Introduction to Data Mining

Data mining is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. It has been defined as:

The automated analysis of large or complex data sets in order to discover significant patterns or trends that would otherwise go unrecognised.

The key elements that make data mining tools a distinct form of software are:

Automated analysis

Data mining automates the process of sifting through historical data in order to discover new information. This is one of the main differences between data mining and statistics, where a model is usually devised by a statistician to deal with a specific analysis problem. It also distinguishes data mining from expert systems, where the model is built by a knowledge engineer from rules extracted from the experience of an expert.

The emphasis on *automated* discovery also separates data mining from OLAP and simpler query and reporting tools, which are used to verify hypotheses formulated by the user. Data mining does not rely on a user to define a specific query, merely to formulate a goal - such as the identification of fraudulent claims.

Large or complex data sets

One of the attractions of data mining is that it makes it possible to analyse very large data sets in a reasonable time scale. Data mining is also suitable for complex problems involving relatively small amounts of data but where there are many fields or variables to analyse. However, for small, relatively simple data analysis problems there may be simpler, cheaper and more effective solutions.

Discovering significant patterns or trends that would otherwise go unrecognised

The goal of data mining is to unearth relationships in data that may provide useful insights.

Data mining tools can sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions, performance bottlenecks in a network system and identifying *anomalous data* that could represent data entry keying errors. The ultimate significance of these patterns will be assessed by a domain expert - a marketing manager or network supervisor - so the results must be presented in a way that human experts can understand.

Data mining tools can also automate the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms to enhance the value of existing information resources, and can be implemented on new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing systems, they can analyse massive databases to deliver answers to questions such as:

"Which clients are most likely to respond to my next promotional mailing, and why?"

Data mining is ready for application because it is supported by three technologies that are now sufficiently mature:

- Massive data collection
- Powerful multiprocessor computers
- Data mining algorithms

Commercial databases are growing at unprecedented rates, especially in the retail sector. The accompanying need for improved computational engines can now be met in a cost-effective manner with parallel multiprocessor computer technology. Data mining algorithms embody techniques that have existed for at least 10 years, but have only recently been implemented as mature, reliable, understandable tools that consistently outperform older statistical methods.

The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. Today, the maturity of these techniques, coupled with high-performance relational database engines and broad data integration efforts, make these technologies practical for current data warehouse environments.

The key to understanding the different facets of data mining is to distinguish between data mining applications, operations, techniques and algorithms.

Applications	Database marketing customer segmentation customer retention fraud detection credit checking web site analysis
Operations	Classification and prediction clustering association analysis forecasting
Techniques	Neural networks decision trees K-nearest neighbour algorithms naive Bayesian cluster analysis

Applications

A data mining application is an implementation of data mining technology that solves a specific business or research problem. Example application areas include:

- A pharmaceutical company can analyze its recent sales force activity and their results to improve targeting of high-value physicians and determine which marketing activities will have the greatest impact in the next few months. The data needs to include competitor market activity as well as information about the local health care systems. The results can be distributed to the sales force via a wide-area network that enables the representatives to review the recommendations from the perspective of the key attributes in the decision process. The ongoing, dynamic analysis of the data warehouse allows best practices from throughout the organization to be applied in specific sales situations.
- A credit card company can leverage its vast warehouse of customer transaction data to identify customers most likely to be interested in a new credit product. Using a small test mailing, the attributes of customers with an affinity for the product can be identified. Recent projects have indicated more than a 20-fold decrease in costs for targeted mailing campaigns over conventional approaches.

- A diversified transportation company with a large direct sales force can apply data mining to identify the best prospects for its services. Using data mining to analyze its own customer experience, this company can build a unique segmentation identifying the attributes of high-value prospects. Applying this segmentation to a general business database such as those provided by Dun & Bradstreet can yield a prioritized list of prospects by region.
- A large consumer package goods company can apply data mining to improve its sales process to retailers. Data from consumer panels, shipments, and competitor activity can be applied to understand the reasons for brand and store switching. Through this analysis, the manufacturer can select promotional strategies that best reach their target customer segments.

Operations

An application that uses data mining technology will implement one or more data mining operations (sometime referred to as data mining ‘tasks’). Each operation reflects a different way of distinguishing patterns or trends in a complex data set.

Classification and prediction

Classification is the operation most commonly supported by commercial data mining tools. It is an operation that enables organisations to discover pat-terns in large or complex data sets in order to solve specific business problems.

Classification is the process of sub-dividing a data set with regard to a number of specific outcomes. For example, we might want to classify our customers into ‘high’ and ‘low’ categories with regard to credit risk. The category or ‘class’ into which each customer is placed is the ‘outcome’ of our classification.

A crude method would be to classify customers by whether their income is above or below a certain amount. A slightly more subtle approach tries to find a linear relationship between two different factors - such as income and age -to divide a data set into two groups. Real-world classification problems usually involve many more dimensions and therefore require a much more complex delimitation between different classes.

An example of classification A financial services organisation wishes to identify those customers likely to be interested in a new investment opportunity. It has sold a similar product before and has historical data showing which of its customers responded to the previous offer. The aim is to understand which factors identify likely responders to the offer, so that the marketing and sales effort can be targeted more efficiently

There is a field in the customer record that is set to true or false, depending on whether a customer did or did not respond to the offer. This field is called the ‘target field’ or ‘dependent variable’ for the classification. The aim is to analyse the way other attributes of the customer (such as level of income, type of job, age, sex, marital status, number of years as a customer, and other types of investments or products purchased) influence the class to which they belong (as indicated by the target field). This information will usually be stored in other fields in the customer record. The various fields included in the analysis are called ‘independent’ or ‘predictor’ fields or variables.

