has never been collected, so a data-gathering phase may be necessary. This may include the acquisition of external data from public databases (such as census or weather data) or proprietary databases (such as credit bureau data).

   b. **Assessment.** 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 survey identifies characteristics of the data that will affect the model quality.

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. When data comes from multiple sources, it must be consolidated into a single database. At that time you need to deal with consolidation problems such as inconsistent data definitions, different encodings of data, and inconsistent values for the same data item.

Even when the data comes from a single database, you need to examine it carefully for data problems such as missing data or values that violate integrity constraints. You will also need to look at things such as where the data came from, the storage data type, the units of measure, how often the data was collected and when it was collected. You will also need to understand how and where it was collected and under what conditions.

Essentially, you are trying to ensure as much as possible that all the data you have is measuring the same thing in the same way.

c. **Consolidation and cleaning.** In this step you build the database from which you choose the data to mine. You will consolidate the data and repair, insofar as possible, the problems you identified in the data survey. You may be sure that you will not be able to fix all the problems and so you will need to work around them as best as possible. Seemingly simple solutions can actually make things worse. For example, ignoring rows with important data missing may lead to missing the best predictor — the fact that the data was missing.

d. **Data selection.** Once you have gathered the data you need for the models you will be building, you need to select the specific data for each model. For a predictive model, this usually means selecting the independent variables (or columns), the dependent variables, and the cases (or rows) that you will use to train the model.

   While in principle some data mining algorithms will automatically ignore irrelevant variables and properly account for related (or 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 independent variables will at best have no benefit and at worst may reduce the weight of other important variables.

   Graphics tools that help you visualize the data and its relationships can be very useful in this step. They can help identify important independent variables and reveal collinear variables.

   You may want to throw out data that are clearly outliers. While in some cases outliers may contain important information 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.

   Similarly, 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.

e. **Transformation.** Once you have selected your data, there may be some additional data transformations required. You may want to use derived data to build your model; for example, forecasting credit risk using a debt-to-income ratio rather than just debt and income as independent variables. The tool you choose may also dictate how you represent your data. For instance, neural nets work best when categorical data such as State is broken up into multiple dichotomous variables, each with a "yes" or "no" value (the so-called categorical explosion). Many decision trees used for classification require continuous data such as income to be grouped in ranges (or bins) such as High, Low, or Medium. Notice that the encoding you select can influence the result of your model.

3. **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. In fact, 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.

   The process of model building differs between supervised learning (as in classification, regression, and time series problems) and unsupervised learning (as in clustering, association and sequence detection).

   The following description focuses on supervised learning, which is the most well-defined procedure. Unsupervised learning and pattern identification and description may also require the iterative building of models until a satisfactory model is reached, but do not have the well-defined training and validation steps described below.

   Once you have decided on the type of model you want to build, you must choose an algorithm for building that model. This could be a decision tree, a neural net, a proprietary method, or that old standby, logistic regression. Your choice of algorithm 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, and provide you the means for doing this. 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 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. 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.

   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 testing test 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.

If you only have a modest amount of data for building the model, a more reliable method of validation is cross validation, in which the data is randomly divided into two equal sets. A model is built using the entire dataset, but the problem is how to estimate the error of this model. The solution is to build a model on the first set, predict the outcomes in the second set, and calculate an error rate. A model is then built on the second set, used to predict the outcomes in the first set, and an error rate calculated. There are now two independent error estimates which can be averaged to give a better estimate of the true error rate.

For smaller databases of a few thousand rows or less, the database is usually divided into more than just two groups to improve error estimation, and 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, and the model built on the remaining 90% of the data, and then 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 smaller datasets. As in cross validation, the model is built on the entire dataset. Then numerous training data sets are created by re-sampling with replacement from the original dataset and the *resubstitution* error is calculated. Note that records may occur more than once in the training dataset thus created. At least 200 (and sometimes over 1,000) training datasets are created. Usually, the final error estimate is calculated by taking the average of the estimates from each of the bootstrap test sets.

Based upon the results of this model building effort, you may want to build another model using the same technique but different parameters, or perhaps to 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.

