Data Mining Page | White Papers | Data Mining Tutorial

An Overview of Data Mining Techniques

Excerpted from the book **Building Data Mining Applications for CRM**
by Alex Berson, Stephen Smith, and Kurt Thearling

Introduction

This overview provides a description of some of the most common data mining algorithms in use today. We have broken the discussion into two sections, each with a specific theme:

- Classical Techniques: Statistics, Neighborhoods and Clustering
- Next Generation Techniques: Trees, Networks and Rules

Each section will describe a number of data mining algorithms at a high level, focusing on the "big picture" so that the reader will be able to understand how each algorithm fits into the landscape of data mining techniques. Overall, six broad classes of data mining algorithms are covered. Although there are a number of other algorithms and many variations of the techniques described, one of the algorithms from this group of six is almost always used in real world deployments of data mining systems.

I. Classical Techniques: Statistics, Neighborhoods and Clustering

1.1. The Classics

These two sections have been broken up based on when the data mining technique was developed and when it became technically mature enough to be used for business, especially for aiding in the optimization of customer relationship management systems. Thus this section contains descriptions of techniques that have classically been used for decades the next section represents techniques that have only been widely used since the early 1980s.

This section should help the user to understand the rough differences in the techniques and at least enough information to be dangerous and well armed enough to not be baffled by the vendors of different data mining tools.

The main techniques that we will discuss here are the ones that are used 99.9% of the time on existing business problems. There are certainly many other ones as well as proprietary techniques from particular vendors - but in general the industry is converging to those techniques that work consistently and are understandable and explainable.

1.2. Statistics

By strict definition "statistics" or statistical techniques are not data mining. They were being used long before the term data mining was coined to apply to business applications. However, statistical techniques are driven by the data and are used to discover patterns and build predictive models. And from the users perspective you will be faced with a conscious choice when solving a "data mining" problem as to whether you wish to attack it with statistical methods or other data mining techniques. For this reason it is important to have some idea of how statistical techniques work and how they can be applied.

What is different between statistics and data mining?

I flew the Boston to Newark shuttle recently and sat next to a professor from one the Boston area Universities. He was going to discuss the drosophila (fruit flies) genetic makeup to a pharmaceutical company in New Jersey. He had compiled the world's largest database on the genetic makeup of the fruit fly and had made it available to other researchers on the internet through Java applications accessing a larger relational database.

He explained to me that they not only now were storing the information on the flies but also were doing "data mining" adding as an aside "which seems to be very important these days whatever that is". I mentioned that I had written a book on the subject and he was interested in knowing what the difference was between "data mining" and statistics. There was no easy answer.

The techniques used in data mining, when successful, are successful for precisely the same reasons that statistical techniques are successful (e.g. clean data, a well defined target to predict and good validation to avoid overfitting). And for the most part the techniques are used in the same places for the same types of problems (prediction, classification discovery). In fact some of the techniques that are classical defined as "data mining" such as CART and CHAID arose from statisticians.

So what is the difference? Why aren't we as excited about "statistics" as we are about data mining? There are several reasons. The first is that the classical data mining techniques such as CART, neural networks and nearest neighbor techniques tend to be more robust to both messier real world data and also more robust to being used by less expert users. But that is not the only reason. The other reason is that the time is right. Because of the use of computers for closed loop business data storage and generation there now exists large quantities of data that is available to users. IF there were no data - there would be no interest in mining it. Likewise the fact that computer hardware has dramatically upped the ante by several orders of magnitude in storing and processing the data makes some of the most powerful data mining techniques feasible today.

The bottom line though, from an academic standpoint at least, is that there is little practical difference between a statistical technique and a classical data mining technique. Hence we have included a description of some of the most useful in this section.

What is statistics?

Statistics is a branch of mathematics concerning the collection and the description of data. Usually statistics is considered to be one of those scary topics in college right up there with chemistry and

physics. However, statistics is probably a much friendlier branch of mathematics because it really can be used every day. Statistics was in fact born from very humble beginnings of real world problems from business, biology, and gambling!

Knowing statistics in your everyday life will help the average business person make better decisions by allowing them to figure out risk and uncertainty when all the facts either aren't known or can't be collected. Even with all the data stored in the largest of data warehouses business decisions still just become more informed guesses. The more and better the data and the better the understanding of statistics the better the decision that can be made.

Statistics has been around for a long time easily a century and arguably many centuries when the ideas of probability began to gel. It could even be argued that the data collected by the ancient Egyptians, Babylonians, and Greeks were all statistics long before the field was officially recognized. Today data mining has been defined independently of statistics though "mining data" for patterns and predictions is really what statistics is all about. Some of the techniques that are classified under data mining such as CHAID and CART really grew out of the statistical profession more than anywhere else, and the basic ideas of probability, independence and causality and overfitting are the foundation on which both data mining and statistics are built.

Data, counting and probability

One thing that is always true about statistics is that there is always data involved, and usually enough data so that the average person cannot keep track of all the data in their heads. This is certainly more true today than it was when the basic ideas of probability and statistics were being formulated and refined early this century. Today people have to deal with up to terabytes of data and have to make sense of it and glean the important patterns from it. Statistics can help greatly in this process by helping to answer several important questions about your data:

- What patterns are there in my database?
- What is the chance that an event will occur?
- Which patterns are significant?
- What is a high level summary of the data that gives me some idea of what is contained in my database?

Certainly statistics can do more than answer these questions but for most people today these are the questions that statistics can help answer. Consider for example that a large part of statistics is concerned with summarizing data, and more often than not, this summarization has to do with counting. One of the great values of statistics is in presenting a high level view of the database that provides some useful information without requiring every record to be understood in detail. This aspect of statistics is the part that people run into every day when they read the daily newspaper and see, for example, a pie chart reporting the number of US citizens of different eye colors, or the average number of annual doctor visits for people of different ages. Statistics at this level is used in the reporting of important information from which people may be able to make useful decisions. There are many different parts of statistics but the idea of collecting data and counting it is often at the base of even these more sophisticated techniques. The first step then in understanding statistics is to

understand how the data is collected into a higher level form - one of the most notable ways of doing this is with the histogram.

Histograms

One of the best ways to summarize data is to provide a histogram of the data. In the simple example database shown in Table 1.1 we can create a histogram of eye color by counting the number of occurrences of different colors of eyes in our database. For this example database of 10 records this is fairly easy to do and the results are only slightly more interesting than the database itself. However, for a database of many more records this is a very useful way of getting a high level understanding of the database.

ID	Name	Prediction	Age	Balance	Income	Eyes	Gender
1	Amy	No	62	$0	Medium	Brown	F
2	Al	No	53	$1,800	Medium	Green	M
3	Betty	No	47	$16,543	High	Brown	F
4	Bob	Yes	32	$45	Medium	Green	M
5	Carla	Yes	21	$2,300	High	Blue	F
6	Carl	No	27	$5,400	High	Brown	M
7	Donna	Yes	50	$165	Low	Blue	F
8	Don	Yes	46	$0	High	Blue	M
9	Edna	Yes	27	$500	Low	Blue	F
10	Ed	No	68	$1,200	Low	Blue	M

Table 1.1 *An Example Database of Customers with Different Predictor Types*

This histogram shown in figure 1.1 depicts a simple predictor (eye color) which will have only a few different values no matter if there are 100 customer records in the database or 100 million. There are, however, other predictors that have many more distinct values and can create a much more complex histogram. Consider, for instance, the histogram of ages of the customers in the population. In this case the histogram can be more complex but can also be enlightening. Consider if you found that the histogram of your customer data looked as it does in figure 1.2.