Techniques for classification How the data mining tool analyses this data, and the type of information it provides, depends on the techniques it uses. The most common techniques for classification are decision trees and neural networks. If a decision tree is used, it will provide a set of branching conditions that successively split the customers into groups defined by the values in the independent variables. The aim is to be able to produce a set of rules or a model of some sort that can identify a high percentage of responders. A decision tree may formulate a condition such as:

> *customers who are male and married and have incomes over £50,000 and who are home-owners responded to our offer*

The condition selects a much high percentage of responders than if you took a random selection of customers.

In contrast, a neural network identifies which class a customer belongs to, but cannot tell you why. The factors that determine its classifications are not available for analysis, but remain implicit in the network itself. Anther set of techniques used for classification are k-nearest neighbour algorithms

Understanding and prediction Sophisticated classification techniques enable us to discover new patterns in large and complex data sets. Classification is therefore a powerful aid to understanding a particular problem, whether it is response rates to a direct mailing campaign or the influence of various factors on the likelihood of a patient recovering from cancer.

In some instances, improved understanding is sufficient. It may suggest new initiatives and provide information that improves future decision making. However, in many instances the reason for developing an accurate classification model is to improve our capability for prediction.

For example, we know that historically 60 per cent of customers who are male, married and have incomes over £60,000 responded to a promotion (compared with only three per cent of all targeted customers). There is therefore a better than average chance that new customers who fit this profile will also be interested in our product. In practice, data mining algorithms can find much more complex relationships involving numerous predictor variables, thus providing a much finer segmentation of customers.

A classification model is said to be ‘trained’ on historical data, for which the outcome is known for each record. It is then applied to a new, unclassified data set in order to predict the outcome for each record.

There are important differences between classifying data in order to under-stand the behaviour of existing customers and using that classification to predict future behaviour. For a historical data set, it is often possible to produce a set of rules or a mathematical function that classifies every record accurately. For example, if you keep refining your rules you end up with a rule for each individual of the form:

100 per cent of customers called Smith who live at 28 Arcadia Street responded to our offer.

Such a rule is of little use for predicting the classification of a new customer. In this case, the model is said to be ‘over-trained’ or ‘overfitted’ to the historical data set.

Building a good predictive model requires that overfitting be avoided by testing and tuning a model to ensure that it can be ‘generalised’ to new data.

Clustering

Clustering is an unsupervised operation. It is used where you wish to find groupings of similar records in your data without any preconditions as to what that similarity may involve. Clustering is used to identify interesting groups in a customer base that may not have been recognised before. For example, it can be used to identify similarities in customers’ telephone usage, in order to devise and market new call services.

Clustering is usually achieved using statistical methods, such as a k-means algorithm, or a special form of neural network called a Kohonen feature map network. Whichever method is used, the basic operation is the same. Each record is compared with a set of existing clusters, which are defined by their ‘centre’. A record is assigned to the cluster it is nearest to, and this in turn changes the value that defines that cluster. Multiple passes are made through a data set to re-assign records and adjust the cluster centres until an optimum solution is found.

Looking for clusters amongst supermarket shoppers, for example, may require the analysis of many factors, including the number of visits made per month, the total spend per visit, spend per product category, time of visit and payment method.

Clustering is often undertaken as an exploratory exercise before doing further data mining using a classification technique. For this reason, good visualisation support is a helpful adjunct to clustering: it enables you to ‘play’ with your clusters in order to see if the clusters identified make sense and are useful in a business context.

Association analysis and sequential analysis

Association analysis is an unsupervised form of data mining that looks for links between records in a data set. Association analysis is sometimes referred to as ‘market basket analysis’, its most common application. The aim is to discover, for example, which items are commonly purchased at the same time to help retailers organise customer incentive schemes and store layouts more efficiently.

Consider the following beer and nappy example:

500,000 transactions
20,000 transactions contain nappies (4%)
30,000 transactions contain beer (6%)
10,000 transactions contain both nappies and beer (2%)

Support (or prevalence) measures how often items occur together, as a percentage of the total transactions. In this example, beer and nappies occur together 2% of the time (10,000/500,000).

Confidence (or predictability) measures how much a particular item is dependent on another. Because 20,000 transactions contain nappies and 10,000 also contain beer, when people buy nappies, they also buy beer 50% of the time. The confidence for the rule:

When people buy nappies they also buy beer 50% of the time.

is 50%..

The inverse rule, which would be stated as:

When people buy beer they also buy nappies 1/3 of the time

has a confidence of 33.33% (computed as 10,000/30,000).

Note that these two rules have the same support (2% as computed above). Support is not dependent on the direction (or implication) of the rule; it is only dependent on the set of items in the rule.

In the absence of any knowledge about what else was bought, we can also make the following assertions from the available data:

People buy nappies 4% of the time.
People buy beer 6% of the time.

These numbers -- 4% and 6% -- are called the ***expected confidence*** of buying nappies or beer, regardless of what else is purchased.

Lift measures the ratio between the confidence of a rule and the expected confidence that the second product will be purchased. Lift is one measure of the strength of an effect. In our example, the confidence of the nappies-beer buying rule is 50%, whilst the expected confidence is 6% that an arbitrary customer will buy beer. So, the lift provided by the nappies-beer rule is 8.33 (= 50%/6%).

The nappies-beer rule could have been expressed explicitly in terms of lift as:

People buy nappies are 8.33 times more likely to buy beer too.

The interaction between nappies and beer is very strong. A key goal of an association or sequencing data mining exercise is to find rules that have a substantial lift like this.