For each model you build, you must go through the training and testing steps. Furthermore, different tools or techniques may require different preparation of the data or different file formats, requiring you to repeat portions of those steps. The amount of time it takes to find the right model can be significantly shortened by algorithms that take advantage of parallel computing.

a. **Evaluation and interpretation.** After building a model, you must evaluate its results and interpret their significance. As part of the model validation you found an accuracy rate. It is important to remember that this accuracy rate 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.

   For classification problems, a confusion matrix is a very useful tool for understanding results. A confusion matrix (Figure 3) 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 record the actual classes and the rows the predicted classes. Therefore the diagonal records all the correct predictions. In the confusion matrix, we can see that our model predicted 38 of the 46 Class B's correctly, but misclassified 8 of them, 2 as Class A and 6 as Class C. This is much more informative than simply telling us an overall accuracy rate of 82%. 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.

		Actual	
F	**Class A**	**Class B**	**Class C**
Class A			
Class B	[illegible]	[illegible]	[illegible]
Class C	[illegible]	[illegible]	40

Figure 3. Confusion Matrix

   Another important evaluation criterion is the understandability of the model. For many business applications it is important to be able to explain why a decision was made, whereas in others, even slight improvements in accuracy are so important that interpretability is of less interest. In general, decision trees and rules-based systems best convey the reasons implicit in a model's prediction. However, even a tree or rule can become so complicated as to be uninterpretable.

   The lift (or gain) chart (Figure 4) 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. 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%. This improvement in response is called the lift.

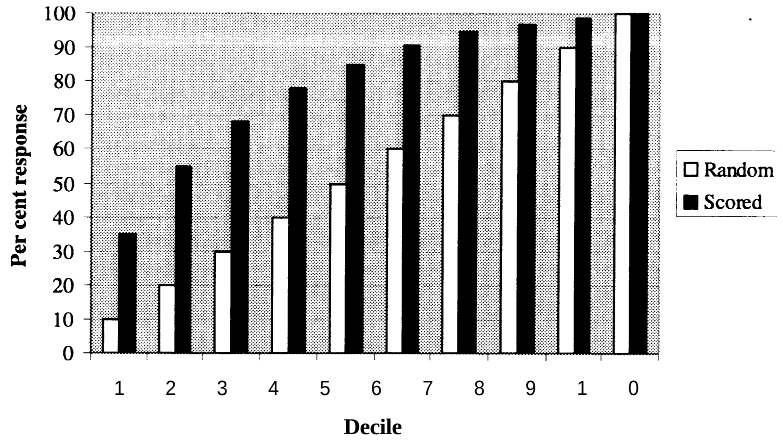

Figure 4. Lift Chart.

Another important component of interpretation is to assess the value of the model. A pattern may be interesting, but acting on it may cost more then the revenue or savings it generates. The ROI (Return on Investment) chart below (Figure 5) is a good example of how attaching values to a response and costs to a program can provide additional guidance to decision making. Note that beyond the 8th decile, the ROI of the scored model becomes negative. It is at maximum at the 2nd decile.

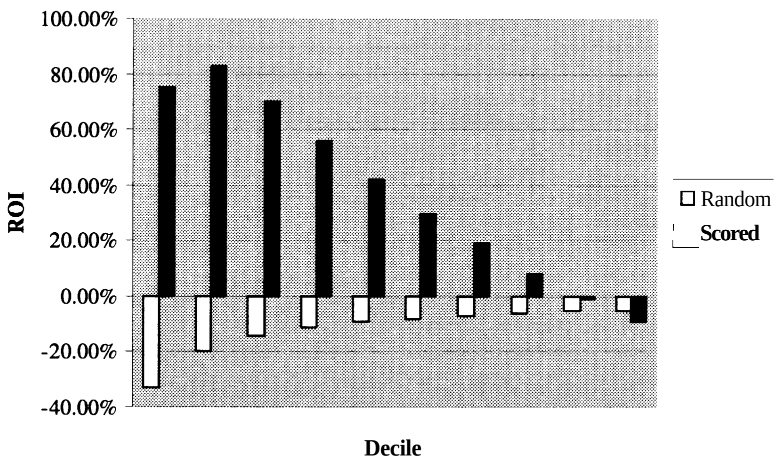

Figure 5. ROI Chart.