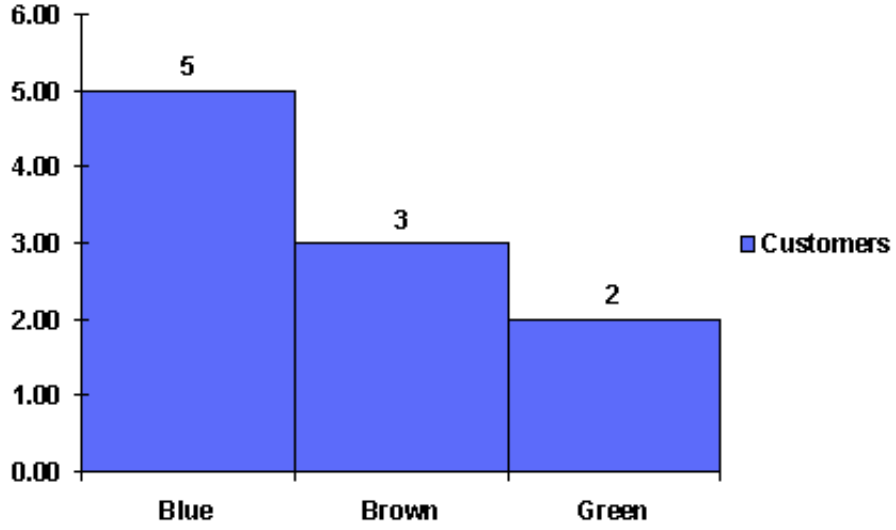

Figure 1.1 *This histogram shows the number of customers with various eye colors. This summary can quickly show important information about the database such as that blue eyes are the most frequent.*

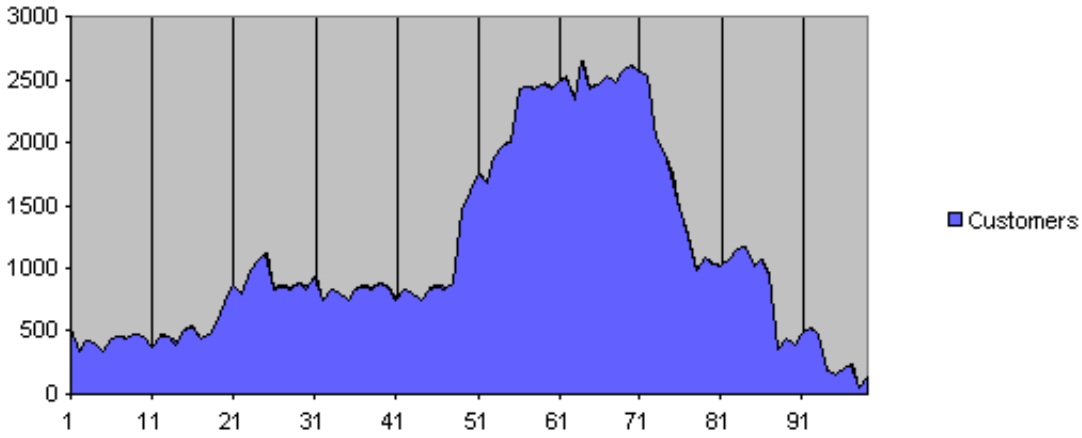

Figure 1.2 *This histogram shows the number of customers of different ages and quickly tells the viewer that the majority of customers are over the age of 50.*

By looking at this second histogram the viewer is in many ways looking at all of the data in the database for a particular predictor or data column. By looking at this histogram it is also possible to build an intuition about other important factors. Such as the average age of the population, the maximum and minimum age. All of which are important. These values are called summary statistics. Some of the most frequently used summary statistics include:

- Max - the maximum value for a given predictor.
- Min - the minimum value for a given predictor.

- Mean - the average value for a given predictor.
- Median - the value for a given predictor that divides the database as nearly as possible into two databases of equal numbers of records.
- Mode - the most common value for the predictor.
- Variance - the measure of how spread out the values are from the average value.

When there are many values for a given predictor the histogram begins to look smoother and smoother (compare the difference between the two histograms above). Sometimes the shape of the distribution of data can be calculated by an equation rather than just represented by the histogram. This is what is called a data distribution. Like a histogram a data distribution can be described by a variety of statistics. In classical statistics the belief is that there is some "true" underlying shape to the data distribution that would be formed if all possible data was collected. The shape of the data distribution can be calculated for some simple examples. The statistician's job then is to take the limited data that may have been collected and from that make their best guess at what the "true" or at least most likely underlying data distribution might be.

Many data distributions are well described by just two numbers, the mean and the variance. The mean is something most people are familiar with, the variance, however, can be problematic. The easiest way to think about it is that it measures the average distance of each predictor value from the mean value over all the records in the database. If the variance is high it implies that the values are all over the place and very different. If the variance is low most of the data values are fairly close to the mean. To be precise the actual definition of the variance uses the square of the distance rather than the actual distance from the mean and the average is taken by dividing the squared sum by one less than the total number of records. In terms of prediction a user could make some guess at the value of a predictor without knowing anything else just by knowing the mean and also gain some basic sense of how variable the guess might be based on the variance.

Statistics for Prediction

In this book the term "prediction" is used for a variety of types of analysis that may elsewhere be more precisely called regression. We have done so in order to simplify some of the concepts and to emphasize the common and most important aspects of predictive modeling. Nonetheless regression is a powerful and commonly used tool in statistics and it will be discussed here.

Linear regression

In statistics prediction is usually synonymous with regression of some form. There are a variety of different types of regression in statistics but the basic idea is that a model is created that maps values from predictors in such a way that the lowest error occurs in making a prediction. The simplest form of regression is simple linear regression that just contains one predictor and a prediction. The relationship between the two can be mapped on a two dimensional space and the records plotted for the prediction values along the Y axis and the predictor values along the X axis. The simple linear regression model then could be viewed as the line that minimized the error rate between the actual prediction value and the point on the line (the prediction from the model). Graphically this would look as it does in Figure 1.3. The simplest form of regression seeks to build a predictive model that is

a line that maps between each predictor value to a prediction value. Of the many possible lines that could be drawn through the data the one that minimizes the distance between the line and the data points is the one that is chosen for the predictive model.

On average if you guess the value on the line it should represent an acceptable compromise amongst all the data at that point giving conflicting answers. Likewise if there is no data available for a particular input value the line will provide the best guess at a reasonable answer based on similar data.

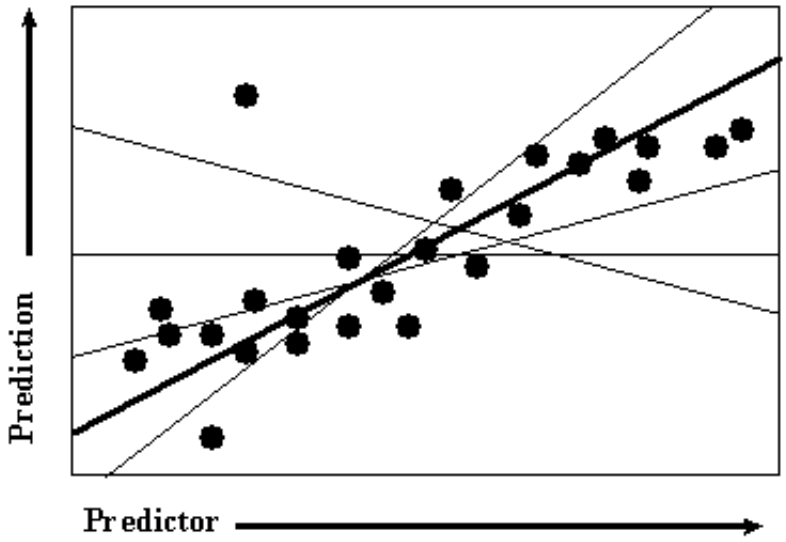

Figure 1.3 *Linear regression is similar to the task of finding the line that minimizes the total distance to a set of data.*

The predictive model is the line shown in Figure 1.3. The line will take a given value for a predictor and map it into a given value for a prediction. The actual equation would look something like: Prediction = a + b * Predictor. Which is just the equation for a line Y = a + bX. As an example for a bank the predicted average consumer bank balance might equal $1,000 + 0.01 * customer's annual income. The trick, as always with predictive modeling, is to find the model that best minimizes the error. The most common way to calculate the error is the square of the difference between the predicted value and the actual value. Calculated this way points that are very far from the line will have a great effect on moving the choice of line towards themselves in order to reduce the error. The values of a and b in the regression equation that minimize this error can be calculated directly from the data relatively quickly.

What if the pattern in my data doesn't look like a straight line?