Although rules with high confidence and support factors are important, those with lower levels may indicate less obvious patterns that suggest new marketing opportunities.

A key requirement for association analysis is the capacity to analyse very large databases. The numbers involved can be daunting: large retailers typically monitor over 100,000 product lines and handle millions of customer transactions each week.

Association analysis is not limited to retail applications. An insurance company, for example, might use it to look at links between health insurance claims for different conditions. It may also use sequential analysis to track the relationships between claims over time.

Sequential analysis is sometimes considered as a separate data mining operation, although we group it with association analysis. Sequential analysis looks for temporal links between purchases, rather than relationships between items in a single transaction. Sequential analysis will typically produce rules such as:

Ten per cent of customers who bought a tent bought a backpack within one month.

Forecasting

Classification identifies a specific group or class to which an item belongs. A prediction based on a classification model will, therefore, be a discrete outcome, identifying a customer as a responder or non-responder, or a patient as having a high or low risk of heart failure. Forecasting, in contrast, concerns the prediction of continuous values such as share values, the level of the stock market, or the future price of a commodity such as oil.

Forecasting is often done with regression functions - statistical methods for examining the relationship between variables in order to predict a future value. Statistical packages, such as SAS and SPSS, provide a wide variety of such functions that can handle increasingly complex problems. However, such statistical functions usually require significant knowledge of the techniques used and the pre-conditions that apply to their implementation. Data mining tools can also provide forecasting functions. In particular, neural networks have been widely used for stock market forecasting.

An important distinction can be made between two different types of forecasting problem.

The simpler problem is that of forecasting a single continuous value based on a series of unordered examples. For example, predicting a person’s in-come based on various personal details. Many data mining tools can provide this form of forecasting using, for example, neural networks or, in some cases, decision trees.

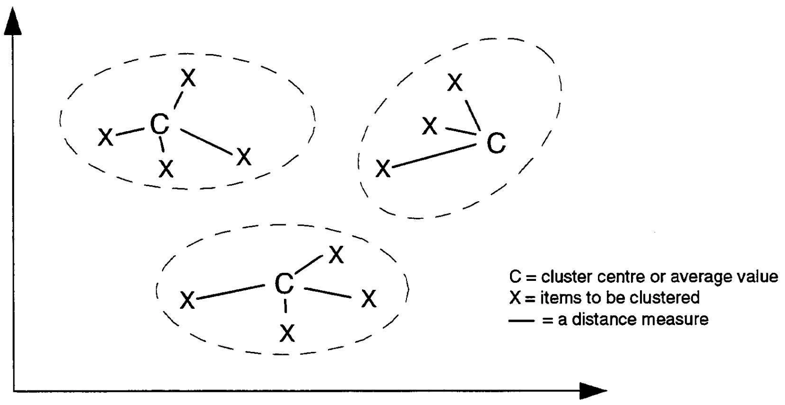
A more complex problem is to predict one or more values based on a sequential pattern, such as the stock market level for the next 30 days based on figures for the previous six months. Fewer data mining tools support this form of forecasting. The limited support for extended time-series forecasting partly reflects the increased algorithmic complexity of the problem, and partly the need to prepare and present the data to the data mining tool in the right way and to provide output in the necessary format. Where it is supported, it usually requires analysts to do much more pre-processing of the data and post-processing of the results.

Techniques

A data mining operation is achieved using one of a number of techniques or methods. Each technique can itself be implemented in different ways, using a variety of algorithms.

Clustering Algorithms

Cluster analysis is the process of identifying the relationships that exist between items on the basis of their similarity and dissimilarity. Unlike classification, clustering does not require a target variable to be identified beforehand. A clustering algorithm takes an unbiased look at the potential groupings within a data set and attempts to derive an optimum delineation of items on the basis of those groups.


C = cluster centre or average value
X = items to be clustered
— = a distance measure

Clusters are typically based around a “centre” or average value. How centres are initially defined and adjusted varies between algorithms. One method is to start with a random set of centres, which are then adjusted, added to and removed as the analysis progresses.

To identify items that belong to a cluster, some measure must be used that gauges the similarity between items within a cluster and their dissimilarity to items in other clusters. The similarity and dissimilarity between items is typically measured as their distance from each other and from the cluster centres within a multi-dimensional space, where each dimension represents one of the variables being compared.

Nearest Neighbour

Nearest Neighbour (more precisely k-nearest neighbour, also k-NN) is a predictive technique suitable for classification models.

Unlike other predictive algorithms, the training data is not scanned or processed to create the model. Instead, the training data is the model. When a new case or instance is presented to the model, the algorithm looks at all the data to find a subset of cases that are most similar to it and uses them to predict the outcome.

There are two principal drivers in the k-NN algorithm: the number of nearest cases to be used (k) and a metric to measure what is meant by nearest.

Each use of the k-NN algorithm requires that we specify a positive integer value for k. This determines how many existing cases are looked at when predicting a new case. k-NN refers to a family of algorithms that we could denote as 1-NN, 2-NN, 3-NN, and so forth. For example, 4-NN indicates that the algorithm will use the four nearest cases to predict the outcome of a new case.

As the term *nearest* implies, k-NN is based on a concept of distance, and this requires a metric to determine distances. All metrics must result in a specific number for comparison purposes. Whatever metric is used is both arbitrary and extremely important. It is arbitrary because there is no preset definition of what constitutes a "good" metric. It is important because the choice of a metric greatly affects the predictions. Different metrics, used on the same training data, can result in completely different predictions. This means that a business expert is needed to help determine a good metric.
To classify a new case, the algorithm computes the distance from the new case to each case (row) in the training data. The new case is predicted to have the same outcome as the predominant outcome in the k closest cases in the training data.