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 really does reflect 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, variables such as inflation rate may not be part of a model to predict the propensity of an individual to buy, but a jump in inflation from 3% to 17% will certainly affect people's behavior. Another cause is that there are inevitably problems with the data itself which might possibly lead to an incorrect model.

Therefore it is important to test a model in the real world. For example, if a model is used to select a subset of a mailing list, it may be wise to do a test mailing to verify the model. If a model is used to predict credit risk, it may be prudent to 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.

4. **Use the model.** 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 simply view 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.

   Often, the analyst may want to apply the model to some different data sets. Some ways this can be done include flagging records based on their classification, or assigning a score such as the probability of an action (e.g., responding to a direct mail solicitation). Alternatively, the analyst may use the model to select some records from the database and subject these to further analyses with an OLAP tool. If the database is sufficiently large, parallel database access and model application may be necessary.

   Based on these uses of the model, the analyst may then recommend actions that the business can take. However, 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 simulation that a manager can use for decision making such as planning promotions. Alternatively, the 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 probably be easily 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. This requires access to the general database.

5. **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 and the data that they produce changes. Salespeople know that purchasing patterns change over time. External variables such as inflation rate may change enough to alter the way people behave and the factors that influence their behavior. Even precise manufacturing systems age and cause drift in the data they produce. 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.

Data Mining Tools and Technologies

Having discussed the types of problems for which data mining is useful, we will describe some of the types of tools used to solve those problems. Many products use variations of algorithms and tools that have been published in computer science or statistics journals, with their specific algorithms and 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 or address shortcomings of the published version of these approaches. Some vendors have proprietary algorithms that are not extensions or enhancements of any published approach, but which may work quite well.

Most of the tools 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 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 tool or set of tools can or should be used exclusively. For any given problem, the nature of the data itself will affect the choice of tool you choose. Consequently, you will need a variety of tools and technologies in order to find the best possible model.

Neural networks

The development of neural networks took place largely in the artificial intelligence community. Neural networks are loosely inspired by biology in that they are represented as networks of simple neuron-like processors. Actual biological neural networks are incomparably more complex than their artificial counterparts. Real neurons are living cells with complex biochemistry, a complex connectivity, and a variety of mechanisms by which the network's elements and their connections may be altered.

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 independent variables that have many interactions. They may be used in classification problems (i.e., the output is a categorical variable) or for regressions (i.e., the output variable is continuous).

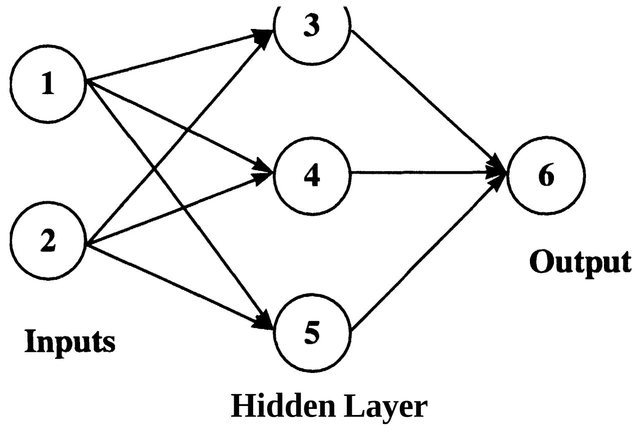

Figure 6. A neural network with one hidden layer.

A neural network (Figure 6) starts with an ***input layer,*** where each node corresponds to an independent variable (also called predictor or input). These input nodes are connected to a number of nodes in a ***hidden layer.*** Each input node is usually 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 dependent variables.