Regression can become more complicated than the simple linear regression we've introduced so far. It can get more complicated in a variety of different ways in order to better model particular database problems. There are, however, three main modifications that can be made:

1. More predictors than just one can be used.

2. Transformations can be applied to the predictors.

3. Predictors can be multiplied together and used as terms in the equation.

4. Modifications can be made to accommodate response predictions that just have yes/no or 0/1 values.

Adding more predictors to the linear equation can produce more complicated lines that take more information into account and hence make a better prediction. This is called multiple linear regression and might have an equation like the following if 5 predictors were used (X1, X2, X3, X4, X5):

$$Y = a + b1(X1) + b2(X2) + b3(X3) + b4(X4) + b5(X5)$$

This equation still describes a line but it is now a line in a6 dimensional space rather than the two dimensional space.

By transforming the predictors by squaring, cubing or taking their square root it is possible to use the same general regression methodology and now create much more complex models that are no longer simple shaped like lines. This is called non-linear regression. A model of just one predictor might look like this: $Y = a + b1(X1) + b2 (X12)$. In many real world cases analysts will perform a wide variety of transformations on their data just to try them out. If they do not contribute to a useful model their coefficients in the equation will tend toward zero and then they can be removed. The other transformation of predictor values that is often performed is multiplying them together. For instance a new predictor created by dividing hourly wage by the minimum wage might be a much more effective predictor than hourly wage by itself.

When trying to predict a customer response that is just yes or no (e.g. they bought the product or they didn't or they defaulted or they didn't) the standard form of a line doesn't work. Since there are only two possible values to be predicted it is relatively easy to fit a line through them. However, that model would be the same no matter what predictors were being used or what particular data was being used. Typically in these situations a transformation of the prediction values is made in order to provide a better predictive model. This type of regression is called logistic regression and because so many business problems are response problems, logistic regression is one of the most widely used statistical techniques for creating predictive models.

1.3. Nearest Neighbor

Clustering and the Nearest Neighbor prediction technique are among the oldest techniques used in data mining. Most people have an intuition that they understand what clustering is - namely that like records are grouped or clustered together. Nearest neighbor is a prediction technique that is quite similar to clustering - its essence is that in order to predict what a prediction value is in one record look for records with similar predictor values in the historical database and use the prediction value from the record that it "nearest" to the unclassified record.

A simple example of clustering

A simple example of clustering would be the clustering that most people perform when they do the laundry - grouping the permanent press, dry cleaning, whites and brightly colored clothes is important because they have similar characteristics. And it turns out they have important attributes in common about the way they behave (and can be ruined) in the wash. To "cluster" your laundry

most of your decisions are relatively straightforward. There are of course difficult decisions to be made about which cluster your white shirt with red stripes goes into (since it is mostly white but has some color and is permanent press). When clustering is used in business the clusters are often much more dynamic - even changing weekly to monthly and many more of the decisions concerning which cluster a record falls into can be difficult.

A simple example of nearest neighbor

A simple example of the nearest neighbor prediction algorithm is that if you look at the people in your neighborhood (in this case those people that are in fact geographically near to you). You may notice that, in general, you all have somewhat similar incomes. Thus if your neighbor has an income greater than $100,000 chances are good that you too have a high income. Certainly the chances that you have a high income are greater when all of your neighbors have incomes over $100,000 than if all of your neighbors have incomes of $20,000. Within your neighborhood there may still be a wide variety of incomes possible among even your "closest" neighbors but if you had to predict someone's income based on only knowing their neighbors you're best chance of being right would be to predict the incomes of the neighbors who live closest to the unknown person.

The nearest neighbor prediction algorithm works in very much the same way except that "nearness" in a database may consist of a variety of factors not just where the person lives. It may, for instance, be far more important to know which school someone attended and what degree they attained when predicting income. The better definition of "near" might in fact be other people that you graduated from college with rather than the people that you live next to.

Nearest Neighbor techniques are among the easiest to use and understand because they work in a way similar to the way that people think - by detecting closely matching examples. They also perform quite well in terms of automation, as many of the algorithms are robust with respect to dirty data and missing data. Lastly they are particularly adept at performing complex ROI calculations because the predictions are made at a local level where business simulations could be performed in order to optimize ROI. As they enjoy similar levels of accuracy compared to other techniques the measures of accuracy such as lift are as good as from any other.

How to use Nearest Neighbor for Prediction

One of the essential elements underlying the concept of clustering is that one particular object (whether they be cars, food or customers) can be closer to another object than can some third object. It is interesting that most people have an innate sense of ordering placed on a variety of different objects. Most people would agree that an apple is closer to an orange than it is to a tomato and that a Toyota Corolla is closer to a Honda Civic than to a Porsche. This sense of ordering on many different objects helps us place them in time and space and to make sense of the world. It is what allows us to build clusters - both in databases on computers as well as in our daily lives. This definition of nearness that seems to be ubiquitous also allows us to make predictions.

The nearest neighbor prediction algorithm simply stated is:

Objects that are "near" to each other will have similar prediction values as well. Thus if you know the prediction value of one of the objects you can predict it for it's nearest neighbors.

Where has the nearest neighbor technique been used in business?

One of the classical places that nearest neighbor has been used for prediction has been in text retrieval. The problem to be solved in text retrieval is one where the end user defines a document (e.g. Wall Street Journal article, technical conference paper etc.) that is interesting to them and they solicit the system to "find more documents like this one". Effectively defining a target of: "this is the interesting document" or "this is not interesting". The prediction problem is that only a very few of the documents in the database actually have values for this prediction field (namely only the documents that the reader has had a chance to look at so far). The nearest neighbor technique is used to find other documents that share important characteristics with those documents that have been marked as interesting.

Using nearest neighbor for stock market data

As with almost all prediction algorithms, nearest neighbor can be used in a variety of places. Its successful use is mostly dependent on the pre-formatting of the data so that nearness can be calculated and where individual records can be defined. In the text retrieval example this was not too difficult - the objects were documents. This is not always as easy as it is for text retrieval. Consider what it might be like in a time series problem - say for predicting the stock market. In this case the input data is just a long series of stock prices over time without any particular record that could be considered to be an object. The value to be predicted is just the next value of the stock price.

The way that this problem is solved for both nearest neighbor techniques and for some other types of prediction algorithms is to create training records by taking, for instance, 10 consecutive stock prices and using the first 9 as predictor values and the 10th as the prediction value. Doing things this way, if you had 100 data points in your time series you could create 10 different training records.

You could create even more training records than 10 by creating a new record starting at every data point. For instance in the you could take the first 10 data points and create a record. Then you could take the 10 consecutive data points starting at the second data point, then the 10 consecutive data point starting at the third data point. Even though some of the data points would overlap from one record to the next the prediction value would always be different. In our example of 100 initial data points 90 different training records could be created this way as opposed to the 10 training records created via the other method.

Why voting is better - K Nearest Neighbors

One of the improvements that is usually made to the basic nearest neighbor algorithm is to take a vote from the "K" nearest neighbors rather than just relying on the sole nearest neighbor to the unclassified record. In Figure 1.4 we can see that unclassified example C has a nearest neighbor that is a defaulter and yet is surrounded almost exclusively by records that are good credit risks. In this case the nearest neighbor to record C is probably an outlier - which may be incorrect data or some non-repeatable idiosyncrasy. In either case it is more than likely that C is a non-defaulter yet would be predicted to be a defaulter if the sole nearest neighbor were used for the prediction.