A credit risk example:

Consider a bank that seeks to minimise loan defaults. Loan officers must be able to identify potential credit risks during the loan approval cycle. The problem is one of simple classification: to predict whether or not an applicant will be a good or poor credit risk. The following table shows the training dataset for this problem. For simplicity the size has been limited to five records – a very small dataset.

Name	Debt	Income	Married?	Risk
Peter	High	High	Yes	Good
Sue	Low	High	Yes	Good
John	Low	High	No	Poor
Mary	High	Low	Yes	Poor
Fred	Low	Low	Yes	Poor

This dataset contains information about people to whom the institution previously loaned money. The lender determined if each applicant was a good or poor credit risk. Because Risk is what we wish to predict, it is the dependent column, also called the target variable. The other columns used to build the model are known as independent columns. In this example they include debt level (High or Low), income level (High or Low), and marital status (Yes or No). The Name column will be ignored because it is highly unlikely that a person's name affects his credit risk. In our example, all the columns, except Name, have two possible values. The restriction to two values is only to keep the example simple.

We will use k=3, or 3-NN. For our distance measure we will use a simple metric that is computed by summing scores for each of the three independent columns, where the score for a column is 0 if the values in the two instances are the same, and 1 if they differ.

We can now compute the distance between Peter and Sue. The three column scores for Peter and Sue are 1, 0, and 0 because they have different values only in the Debt column. The distance between Peter and Sue – the sum of these scores – is equal to 1. If two cases have all values in common, their distance is zero.

	Peter	Sue	John	Mary	Fred
Peter	0	1	2	1	2
Sue	1	0	1	2	1
John	2	1	0	3	2
Mary	1	2	3	0	1
Fred	2	1	2	1	0

This table summarises all the distances between all the records in our training dataset. Note that this matrix is symmetrical; only the numbers below (or above) the diagonal of zeroes are needed. However, presenting it this way makes it easy to see all of the distances for any one case (row) at a glance.

Now we are ready to apply the k-NN technique to see how it classifies our training data. Remember that we chose k=3, so we are interested in the three neighbours nearest to Peter. The distances in column 1 show that Peter, Sue, and Mary are Peter's three nearest neighbours because they have the lowest distance scores.

But how can Peter be his own neighbour? Isn't this somehow invalid, to use the information we have about Peter to predict an outcome for Peter? Well, in one sense it is, but this is absolutely no different than for any other model when it is tested with the training data. This is why we always need to use a separate test dataset to validate a model.

The Risk values for Peter's three nearest neighbours (Peter himself, Sue, and Mary) are *Good*, *Good*, and *Poor*, respectively. The predicted Risk for Peter is the value that is most frequent among the *k* neighbours, or *Good*, in this case.

Who are Sue's nearest 3 neighbours? Clearly Sue herself, but what about Peter, John and Fred, who are all the same distance (1) from Sue? We could include all three, exclude all three, or include all three with a proportionate vote (2/3 each in this case). The decision is entirely up to the implementers of the algorithm. For our example, we'll use 2/3 vote each, resulting in votes of *Good (Sue herself), 2/3 Good, 2/3 Poor*, and *2/3 Poor*, for a consensus of *Good*.

The following table enumerates the predictions from the 3-NN algorithm. The accuracy for 3-NN on the training set is 80 percent. Results from using 2-NN in this case would be exactly the same.

Name	Debt	Income	Married?	Risk	3-NN Prediction
Peter	High	High	Yes	Good	Good
Sue	Low	High	Yes	Good	Good
John	Low	High	No	Poor	-
Mary	High	Low	Yes	Poor	Poor
Fred	Low	Low	Yes	Poor	Poor

Since both the value of *k* and the distance metric have tremendous impact on the quality of the model it is important to choose them carefully.

What is a good value for *k*? The 1-NN algorithm tries to find one case in the training set that most closely matches the case we are trying to predict. When it is found, the predicted value is assumed to be the outcome of the matching case. For 1-NN accuracy on the training set will be 100 percent by definition. However, 1-NN, because it looks for only the one best match, is extremely susceptible to noise and will never reflect any kind of pattern in the data. Many commercial algorithms start with a default k value of 10. The best value for k will require some experimentation with the data at hand, comparing results for different k values against a test dataset.

k-NN does not make a learning pass through the data. Unfortunately, this does not make k-NN an especially efficient algorithm, particularly with large training datasets. That's because all the processing is postponed until predictions are being made. Then each new case being predicted requires a complete pass through the training data to compute the distance between the target instance and each instance in the training set. The algorithm keeps track of the k nearest instances as it proceeds through the training set, predicting an outcome when the pass is complete.

While the nearest neighbour technique is simple in concept, the selection of *k* and the choice of distance metrics pose definite challenges. The absence of any real training process distinguishes the nearest neighbour technique from the other predictive methods. Both k and the distance metrics need to be set, and both will require some experimentation. The model builder will need to build multiple models, validating each with independent test sets to determine which values are appropriate.