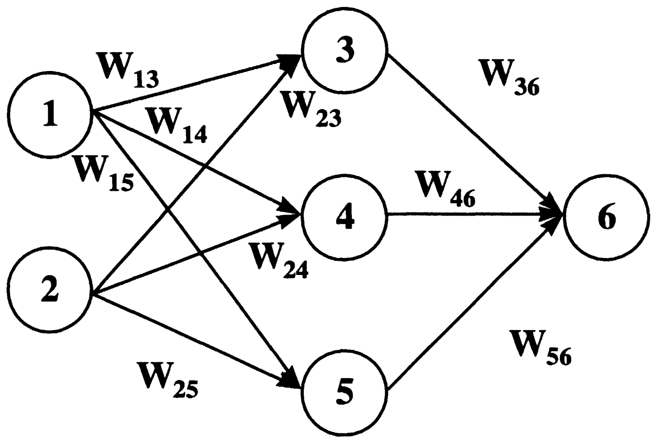

Figure 7. W_{xy} is the weight from node x to node y.

Figure 7 illustrates how 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}), 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. 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. In fact, if there is a linear activation function and no hidden layer, neural nets are equivalent to a linear regression.

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)

The connection weights 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 choice of input and output variables, the number of nodes and hidden layers, and their connectivity. 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.

We will focus our attention on a particular type of neural network known as the feed-forward backpropagation network, which is the one most commonly used. 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. 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. This is the "feed-forward" part. Next, 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 is the "backpropagation" part. 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 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. The problem, of course, is how well it will do on other data. The danger is an overfitted neural network which will only work well on the training data. The question therefore is to 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. If further training does not reduce the error on the test set, then the user should go back to an earlier model.

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 6. 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, the weights associated with a given output are not unique. Consequently, the network weights in general do not aid in understanding the underlying process generating the prediction. However, this is acceptable and even desirable in many applications. A bank may want to automatically recognize handwritten applications, but does not care what the form of the functional relationship is 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 the neural network representation is that it can easily be implemented in massively parallel computers with each node simultaneously doing its calculations. Several commercial neural network chips and boards are available.

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. This is a point often misunderstood by practitioners who view the good fit as a success rather than a drawback of the network.

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.

Finally, neural networks tend to work best when the problem is very 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 8 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.

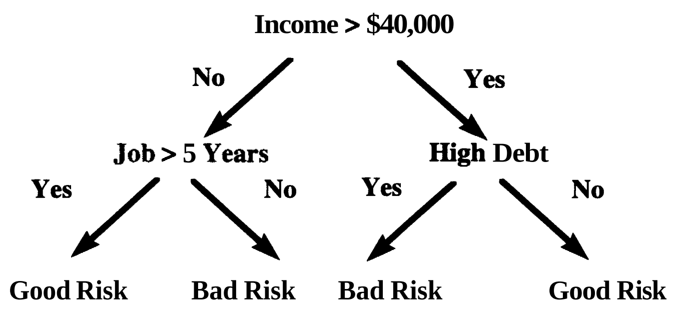

Figure 8. A simple classification tree.

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.

Each branch will lead either to another decision node, or to the bottom of the tree (i.e., a leaf node or output class). At this node, we know the class to which an individual is assigned.

By navigating the decision tree you can classify a case by deciding which branch to take, starting at the root node and moving to each subsequent node until an output class 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."

Data mining is used to create decision trees by examining the data and inducing the rules that make up the tree. 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 dependent variable with a dozen output classes. Such a tree would be extremely difficult to understand. However, 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? Finally, decision trees become more difficult to interpret as they increase in size.

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 independent 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. For example, 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.

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 independent 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 independent variables harder to detect.

Decision trees that are not limited to univariate splits could use multiple independent 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 explosion of independent variables inherent in neural nets.

Some classification trees were designed for and therefore work best when the independent 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 output variables (i.e., will not build regression trees), in which case the dependent variables in the training set must also be binned to output classes.

Rule induction

Rule induction is a method for deriving a set of rules to classify cases. As we saw above, any decision tree can generate a set of rules, and while that is sometimes called rule induction, there is also a different meaning. 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.

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 9). 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.