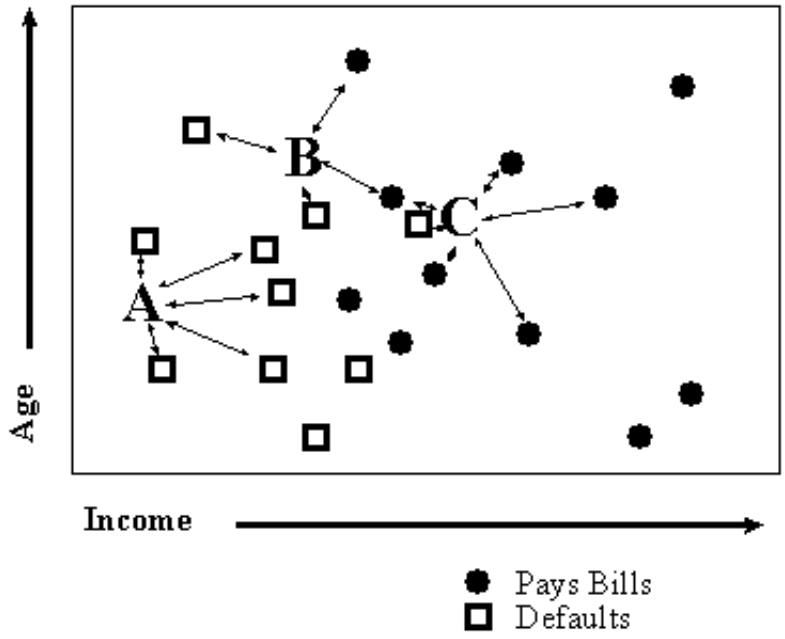

Figure 1.4 *The nearest neighbors are shown graphically for three unclassified records: A, B, and C.*

In cases like these a vote of the 9 or 15 nearest neighbors would provide a better prediction accuracy for the system than would just the single nearest neighbor. Usually this is accomplished by simply taking the majority or plurality of predictions from the K nearest neighbors if the prediction column is a binary or categorical or taking the average value of the prediction column from the K nearest neighbors.

How can the nearest neighbor tell you how confident it is in the prediction?

Another important aspect of any system that is used to make predictions is that the user be provided with, not only the prediction, but also some sense of the confidence in that prediction (e.g. the prediction is defaulter with the chance of being correct 60% of the time). The nearest neighbor algorithm provides this confidence information in a number of ways:

The distance to the nearest neighbor provides a level of confidence. If the neighbor is very close or an exact match then there is much higher confidence in the prediction than if the nearest record is a great distance from the unclassified record.

The degree of homogeneity amongst the predictions within the K nearest neighbors can also be used. If all the nearest neighbors make the same prediction then there is much higher confidence in the prediction than if half the records made one prediction and the other half made another prediction.

1.4. Clustering

Clustering for Clarity

Clustering is the method by which like records are grouped together. Usually this is done to give the

end user a high level view of what is going on in the database. Clustering is sometimes used to mean segmentation - which most marketing people will tell you is useful for coming up with a birds eye view of the business. Two of these clustering systems are the PRIZM™ system from Claritas corporation and MicroVision™ from Equifax corporation. These companies have grouped the population by demographic information into segments that they believe are useful for direct marketing and sales. To build these groupings they use information such as income, age, occupation, housing and race collect in the US Census. Then they assign memorable "nicknames" to the clusters. Some examples are shown in Table 1.2.

Name	Income	Age	Education	Vendor
Blue Blood Estates	Wealthy	35-54	College	Claritas Prizm™
Shotguns and Pickups	Middle	35-64	High School	Claritas Prizm™
Southside City	Poor	Mix	Grade School	Claritas Prizm™
Living Off the Land	Middle-Poor	School Age Families	Low	Equifax MicroVision™
University USA	Very low	Young - Mix	Medium to High	Equifax MicroVision™
Sunset Years	Medium	Seniors	Medium	Equifax MicroVision™

Table 1.2 *Some Commercially Available Cluster Tags*

This clustering information is then used by the end user to tag the customers in their database. Once this is done the business user can get a quick high level view of what is happening within the cluster. Once the business user has worked with these codes for some time they also begin to build intuitions about how these different customers clusters will react to the marketing offers particular to their business. For instance some of these clusters may relate to their business and some of them may not. But given that their competition may well be using these same clusters to structure their business and marketing offers it is important to be aware of how you customer base behaves in regard to these clusters.

Finding the ones that don't fit in - Clustering for Outliers

Sometimes clustering is performed not so much to keep records together as to make it easier to see when one record sticks out from the rest. For instance:

Most wine distributors selling inexpensive wine in Missouri and that ship a certain volume of product produce a certain level of profit. There is a cluster of stores that can be formed with these characteristics. One store stands out, however, as producing significantly lower profit. On closer examination it turns out that the distributor was delivering product to but not collecting payment from one of their customers.

A sale on men's suits is being held in all branches of a department store for southern California . All stores with these characteristics have seen at least a 100% jump in revenue since the start of the sale

except one. It turns out that this store had, unlike the others, advertised via radio rather than television.

How is clustering like the nearest neighbor technique?

The nearest neighbor algorithm is basically a refinement of clustering in the sense that they both use distance in some feature space to create either structure in the data or predictions. The nearest neighbor algorithm is a refinement since part of the algorithm usually is a way of automatically determining the weighting of the importance of the predictors and how the distance will be measured within the feature space. Clustering is one special case of this where the importance of each predictor is considered to be equivalent.

How to put clustering and nearest neighbor to work for prediction

To see clustering and nearest neighbor prediction in use let's go back to our example database and now look at it in two ways. First let's try to create our own clusters - which if useful we could use internally to help to simplify and clarify large quantities of data (and maybe if we did a very good job sell these new codes to other business users). Secondly let's try to create predictions based on the nearest neighbor.

First take a look at the data. How would you cluster the data in Table 1.3?

ID	Name	Prediction	Age	Balance	Income	Eyes	Gender
1	Amy	No	62	$0	Medium	Brown	F
2	Al	No	53	$1,800	Medium	Green	M
3	Betty	No	47	$16,543	High	Brown	F
4	Bob	Yes	32	$45	Medium	Green	M
5	Carla	Yes	21	$2,300	High	Blue	F
6	Carl	No	27	$5,400	High	Brown	M
7	Donna	Yes	50	$165	Low	Blue	F
8	Don	Yes	46	$0	High	Blue	M
9	Edna	Yes	27	$500	Low	Blue	F
10	Ed	No	68	$1,200	Low	Blue	M

Table 1.3 *A Simple Example Database*

If these were your friends rather than your customers (hopefully they could be both) and they were single, you might cluster them based on their compatibility with each other. Creating your own mini dating service. If you were a pragmatic person you might cluster your database as follows because you think that marital happiness is mostly dependent on financial compatibility and create three clusters as shown in Table 1.4.

ID	Name	Prediction	Age	Balance	Income	Eyes	Gender

3	Betty	No	47	$16,543	High	Brown	F
5	Carla	Yes	21	$2,300	High	Blue	F
6	Carl	No	27	$5,400	High	Brown	M
8	Don	Yes	46	$0	High	Blue	M

1	Amy	No	62	$0	Medium	Brown	F
2	Al	No	53	$1,800	Medium	Green	M
4	Bob	Yes	32	$45	Medium	Green	M

7	Donna	Yes	50	$165	Low	Blue	F
9	Edna	Yes	27	$500	Low	Blue	F
10	Ed	No	68	$1,200	Low	Blue	M

Table 1.4. *A Simple Clustering of the Example Database*

Is the another "correct" way to cluster?

If on the other hand you are more of a romantic you might note some incompatibilities between 46 year old Don and 21 year old Carla (even though they both make very good incomes). You might instead consider age and some physical characteristics to be most important in creating clusters of friends. Another way you could cluster your friends would be based on their ages and on the color of their eyes. This is shown in Table 1.5. Here three clusters are created where each person in the cluster is about the same age and some attempt has been made to keep people of like eye color together in the same cluster.

ID	Name	Prediction	Age	Balance	Income	Eyes	Gender

5	Carla	Yes	21	$2,300	High	Blue	F
9	Edna	Yes	27	$500	Low	Blue	F
6	Carl	No	27	$5,400	High	Brown	M
4	Bob	Yes	32	$45	Medium	Green	M

8	Don	Yes	46	$0	High	Blue	M
7	Donna	Yes	50	$165	Low	Blue	F
10	Ed	No	68	$1,200	Low	Blue	M

3	Betty	No	47	$16,543	High	Brown	F
2	Al	No	53	$1,800	Medium	Green	M
1	Amy	No	62	$0	Medium	Brown	F

Table 1.5 *A More "Romantic" Clustering of the Example Database to Optimize for Your Dating Service*

There is no best way to cluster.