Neural Networks

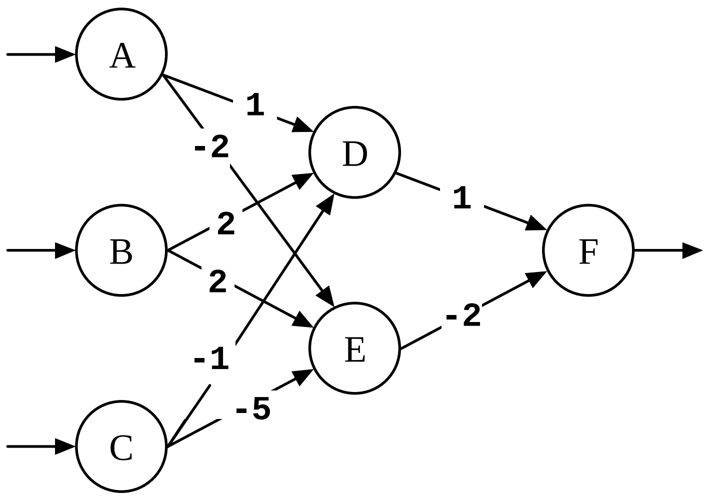
A key difference between neural networks and many other techniques is that neural nets only operate directly on numbers. As a result, any nonnumeric data in either the independent or dependent (output) columns must be converted to numbers before we can use the data with a neural net.

The following table shows our sample credit risk data with all two-valued categorical variables converted into values of either 0 or 1. *High* Debt and Income, Married=*Yes*, and *Good* Risk were all replaced by the value 1. *Low* Debt and Income, Married=*No*, and *Poor* Risk were replaced by 0. No conversion is necessary for the Name column because it is not used as an independent or dependent column.

Name	Debt	Income	Married?	Risk
Peter	1	1	1	1
Sue	0	1	1	1
John	0	1	0	0
Mary	1	0	1	0
Fred	0	0	1	0

Consider, first, a trained neural net that can be used to predict *Good* and *Poor* risks for our credit risk classification problem. This network contains six *nodes*, which we have marked A through F. The left-hand nodes (A, B and C) are *input* nodes and constitute the *input layer*. The input nodes correspond to the independent variable columns in the credit risk problem (Debt, Income, and Married).

The right-hand node (F) is the *output* node and makes up the *output layer*. In this case there is only one output node, corresponding to Risk, the dependent column. But neural network techniques in general do not restrict the number of output columns. That there can be multiple outputs representing multiple simultaneous predictions is one way that neural nets differ from most other predictive techniques.



The two middle nodes (D and E) are the *hidden* nodes and constitute a single *hidden layer*. The number of hidden nodes and, for that matter, the number of hidden layers, are set at the user's discretion. The number of hidden nodes often increases with the number of inputs and the complexity of the problem. Too many hidden nodes can lead to ***overfitting*** (where the network has become so specialised that it deals perfectly with the training set but is pretty hopeless with new cases)**,** and too few hidden nodes can result in models with poor accuracy. Finding an appropriate number of hidden nodes is an important part of any data mining effort with neural nets. The number of input, hidden, and output nodes is referred to as the neural net ***topology*** or the network ***architecture***.

The diagram shows ***weights*** on the arrows between the nodes. Typically, there are no weights on the arrows coming into the input layer or coming out of the output layer. The values of the other weights are determined during the neural net training or learning process. Note that weights can be both positive and negative. Neural net algorithms usually restrict weights to a narrow range such as between plus and minus 1 or between plus and minus 10. Although weights are typically real numbers we have used integers to simplify our calculations.

The heart of the neural net algorithm involves a series of mathematical operations that use the weights to compute a ***weighted sum of the inputs*** at each node. In addition, each node also has a ***squashing function*** that converts the weighted sum of the inputs to an output value. For our neural net we will use a very simple squashing function:

if the weighted sum of the inputs is greater than zero, the output is 1, otherwise the output is 0.

Equations for the output values at nodes D, E and F can now be written as follows:

D = If (A + 2B - C) > 0 Then 1 Else 0

E = If (-2A + 2B - 5C) > 0 Then 1 Else 0

F = If (D - 2E) > 0 Then 1 Else 0

The following table shows the sample data, with the three independent variables (Debt, Income, and Married) converted to numbers, the actual risk, and the computed values for nodes D, E, and F. The output value for F (1 on the first row) is the predicted value of Risk for Peter. It equals the actual value, so the neural net made a correct prediction in this case. In fact, our net makes correct predictions for all five rows in the training set, as shown in this table:

Node:	**A**	**B**	**C**		**D**	**E**	**F**
Name	**Debt**	**Income**	**Married**	**Risk**			
Peter	1	1	1	1	1	0	1
Sue	0	1	1	1	1	0	1
John	0	1	0	0	1	1	0
Mary	1	0	1	0	0	0	0
Fred	0	0	1	0	0	0	0

Finding the best combination of weights is a significant searching problem. A number of search techniques, have been used to search for the best combination of weights. The most common is a class of algorithms called ***gradient descent***. A gradient descent algorithm starts with a solution, usually a set of weights that have been randomly generated. Then a case from the learning set is presented to the net. The net (initially with random weights) is used to compute an output, the output is compared to the desired result, and the difference, called the error, is computed. The weights are then altered slightly so that if the same case were presented again, the error would be less. This gradual reduction in error is the descent.

The most common gradient descent algorithm is called ***back-propagation***. It works backwards from the output node, computing in turn each prior node's contribution to the error. From this it is possible to compute not only each node’s but also each weight's share of the error. In this way, the error is propagated backwards through the entire network, resulting in adjustments to all weights that contributed to the error.

This cycle is repeated for each case in the training set, with small adjustments being made in the weights after each case. When the entire training set has been processed, it is processed again. Each run through the entire training set is called an ***epoch***. It is quite possible that training the net will require several hundred or even several thousand epochs. In this way, even though each case results in only small adjustments to the weights in each epoch, each case is seen in each of several hundred (or several thousand) epochs, resulting in a much larger cumulative effect.

Neural net algorithms use a number of different ***stopping rules*** to control when training ends. Common stopping rules include:

- Stop after a specified number of epochs.
- Stop when an error measure falls below a preset level.
- Stop when the error measure has seen no improvement over a certain number of epochs.