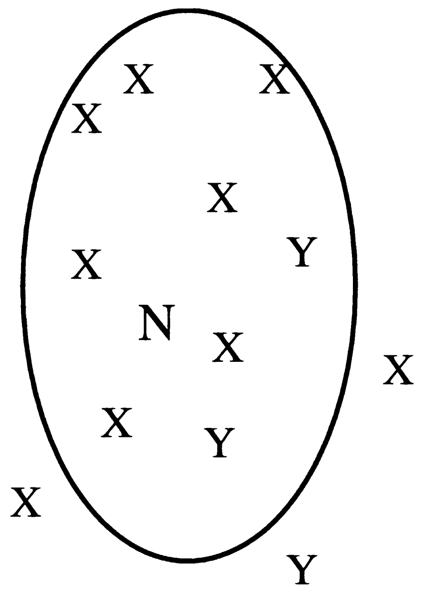

Figure 9. K-nearest neighbor. N is a new case. It would be assigned to the class X because the seven X's within the ellipse outnumber the two Y's.

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 the calculation be made for every 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 independent 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.

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 do add a lot of computational overhead.

Association and sequence detection

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 detection* is very similar, in that a sequence is an association related over time.

Associations are written as $A \Rightarrow B$, 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 is 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 a total 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.

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% (1511,000)
Support for "hammer, nails and lumber" = 0.5% (511,000)
Confidence of "hammer ⇒ nails" = 30% (15150)
Confidence of "nails ⇒ hammer" = 19% (15180)
Confidence of "hammer and nails ⇒ lumber" = 33% (5115)
Confidence of "lumber ⇒ hammer and nails" = 25% (5120)

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%)

The hardware retailer might interpret these figures as showing that hammer-and-nails transactions are a significant predictor of lumber sales — more so than hammer sales are for nail sales. If lumber is a high-margin item, this could suggest a profitable marketing strategy.

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 organize the events that make up an association or transaction 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 skates are the consequent, that have a confidence factor of over 80% and a support of over 2%").

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.

Sequence detection adds a time variable that enables you to track series of events to find interesting behavior patterns over time.

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.

Logistic regression

Logistic regression is a generalization of linear regression. It is used for predicting a binary variable (with values such as yes/no or 0/1). Because the dependent variable is binary, it cannot be modeled directly by linear regression.

Therefore, rather than predict whether the event itself (the dependent variable) will occur, we build the model using 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 the a particular team will win a football game, we mean that the probability of their winning is three times as great as the probability of their losing. So they have (we believe) 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 (i.e., 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.

Just as with neural nets, logistic regression is a classification tool when used to predict categorical variables such as whether an individual is likely to purchase or not, and a regression tool when used to predict continuous variables such as the probability that an individual will make a purchase.

While logistic regression is a very powerful modeling tool, it assumes that the dependent 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 predictor variables and specify their functional relationship to the dependent variable. So, for example, the modeler must choose among income or the square of the income or the logarithm of the income as a predictor variable. Additionally the modeler must explicitly put in any interactions as predictor variables. 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. It is interesting to note that logit transformation plays exactly the same role in logistic regression as the activation function plays in neural nets, and is the main reason that a no-hidden-layer neural net is just logistic regression.

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 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 independent 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 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 classification of a binary response. The main added feature is that the linear assumption is lifted. The output variable can be virtually any function of the predictor as long as it doesn't have 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 like 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.

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 (or 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.

Visualization

Data visualization is often used along with data mining tools. The importance of visualization to effective data analysis cannot be overemphasized. While statistical modeling and confirmatory analyses are vital to the production stage of data analysis, it is data visualization that most often provides the ***Aha!*** leading to new insights and success. As more complex patterns in data are being explored, more powerful and sophisticated tools for visualizing data are being developed too.

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 age, sex, marital status, own-or-rent, years in job, etc., and credit risk. Consequently, visualization tools must use clever representations to collapse n dimensions into two.

Color-blind users or people who are not spatially oriented may have problems with visualization tools. Also, while visualizations convey a great deal of information, they often require people to train their eye through practice.

Selecting Data Mining Products

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.

Let's examine how products differ in a number of significant areas:

1. **Model types.** This refers to classification, regression, time series, clustering, associations, and sequence discovery, as discussed above. Some products specialize in only a few of these types, while others offer a suite of model-building approaches.