This example, though simple, points up some important questions about clustering. For instance: Is it

possible to say whether the first clustering that was performed above (by financial status) was better or worse than the second clustering (by age and eye color)? Probably not since the clusters were constructed for no particular purpose except to note similarities between some of the records and that the view of the database could be somewhat simplified by using clusters. But even the differences that were created by the two different clusterings were driven by slightly different motivations (financial vs. Romantic). In general the reasons for clustering are just this ill defined because clusters are used more often than not for exploration and summarization as much as they are used for prediction.

How are tradeoffs made when determining which records fall into which clusters?

Notice that for the first clustering example there was a pretty simple rule by which the records could be broken up into clusters - namely by income. In the second clustering example there were less clear dividing lines since two predictors were used to form the clusters (age and eye color). Thus the first cluster is dominated by younger people with somewhat mixed eye colors whereas the latter two clusters have a mix of older people where eye color has been used to separate them out (the second cluster is entirely blue eyed people). In this case these tradeoffs were made arbitrarily but when clustering much larger numbers of records these tradeoffs are explicitly defined by the clustering algorithm.

Clustering is the happy medium between homogeneous clusters and the fewest number of clusters.

In the best possible case clusters would be built where all records within the cluster had identical values for the particular predictors that were being clustered on. This would be the optimum in creating a high level view since knowing the predictor values for any member of the cluster would mean knowing the values for every member of the cluster no matter how large the cluster was. Creating homogeneous clusters where all values for the predictors are the same is difficult to do when there are many predictors and/or the predictors have many different values (high cardinality).

It is possible to guarantee that homogeneous clusters are created by breaking apart any cluster that is inhomogeneous into smaller clusters that are homogeneous. In the extreme, though, this usually means creating clusters with only one record in them which usually defeats the original purpose of the clustering. For instance in our 10 record database above 10 perfectly homogeneous clusters could be formed of 1 record each, but not much progress would have been made in making the original database more understandable.

The second important constraint on clustering is then that a reasonable number of clusters are formed. Where, again, reasonable is defined by the user but is difficult to quantify beyond that except to say that just one cluster is unacceptable (too much generalization) and that as many clusters and original records is also unacceptable Many clustering algorithms either let the user choose the number of clusters that they would like to see created from the database or they provide the user a "knob" by which they can create fewer or greater numbers of clusters interactively after the clustering has been performed.

What is the difference between clustering and nearest neighbor prediction?

The main distinction between clustering and the nearest neighbor technique is that clustering is what

is called an unsupervised learning technique and nearest neighbor is generally used for prediction or a supervised learning technique. Unsupervised learning techniques are unsupervised in the sense that when they are run there is not particular reason for the creation of the models the way there is for supervised learning techniques that are trying to perform prediction. In prediction, the patterns that are found in the database and presented in the model are always the most important patterns in the database for performing some particular prediction. In clustering there is no particular sense of why certain records are near to each other or why they all fall into the same cluster. Some of the differences between clustering and nearest neighbor prediction can be summarized in Table 1.6.

Nearest Neighbor	Clustering
Used for prediction as well as consolidation.	Used mostly for consolidating data into a high-level view and general grouping of records into like behaviors.
Space is defined by the problem to be solved (supervised learning).	Space is defined as default n-dimensional space, or is defined by the user, or is a predefined space driven by past experience (unsupervised learning).
Generally only uses distance metrics to determine nearness.	Can use other metrics besides distance to determine nearness of two records - for example linking two points together.

Table 1.6 *Some of the Differences Between the Nearest-Neighbor Data Mining Technique and Clustering*

What is an n-dimensional space? Do I really need to know this?

When people talk about clustering or nearest neighbor prediction they will often talk about a "space" of "N" dimensions. What they mean is that in order to define what is near and what is far away it is helpful to have a "space" defined where distance can be calculated. Generally these spaces behave just like the three dimensional space that we are familiar with where distance between objects is defined by euclidean distance (just like figuring out the length of a side in a triangle).

What goes for three dimensions works pretty well for more dimensions as well. Which is a good thing since most real world problems consists of many more than three dimensions. In fact each predictor (or database column) that is used can be considered to be a new dimension. In the example above the five predictors: age, income, balance, eyes and gender can all be construed to be dimensions in an n dimensional space where n, in this case, equal 5. It is sometimes easier to think about these and other data mining algorithms in terms of n-dimensional spaces because it allows for some intuitions to be used about how the algorithm is working.

Moving from three dimensions to five dimensions is not too large a jump but there are also spaces in real world problems that are far more complex. In the credit card industry credit card issuers typically have over one thousand predictors that could be used to create an n-dimensional space. For text retrieval (e.g. finding useful Wall Street Journal articles from a large database, or finding useful web

sites on the internet) the predictors (and hence the dimensions) are typically words or phrases that are found in the document records. In just one year of the Wall Street Journal there are more than 50,000 different words used - which translates to a 50,000 dimensional space in which nearness between records must be calculated.

How is the space for clustering and nearest neighbor defined?

For clustering the n-dimensional space is usually defined by assigning one predictor to each dimension. For the nearest neighbor algorithm predictors are also mapped to dimensions but then those dimensions are literally stretched or compressed based on how important the particular predictor is in making the prediction. The stretching of a dimension effectively makes that dimension (and hence predictor) more important than the others in calculating the distance.

For instance if you are a mountain climber and someone told you that you were 2 miles from your destination the distance is the same whether it's 1 mile north and 1 mile up the face of the mountain or 2 miles north on level ground but clearly the former route is much different from the latter. The distance traveled straight upward is the most important if figuring out how long it will really take to get to the destination and you would probably like to consider this "dimension" to be more important than the others. In fact you, as a mountain climber, could "weight" the importance of the vertical dimension in calculating some new distance by reasoning that every mile upward is equivalent to 10 miles on level ground.

If you used this rule of thumb to weight the importance of one dimension over the other it would be clear that in one case you were much "further away" from your destination ("11 miles") than in the second ("2 miles"). In the next section we'll show how the nearest neighbor algorithm uses distance measure that similarly weight the important dimensions more heavily when calculating a distance measure.

Hierarchical and Non-Hierarchical Clustering

There are two main types of clustering techniques, those that create a hierarchy of clusters and those that do not. The hierarchical clustering techniques create a hierarchy of clusters from small to big. The main reason for this is that, as was already stated, clustering is an unsupervised learning technique, and as such, there is no absolutely correct answer. For this reason and depending on the particular application of the clustering, fewer or greater numbers of clusters may be desired. With a hierarchy of clusters defined it is possible to choose the number of clusters that are desired. At the extreme it is possible to have as many clusters as there are records in the database. In this case the records within the cluster are optimally similar to each other (since there is only one) and certainly different from the other clusters. But of course such a clustering technique misses the point in the sense that the idea of clustering is to find useful patters in the database that summarize it and make it easier to understand. Any clustering algorithm that ends up with as many clusters as there are records has not helped the user understand the data any better. Thus one of the main points about clustering is that there be many fewer clusters than there are original records. Exactly how many clusters should be formed is a matter of interpretation. The advantage of hierarchical clustering methods is that they allow the end user to choose from either many clusters or only a few.

The hierarchy of clusters is usually viewed as a tree where the smallest clusters merge together to create the next highest level of clusters and those at that level merge together to create the next

highest level of clusters. Figure 1.5 below shows how several clusters might form a hierarchy. When a hierarchy of clusters like this is created the user can determine what the right number of clusters is that adequately summarizes the data while still providing useful information (at the other extreme a single cluster containing all the records is a great summarization but does not contain enough specific information to be useful).

This hierarchy of clusters is created through the algorithm that builds the clusters. There are two main types of hierarchical clustering algorithms:

- Agglomerative - Agglomerative clustering techniques start with as many clusters as there are records where each cluster contains just one record. The clusters that are nearest each other are merged together to form the next largest cluster. This merging is continued until a hierarchy of clusters is built with just a single cluster containing all the records at the top of the hierarchy.

- Divisive - Divisive clustering techniques take the opposite approach from agglomerative techniques. These techniques start with all the records in one cluster and then try to split that cluster into smaller pieces and then in turn to try to split those smaller pieces.