Naïve-Bayes

Naïve-Bayes is a classification technique that is both predictive and descriptive. It analyses the relationship between each independent variable and the dependent variable to derive a conditional probability for each relationship.

Naïve-Bayes requires only one pass through the training set to generate a classification model. This makes it the most efficient data mining technique. However, Naïve-Bayes does not handle continuous data, so any independent or dependent variables that contain continuous values must be binned or bracketed. For instance, if one of the independent variables is age, the values must be transformed from the specific value into ranges such as "less than 20 years," "21 to 30 years," "31 to 40 years" and so on.

Using Naïve-Bayes for classification is a fairly simple process. During training, the probability of each outcome (dependent variable value) is computed by counting how many times it occurs in the training dataset. This is called the *prior* probability. For example, if the *Good* Risk outcome occurs 2 times in a total of 5 cases, then the prior probability for *Good* Risk is 0.4. You can think of the prior probability in the following way: "If I know nothing else about a loan applicant, there is a 0.4 probability that the applicant is a *Good* Risk." In addition to the prior probabilities, Naïve-Bayes also computes how frequently each independent variable value occurs in combination with each dependent (output) variable value. These frequencies are then used to compute conditional probabilities that are combined with the prior probability to make the predictions.

Consider the credit risk classification problem whose goal is to be able to predict whether an applicant will be a *Good* or a *Poor* Credit Risk. In our example, all the columns – both independent and dependent -- are categorical, so we do not have to convert any values to categorical variables by binning (grouping) values.

From the sample data, we cross-tabulate counts of each Risk outcome (*Good* or *Poor*) and each value in the independent variable columns. For example, row 3 reports two cases where Income is *High* and Risk is *Good* and one case where Income is *High* and Risk is *Poor*.

		Counts	**Counts**	**Likelihood**	
Independent Variable	**Value**	**Good Risk**	**Poor Risk**	**given Good Risk**	**given Poor Risk**
Debt	High	1	1	0.50	0.33
Debt	Low	1	2	0.50	0.67
Income	High	2	1	1.00	0.33
Income	Low	0	2	0	0.67
Married	Yes	2	2	1.00	0.67
Married	No	0	1	0	0.33
Total by Risk		**2**	**3**		

We can easily compute the **likelihood** that one of the independent variables has a particular value, given a known risk level, by using the count and dividing by the "Total by Risk" number (on the bottom row). For example, the likelihood that a *Good* Risk has *High* Income is 1.00 (see row 3). That's because both instances of *Good* Risk have *High* Income. In the same way, the conditional probability that a *Poor* Risk has a *Low* Income is 0.67, because two out of three *Poor* risks have *Low* Income (see row 4).

The bottom row is also used to compute the **prior probabilities** for *Good* and Poor Risk. In our data the prior probability for Good is 0.40 (two of five cases) and the prior probability for *Poor* is 0.60 (three of five cases).

Given a particular case, we can compute a score, related to posterior probability, for both values of risk level simply by multiplying the prior probability for risk level by all the likelihood figures from the above table for the independent variables (Bayes' Theorem). The highest scoring value becomes the predicted value.

Name	**Debt**	**Income**	**Married?**	**Risk Actual**	**Good Risk Score**	**Poor Risk Score**	**Risk Predicted**
Peter	High	High	Yes	Good	0.2	0.04	Good
Sue	Low	High	Yes	Good	0.2	0.09	Good
John	Low	High	No	Poor	0	0.04	Poor
Mary	High	Low	Yes	Poor	0	0.09	Poor
Fred	Low	Low	Yes	Poor	0	0.18	Poor

For example, in the above table the first row for Peter in the training set has *High* Debt, *High* Income and Married is *Yes*. In the first table the likelihoods associated with these values and *Good* Risk are 0.5, 1.0 and 1.0 respectively (see rows 1, 3 and 5). The product of these three numbers and the prior probability for Good (0.40) is 0.20 (0.50 x 1 x 1 x 0.40). For *Poor* Risk the probabilities (also from rows 1, 3 and 5) are 0.33, 0.33, and 0.67. With a prior probability for *Poor* Risk of 0.60, the score for Poor Risk is 0.044 (0.33 x 0.33 x 0.67 x 0.60). Because the score for *Good* is higher, we predict that Peter will be a *Good* risk.

The second table presents the actual risk, the scores for *Good* risk and *Poor* risk, and the predicted risk for all cases in the sample data. With all outcomes predicted correctly, the algorithm has 100 percent accuracy on the training set. Although this is a good sign, we must also validate the model by using it to predict the outcomes for separate test data, but we will skip that step.

The scores that we computed can also be easily converted to real **posterior probabilities**, by dividing each score by the sum of all scores for that case. For example, the posterior probability that Peter is a *Good* Risk is approximately 82 percent (0.2 divided by 0.244, which is the sum of the *Good* Risk value of 0.2 and the *Poor* Risk value of 0.044.). The posterior probability that Peter is a *Poor* Risk is 18 percent.

Note that we don't need to know values for all the independent variables to make a prediction. In fact, if we know none of them, we can still predict using just the prior probability. If we know only the value for Income, we can use the conditional probabilities associated with Income to modify the prior probabilities, and make a prediction. This ability to make predictions from partial information is a significant advantage of Naïve-Bayes.

The use of Bayes' Theorem in computing the scores and posterior probabilities in this way is valid only if we assume a statistical *independence* between the various independent variables such as debt and income. (Hence the term "naïve".) Despite the fact that this assumption is usually not correct, the algorithm appears to have good results in practice.