2. **Problem complexity.** Problems increase in complexity as the amount of data increases, including not only the number of rows but the number of columns (or attributes), the number of unique values in categorical variables, and the number of independent variables. The amount of interaction among variables, and non-linearity in the variables and the parameters, also contribute to complexity. Furthermore, as the patterns become more subtle and as the need for accuracy rises, finding the right patterns becomes harder.

   Here are some of the ways data mining products enable users to deal with complex problems:

   a. **Diversity of model types.** Combinations of models (see above) can often help find useful patterns in data. For example, using clustering to segment a data set and then developing predictive models for the segments may produce more accurate predictions than building a model on the whole data set.

   b. **Diversity of algorithms.** Many problems, especially those involving classification, will reveal their patterns differently to different algorithms. This may be because of the type of data, or the nature of the patterns themselves. **A** product that provides alternative algorithms for building a model is more likely to be able to handle complications that arise. Some of these algorithms were discussed in the section on data mining tools and technology.

   c. **Validation methods.** When training (or estimating) a model there are a number of possible validation methods. More sophisticated methods such as n-fold cross validation or bootstrapping can help maximize accuracy while controlling overfitting (see above for a discussion of validation methods).

   d. **Data selection and transformation.** Often the patterns hidden in the data are obscured by the number of attributes (or columns) for each instance. Sometimes these columns are slightly different measures of the same thing (such as age and date of birth). Other columns may be irrelevant. Furthermore, the way the data is represented in the columns can affect the tool's ability to find a good model. It is important for systems to deal with this data complexity by providing tools to guide you in selecting appropriate columns and transforming their values.

   e. **Visualization.** Visualization tools provide a mechanism for conveying enormous amounts of information in a compact representation. This can reduce the complexity of the modeling task by, for example, helping the model builder identify important data or evaluate the quality of a model.

f. **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.

3. **Skill level for model building and model deployment.** No data mining product should be used by someone unfamiliar with the data, the application, or model-building. We also need to differentiate between people who build the models and those who use them, as they are often different groups. Skill levels vary; the person may be an analyst who knows the business and how to use the tool, but is not particularly knowledgeable in model building, or the person may be an expert model builder who lacks experience with the organization's business and data.

   To facilitate model building, some products provide a GUI 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.

   A model may be deployed by running it against an existing collection of data or against new cases as they come in. Some tools provide a simple GUI interface for executing the model that allows a business analyst to explore the patterns. Other tools allow the export of the model to a program or a database, using for example a set of rules in SQL or a procedural language like C.

4. **Basic capabilities.** All data mining products must address certain basic areas:

   a. **System architecture.** Is it designed to work on a stand-alone desktop machine or a client-server architecture? It is important to recognize 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. Quite possibly the only things the system architecture may indicate are the amount of data that can be mined and the price of the product.

b. 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:

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

c. 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 scalable 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.

d. Algorithms. You must understand the characteristics of the algorithms the data mining product uses to determine if they match the characteristics of your problem. In particular, learn how the algorithms treat the data types of your dependent and independent 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 (or 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.

e. Model evaluation and interpretation. The patterns and relationships that data mining models produce need to be examined and their value assessed. Products can help the user understand the results by providing measures (e.g., for accuracy and significance) in useful formats such as confusion matrices, by allowing the user to perform sensitivity analysis on the result, and by presenting the result in alternative ways, such as graphically.

f. 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.

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. (All cars have four wheels and an engine, but they certainly feel different on the road.) While feature checklists are an essential part of the purchase decision, they can only rule out products. Actually using a product in a pilot project is necessary to determine if it is the best match for your problem and your organization.

DATA MINING PRODUCTS: PACKAGED SOLUTIONS

The Two Crows data mining study is focused on general-purpose data mining packages intended for knowledge discovery in a broad range of problem domains. However, we should mention another class of data mining products: packaged solutions. These are complete applications customized for a particular data mining problem or for a particular vertical market. Some examples of packaged solutions include products for fraud detection, credit risk assessment, and real estate appraisal. 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 last months.

SUMMARY

- Data mining offers great promise in helping organizations uncover hidden patterns in their data. Models can be used to predict as well as describe. 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.