Of the two the agglomerative techniques are the most commonly used for clustering and have more algorithms developed for them. We'll talk about these in more detail in the next section. The non-hierarchical techniques in general are faster to create from the historical database but require that the user make some decision about the number of clusters desired or the minimum "nearness" required for two records to be within the same cluster. These non-hierarchical techniques often times are run multiple times starting off with some arbitrary or even random clustering and then iteratively improving the clustering by shuffling some records around. Or these techniques some times create clusters that are created with only one pass through the database adding records to existing clusters when they exist and creating new clusters when no existing cluster is a good candidate for the given record. Because the definition of which clusters are formed can depend on these initial choices of which starting clusters should be chosen or even how many clusters these techniques can be less repeatable than the hierarchical techniques and can sometimes create either too many or too few clusters because the number of clusters is predetermined by the user not determined solely by the patterns inherent in the database.

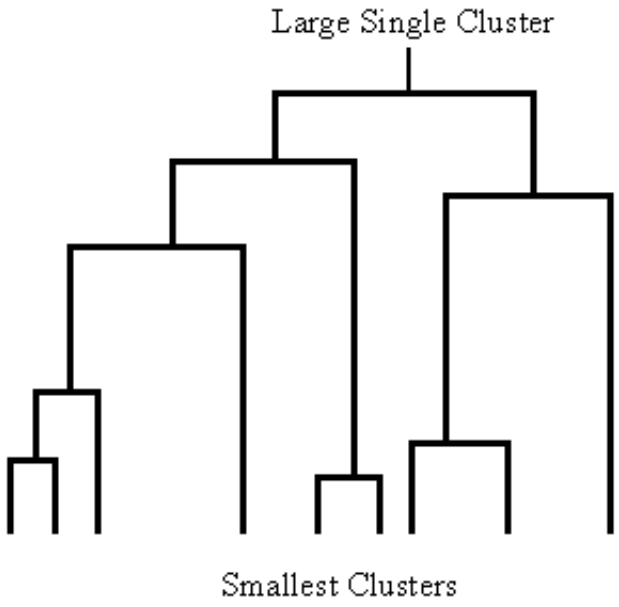

Figure 1.5 *Diagram showing a hierarchy of clusters. Clusters at the lowest level are merged together to form larger clusters at the next level of the hierarchy.*

Non-Hierarchical Clustering

There are two main non-hierarchical clustering techniques. Both of them are very fast to compute on the database but have some drawbacks. The first are the single pass methods. They derive their name from the fact that the database must only be passed through once in order to create the clusters (i.e. each record is only read from the database once). The other class of techniques are called reallocation methods. They get their name from the movement or "reallocation" of records from one cluster to another in order to create better clusters. The reallocation techniques do use multiple passes through the database but are relatively fast in comparison to the hierarchical techniques.

Some techniques allow the user to request the number of clusters that they would like to be pulled out of the data. Predefining the number of clusters rather than having them driven by the data might seem to be a bad idea as there might be some very distinct and observable clustering of the data into a certain number of clusters which the user might not be aware of.

For instance the user may wish to see their data broken up into 10 clusters but the data itself partitions very cleanly into 13 clusters. These non-hierarchical techniques will try to shoe horn these extra three clusters into the existing 10 rather than creating 13 which best fit the data. The saving grace for these methods, however, is that, as we have seen, there is no one right answer for how to cluster so it is rare that by arbitrarily predefining the number of clusters that you would end up with the wrong answer. One of the advantages of these techniques is that often times the user does have some predefined level of summarization that they are interested in (e.g. "25 clusters is too confusing, but 10 will help to give me an insight into my data"). The fact that greater or fewer numbers of clusters would better match the data is actually of secondary importance.

Hierarchical Clustering

Hierarchical clustering has the advantage over non-hierarchical techniques in that the clusters are defined solely by the data (not by the users predetermining the number of clusters) and that the number of clusters can be increased or decreased by simple moving up and down the hierarchy.

The hierarchy is created by starting either at the top (one cluster that includes all records) and subdividing (divisive clustering) or by starting at the bottom with as many clusters as there are records and merging (agglomerative clustering). Usually the merging and subdividing are done two clusters at a time.

The main distinction between the techniques is their ability to favor long, scraggly clusters that are linked together record by record, or to favor the detection of the more classical, compact or spherical cluster that was shown at the beginning of this section. It may seem strange to want to form these long snaking chain like clusters, but in some cases they are the patters that the user would like to have detected in the database. These are the times when the underlying space looks quite different from the spherical clusters and the clusters that should be formed are not based on the distance from the center of the cluster but instead based on the records being "linked" together. Consider the example shown in Figure 1.6 or in Figure 1.7. In these cases there are two clusters that are not very spherical in shape but could be detected by the single link technique.

When looking at the layout of the data in Figure1.6 there appears to be two relatively flat clusters running parallel to each along the income axis. Neither the complete link nor Ward's method would, however, return these two clusters to the user. These techniques rely on creating a "center" for each cluster and picking these centers so that they average distance of each record from this center is minimized. Points that are very distant from these centers would necessarily fall into a different cluster.

What makes these clusters "visible" in this simple two dimensional space is the fact that each point in a cluster is tightly linked to some other point in the cluster. For the two clusters we see the maximum distance between the nearest two points within a cluster is less than the minimum distance of the nearest two points in different clusters. That is to say that for any point in this space, the nearest point to it is always going to be another point in the same cluster. Now the center of gravity of a cluster could be quite distant from a given point but that every point is linked to every other point by a series of small distances.

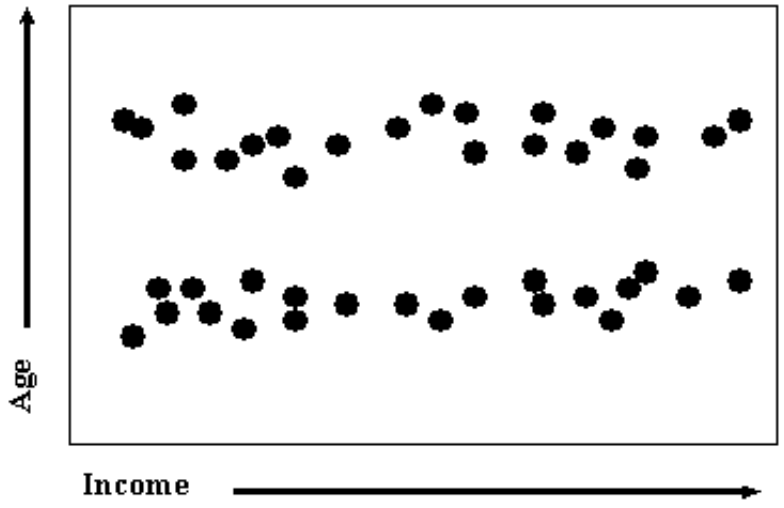

Figure 1.6 *an example of elongated clusters which would not be recovered by the complete link or Ward's methods but would be by the single-link method.*

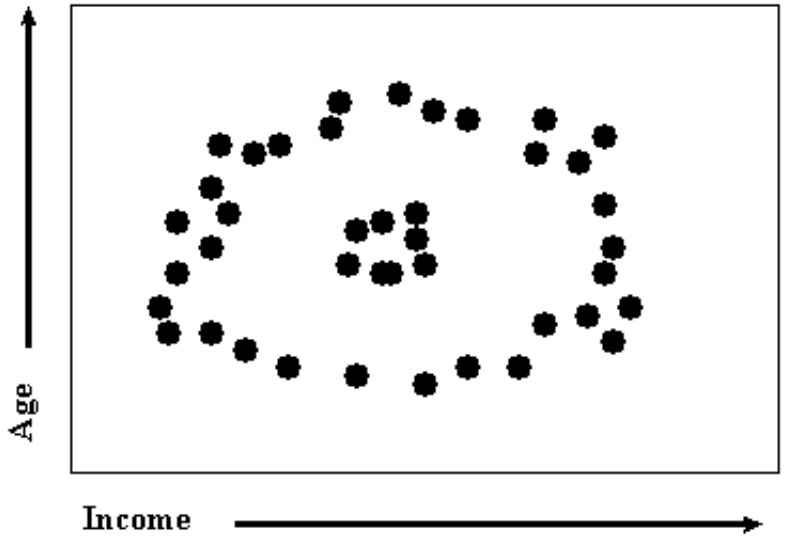

Figure 1.7 *An example of nested clusters which would not be recovered by the complete link or Ward's methods but would be by the single-link method.*

1.5. Choosing the Classics

There is no particular rule that would tell you when to choose a particular technique over another one. Sometimes those decisions are made relatively arbitrarily based on the availability of data mining analysts who are most experienced in one technique over another. And even choosing classical techniques over some of the newer techniques is more dependent on the availability of good tools and good analysts. Whichever techniques are chosen whether classical or next generation all of the techniques presented here have been available and tried for more than two decades. So even the next generation is a solid bet for implementation.