Decision Trees

Decision trees are one of the most common data mining technique and are by far the most popular in tools aimed at the business user. They are easy to set up, their results are understandable by an end-user, they can address a wide range of classification problems, they are robust in the face of different data distributions and formats, and they are effective in analysing large numbers of fields.

A decision tree algorithm works by splitting a data set in order to build a model that successfully classifies each record in terms of a target field or variable. An example is a decision tree which classifies a data set according to whether customers did or did not buy a particular product.

The most common types of decision tree algorithm are CHAID, CART and C4.5. CHAID (Chi-square automatic interaction detection) and CART (Classification and Regression Trees) were developed by statisticians. CHAID can produce tree with multiple sub-nodes for each split. CART requires less data preparation than CHAID, but produces only two-way splits. C4.5 comes from the world of machine Learning, and is based on information theory.

In order to generate a decision tree from the training set of data we need to split the data into progressively smaller subsets. Each iteration considers the data in only one node. The first iteration considers the root node that contains all the data. Subsequent iterations work on derivative nodes that will contain subsets of the data.

At each iteration we need to chose the independent variable that most effectively splits the data. This means that the subsets produced by splitting the data according to the value of the independent variable should be as "homogeneous" as possible, with respect to the dependent variable.

We shall use the relatively simple ID3 algorithm (a predecessor of C4.5) to produce a decision tree from our credit risk dataset. The ID3 algorithm uses the entropy concept from information theory to measure how much uncertainty there is in a set, with respect to the value of the dependent variable. Entropy is measured on a scale from 0 to 1. If a set were split 50-50 between good and poor risks, we would be completely uncertain whether a person picked at random from the set would be a good or poor risk. In this case, the entropy of the set would be 1. If, on the other hand, the whole set were good risks there would be no uncertainty and the entropy would be 0. Similarly if they were all poor risks.

We start out by measuring the entropy of the complete training set. We find the proportion p_1 of good risks in the set and the proportion p_2 of poor risks in the set. Shannon's formula for **entropy** is:

$$\text{entropy} = -\sum_i p_i \log_2 p_i$$

where we take $p \log_2 p = 0 \text{ if } \mathrm{p} = 0$, and i runs over the different subsets.

NOTE: When using a calculator, we can employ the identity: $\log_2 p = \dfrac{\log_{10} p}{\log_{10} 2}$

There are two good risks and three poor risks in the complete training set, and so:

$$\text{entropy} \quad = -(\frac{2}{5}\log_2 \frac{2}{5} + \frac{3}{5}\log_2 \frac{3}{5})$$

= -(0.4*(-1.32193) + 0.6*(-0.73697))

= -(-0.52877 -0.44218)

= **0.97095**

We then consider all the possible ways of splitting the set - by debt, income and marital status - and calculate the overall entropy of the resulting subsets for each of the three cases, in turn.

Consider, first, splitting the training set by debt. The following is a cross-tabulation of the training set by debt and by risk:

	Good Risk	Poor Risk	Total
High Debt	1	1	2
Low Debt	1	2	3
Total	2	3	5

The subset with high debt has one good risk and one poor risk, and so:

$$\text{entropy} \quad = -(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2})$$

= -(0.5*(-1) + 0.5*(-1))

= -(-0.5 -0.5)

= **1.00000**

as we would expect for a set that is split down the middle.

The subset with low debt has one good risk and two poor risks, and so:

$$\text{entropy} \quad = -(\frac{1}{3}\log_2\frac{1}{3}+\frac{2}{3}\log_2\frac{2}{3})$$

= -((1/3)*(-1.58496) + (2/3)*(-0.58496))

= -(-(0.52832) -(0.38998))

= **0.91830**

Since there are altogether two high debts and three low debts, the average (or expected) entropy for the two subsets resulting from splitting by debt is:

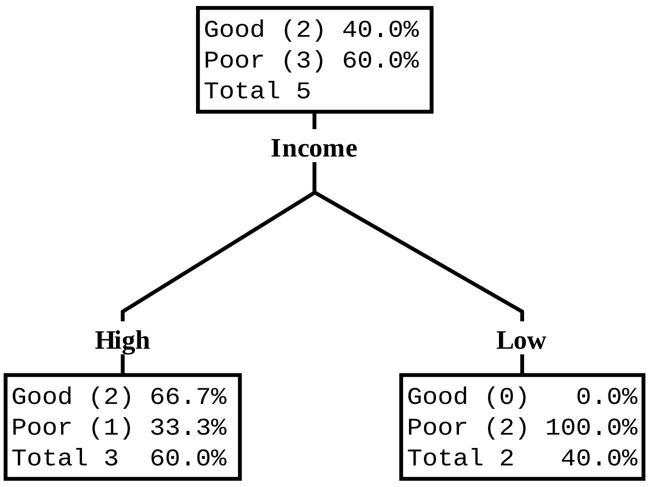
(2/5)*1 + (3/5)*0.91830 = 0.95098

In other words, splitting by debt reduces the entropy by:

0.97095 - 0.95098 = **0.01998**

This value is called the **Information Gain**. Similar calculations show that splitting the training set by income or by marital status yields information gains of 0.41998 or of 0.17095, respectively.

So, splitting by income produces the greatest information gain. It is the most effective way of reducing the entropy in the training data set, and thereby producing as homogeneous subsets as possible:



The second of these subsets (low income) consists of 100% poor risks. Since it is totally homogeneous (and has an entropy of 0), there is no more work to be done on that branch.

But the first branch is a mix of good and poor risks. It has an entropy of 0.91830 and needs to be split by a second independent variable. Should it be debt or marital status?

Consider, first, splitting the high income set by debt. The following is a cross-tabulation of the training set by debt and by risk:

	Good Risk	Poor Risk	Total
High Debt	1	0	1
Low Debt	1	1	2
Total	2	1	3

The subset with high debt has one good risk and no poor risk. It is completely homogeneous and has an entropy of 0. The subset with low debt has one good risk and one poor risk. It is split down the middle and has an entropy of 1.

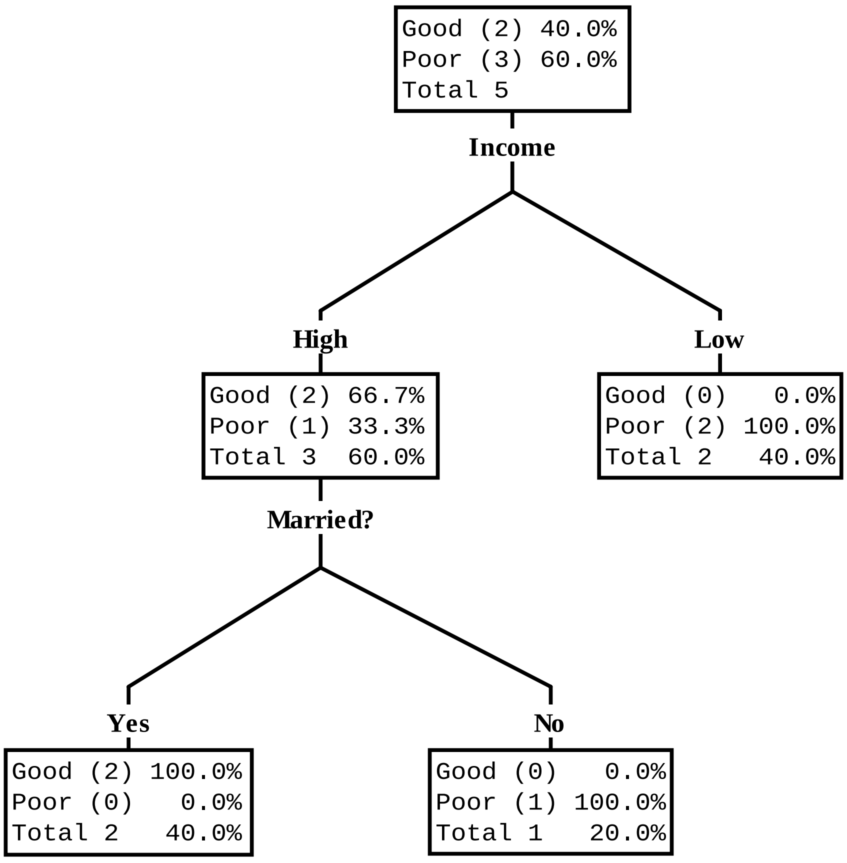
Since there are altogether one high debt and two low debts, the average (or expected) entropy for the two subsets resulting from splitting the high incomes by debt is:

(1/3)*0 + (2/3)*1 = 0.66667

The entropy of the high income subset was 0.91830. So splitting the high incomes by debt produces an information gain of:

0.91830 - 0.66667 = 0.25163

On the other hand if we use marital status, we obtain two completely homogeneous subsets and so the entropy is brought down to zero. This is the attribute by which to split high incomes.



No more splitting needs to be done and we have a decision tree that we can apply to new cases. The operational tree can be expressed as a set of rules:

IF Income = High AND Married? = Yes THEN Risk = Good
IF Income = High AND Married? = No THEN Risk = Poor
IF Income = Low THEN Risk = Poor

A shortcoming of the information gain criterion for splitting sets is that it favours attributes with many values. Such attributes would tend to split the set into large numbers of small subsets, which would be likely to be quite homogeneous, regardless of any true correlation between the attribute and the classification.

To overcome this problem, we can instead use the **Information Gain Ratio**, calculated by dividing the Information Gain of an attribute by its **Split Information**, i.e., the amount of information required to determine the value of that attribute, for a random member of the training set. This will tend to be higher for attributes with more values and so the Information Gain Ratio will be reduced for such many- valued attributes.

Taking the credit risk example, with “Name” as an attribute, the information gain values are:

Attribute	**Information Gain**
Debt	0.0200
Income	0.4200
Marital Status	0.1710
Name	0.9710

Under the Information Gain criterion, Name would be the attribute selected for splitting the set, leading five wholly homogeneous subsets and a tree that classified the training set perfectly, but was unable to cope with new data.

Since there are three high income and two low income people in the original training set, the split information for Income is:

$$-(\frac{3}{5}\log_2\frac{3}{5}+\frac{2}{5}\log_2\frac{2}{5}) = 0.9710$$

and so the Information Gain Ratio for Income is:

$$\text{information gain ratio} = \frac{\text{information gain}}{\text{split information}} = \frac{0.4200}{0.9710} = 0.4325$$

Since splitting by name would produce five subsets, each of size 1, the split information for Name is:

$$-(\frac{1}{5}\log_2\frac{1}{5}+\frac{1}{5}\log_2\frac{1}{5}+\frac{1}{5}\log_2\frac{1}{5}+\frac{1}{5}\log_2\frac{1}{5}+\frac{1}{5}\log_2\frac{1}{5}) = 2.3219$$

and so the Information Gain Ratio for Name is:

$$\text{information gain ratio} = \frac{\text{information gain}}{\text{split information}} = \frac{0.9710}{2.3219} = 0.4182$$

Similarly, the Information Gain Ratios for Debt and Marital Status are 0.0206 and 0.2368, respectively.

Since Income has the highest information gain ratio, it is chosen for the first level split, as it had been under the information gain criterion, before Name was introduced (mischievously) as another an attribute.