II. Next Generation Techniques: Trees, Networks and Rules

2.1. The Next Generation

The data mining techniques in this section represent the most often used techniques that have been developed over the last two decades of research. They also represent the vast majority of the techniques that are being spoken about when data mining is mentioned in the popular press. These techniques can be used for either discovering new information within large databases or for building predictive models. Though the older decision tree techniques such as CHAID are currently highly used the new techniques such as CART are gaining wider acceptance.

2.2. Decision Trees

What is a Decision Tree?

A decision tree is a predictive model that, as its name implies, can be viewed as a tree. Specifically each branch of the tree is a classification question and the leaves of the tree are partitions of the dataset with their classification. For instance if we were going to classify customers who churn (don't renew their phone contracts) in the Cellular Telephone Industry a decision tree might look something like that found in Figure 2.1.

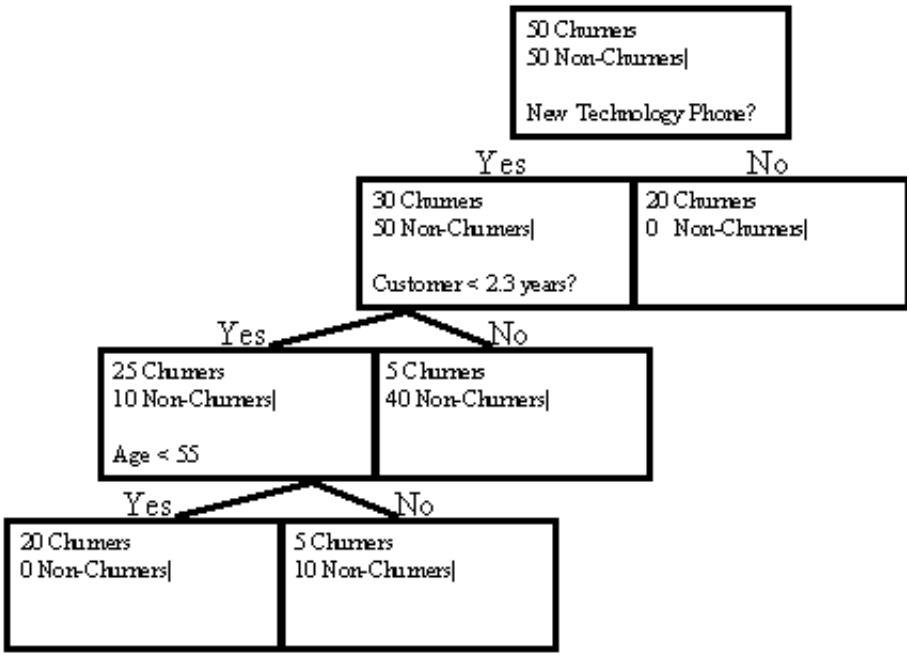

Figure 2.1 *A decision tree is a predictive model that makes a prediction on the basis of a series of decision much like the game of 20 questions.*

You may notice some interesting things about the tree:

- It divides up the data on each branch point without losing any of the data (the number of total records in a given parent node is equal to the sum of the records contained in its two children).
- The number of churners and non-churners is conserved as you move up or down the tree
- It is pretty easy to understand how the model is being built (in contrast to the models from neural networks or from standard statistics).
- It would also be pretty easy to use this model if you actually had to target those customers that are likely to churn with a targeted marketing offer.

You may also build some intuitions about your customer base. E.g. "customers who have been with you for a couple of years and have up to date cellular phones are pretty loyal".

Viewing decision trees as segmentation with a purpose

From a business perspective decision trees can be viewed as creating a segmentation of the original dataset (each segment would be one of the leaves of the tree). Segmentation of customers, products,

and sales regions is something that marketing managers have been doing for many years. In the past this segmentation has been performed in order to get a high level view of a large amount of data - with no particular reason for creating the segmentation except that the records within each segmentation were somewhat similar to each other.

In this case the segmentation is done for a particular reason - namely for the prediction of some important piece of information. The records that fall within each segment fall there because they have similarity with respect to the information being predicted - not just that they are similar - without similarity being well defined. These predictive segments that are derived from the decision tree also come with a description of the characteristics that define the predictive segment. Thus the decision trees and the algorithms that create them may be complex, the results can be presented in an easy to understand way that can be quite useful to the business user.

Applying decision trees to Business

Because of their tree structure and ability to easily generate rules decision trees are the favored technique for building understandable models. Because of this clarity they also allow for more complex profit and ROI models to be added easily in on top of the predictive model. For instance once a customer population is found with high predicted likelihood to attrite a variety of cost models can be used to see if an expensive marketing intervention should be used because the customers are highly valuable or a less expensive intervention should be used because the revenue from this sub-population of customers is marginal.

Because of their high level of automation and the ease of translating decision tree models into SQL for deployment in relational databases the technology has also proven to be easy to integrate with existing IT processes, requiring little preprocessing and cleansing of the data, or extraction of a special purpose file specifically for data mining.

Where can decision trees be used?

Decision trees are data mining technology that has been around in a form very similar to the technology of today for almost twenty years now and early versions of the algorithms date back in the 1960s. Often times these techniques were originally developed for statisticians to automate the process of determining which fields in their database were actually useful or correlated with the particular problem that they were trying to understand. Partially because of this history, decision tree algorithms tend to automate the entire process of hypothesis generation and then validation much more completely and in a much more integrated way than any other data mining techniques. They are also particularly adept at handling raw data with little or no pre-processing. Perhaps also because they were originally developed to mimic the way an analyst interactively performs data mining they provide a simple to understand predictive model based on rules (such as “90% of the time credit card customers of less than 3 months who max out their credit limit are going to default on their credit card loan.”).

Because decision trees score so highly on so many of the critical features of data mining they can be used in a wide variety of business problems for both exploration and for prediction. They have been used for problems ranging from credit card attrition prediction to time series prediction of the exchange rate of different international currencies. There are also some problems where decision trees will not do as well. Some very simple problems where the prediction is just a simple multiple of

the predictor can be solved much more quickly and easily by linear regression. Usually the models to be built and the interactions to be detected are much more complex in real world problems and this is where decision trees excel.

Using decision trees for Exploration

The decision tree technology can be used for exploration of the dataset and business problem. This is often done by looking at the predictors and values that are chosen for each split of the tree. Often times these predictors provide usable insights or propose questions that need to be answered. For instance if you ran across the following in your database for cellular phone churn you might seriously wonder about the way your telesales operators were making their calls and maybe change the way that they are compensated: "IF customer lifetime < 1.1 years AND sales channel = telesales THEN chance of churn is 65%.

Using decision trees for Data Preprocessing

Another way that the decision tree technology has been used is for preprocessing data for other prediction algorithms. Because the algorithm is fairly robust with respect to a variety of predictor types (e.g. number, categorical etc.) and because it can be run relatively quickly decision trees can be used on the first pass of a data mining run to create a subset of possibly useful predictors that can then be fed into neural networks, nearest neighbor and normal statistical routines - which can take a considerable amount of time to run if there are large numbers of possible predictors to be used in the model.

Decision tress for Prediction

Although some forms of decision trees were initially developed as exploratory tools to refine and preprocess data for more standard statistical techniques like logistic regression. They have also been used and more increasingly often being used for prediction. This is interesting because many statisticians will still use decision trees for exploratory analysis effectively building a predictive model as a by product but then ignore the predictive model in favor of techniques that they are most comfortable with. Sometimes veteran analysts will do this even excluding the predictive model when it is superior to that produced by other techniques. With a host of new products and skilled users now appearing this tendency to use decision trees only for exploration now seems to be changing.

The first step is Growing the Tree

The first step in the process is that of growing the tree. Specifically the algorithm seeks to create a tree that works as perfectly as possible on all the data that is available. Most of the time it is not possible to have the algorithm work perfectly. There is always noise in the database to some degree (there are variables that are not being collected that have an impact on the target you are trying to predict).

The name of the game in growing the tree is in finding the best possible question to ask at each branch point of the tree. At the bottom of the tree you will come up with nodes that you would like to be all of one type or the other. Thus the question: "Are you over 40?" probably does not sufficiently distinguish between those who are churners and those who are not - let's say it is 40%/ 60%. On the other hand there may be a series of questions that do quite a nice job in distinguishing

those cellular phone customers who will churn and those who won't. Maybe the series of questions would be something like: "Have you been a customer for less than a year, do you have a telephone that is more than two years old and were you originally landed as a customer via telesales rather than direct sales?" This series of questions defines a segment of the customer population in which 90% churn. These are then relevant questions to be asking in relation to predicting churn.

The difference between a good question and a bad question

The difference between a good question and a bad question has to do with how much the question can organize the data - or in this case, change the likelihood of a churner appearing in the customer segment. If we started off with our population being half churners and half non-churners then we would expect that a question that didn't organize the data to some degree into one segment that was more likely to churn than the other then it wouldn't be a very useful question to ask. On the other hand if we asked a question that was very good at distinguishing between churners and non-churners - say that split 100 customers into one segment of 50 churners and another segment of 50 non-churners then this would be considered to be a good question. In fact it had decreased the "disorder" of the original segment as much as was possible.

The process in decision tree algorithms is very similar when they build trees. These algorithms look at all possible distinguishing questions that could possibly break up the original training dataset into segments that are nearly homogeneous with respect to the different classes being predicted. Some decision tree algorithms may use heuristics in order to pick the questions or even pick them at random. CART picks the questions in a very unsophisticated way: It tries them all. After it has tried them all CART picks the best one uses it to split the data into two more organized segments and then again asks all possible questions on each of those new segments individually.

When does the tree stop growing?

If the decision tree algorithm just continued growing the tree like this it could conceivably create more and more questions and branches in the tree so that eventually there was only one record in the segment. To let the tree grow to this size is both computationally expensive but also unnecessary. Most decision tree algorithms stop growing the tree when one of three criteria are met:

- The segment contains only one record. (There is no further question that you could ask which could further refine a segment of just one.)
- All the records in the segment have identical characteristics. (There is no reason to continue asking further questions segmentation since all the remaining records are the same.)
- The improvement is not substantial enough to warrant making the split.

Why would a decision tree algorithm stop growing the tree if there wasn't enough data?

Consider the following example shown in Table 2.1 of a segment that we might want to split further which has just two examples. It has been created out of a much larger customer database by selecting only those customers aged 27 with blue eyes and salaries between $80,000 and $81,000.

Name	Age	Eyes	Salary	Churned?
Steve	27	Blue	$80,000	Yes
Alex	27	Blue	$80,000	No

Table 2.1 *Decision tree algorithm segment. This segment cannot be split further except by using the predictor "name".*

In this case all of the possible questions that could be asked about the two customers turn out to have the same value (age, eyes, salary) except for name. It would then be possible to ask a question like: "Is the customer's name Steve?" and create the segments which would be very good at breaking apart those who churned from those who did not:

The problem is that we all have an intuition that the name of the customer is not going to be a very good indicator of whether that customer churns or not. It might work well for this particular 2 record segment but it is unlikely that it will work for other customer databases or even the same customer database at a different time. This particular example has to do with overfitting the model - in this case fitting the model too closely to the idiosyncrasies of the training data. This can be fixed later on but clearly stopping the building of the tree short of either one record segments or very small segments in general is a good idea.

Decision trees aren't necessarily finished after the tree is grown.

After the tree has been grown to a certain size (depending on the particular stopping criteria used in the algorithm) the CART algorithm has still more work to do. The algorithm then checks to see if the model has been overfit to the data. It does this in several ways using a cross validation approach or a test set validation approach. Basically using the same mind numbingly simple approach it used to find the best questions in the first place - namely trying many different simpler versions of the tree on a held aside test set. The tree that does the best on the held aside data is selected by the algorithm as the best model. The nice thing about CART is that this testing and selection is all an integral part of the algorithm as opposed to the after the fact approach that other techniques use.

ID3 and an enhancement - C4.5

In the late 1970s J. Ross Quinlan introduced a decision tree algorithm named ID3. It was one of the first decision tree algorithms yet at the same time built solidly on work that had been done on inference systems and concept learning systems from that decade as well as the preceding decade. Initially ID3 was used for tasks such as learning good game playing strategies for chess end games. Since then ID3 has been applied to a wide variety of problems in both academia and industry and has been modified, improved and borrowed from many times over.

ID3 picks predictors and their splitting values based on the gain in information that the split or splits provide. Gain represents the difference between the amount of information that is needed to correctly make a prediction before a split is made and after the split has been made. If the amount of information required is much lower after the split is made then that split has decreased the disorder of the original single segment. Gain is defined as the difference between the entropy of the original segment and the accumulated entropies of the resulting split segments.

ID3 was later enhanced in the version called C4.5. C4.5 improves on ID3 in several important areas:

- predictors with missing values can still be used
- predictors with continuous values can be used
- pruning is introduced
- rule derivation

Many of these techniques appear in the CART algorithm plus some others so we will go through this introduction in the CART algorithm.

CART - Growing a forest and picking the best tree

CART stands for Classification and Regression Trees and is a data exploration and prediction algorithm developed by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone and is nicely detailed in their 1984 book "Classification and Regression Trees" ([Breiman, Friedman, Olshen and Stone 19 84)]. These researchers from Stanford University and the University of California at Berkeley showed how this new algorithm could be used on a variety of different problems from to the detection of Chlorine from the data contained in a mass spectrum.

Predictors are picked as they decrease the disorder of the data.

In building the CART tree each predictor is picked based on how well it teases apart the records with different predictions. For instance one measure that is used to determine whether a given split point for a give predictor is better than another is the entropy metric. The measure originated from the work done by Claude Shannon and Warren Weaver on information theory in 1949. They were concerned with how information could be efficiently communicated over telephone lines. Interestingly, their results also prove useful in creating decision trees.

CART Automatically Validates the Tree

One of the great advantages of CART is that the algorithm has the validation of the model and the discovery of the optimally general model built deeply into the algorithm. CART accomplishes this by building a very complex tree and then pruning it back to the optimally general tree based on the results of cross validation or test set validation. The tree is pruned back based on the performance of the various pruned version of the tree on the test set data. The most complex tree rarely fares the best on the held aside data as it has been overfitted to the training data. By using cross validation the tree that is most likely to do well on new, unseen data can be chosen.

CART Surrogates handle missing data

The CART algorithm is relatively robust with respect to missing data. If the value is missing for a particular predictor in a particular record that record will not be used in making the determination of the optimal split when the tree is being built. In effect CART will utilizes as much information as it has on hand in order to make the decision for picking the best possible split.

When CART is being used to predict on new data, missing values can be handled via surrogates. Surrogates are split values and predictors that mimic the actual split in the tree and can be used when

the data for the preferred predictor is missing. For instance though shoe size is not a perfect predictor of height it could be used as a surrogate to try to mimic a split based on height when that information was missing from the particular record being predicted with the CART model.

CHAID

Another equally popular decision tree technology to CART is CHAID or Chi-Square Automatic Interaction Detector. CHAID is similar to CART in that it builds a decision tree but it differs in the way that it chooses its splits. Instead of the entropy or Gini metrics for choosing optimal splits the technique relies on the chi square test used in contingency tables to determine which categorical predictor is furthest from independence with the prediction values.

Because CHAID relies on the contingency tables to form its test of significance for each predictor all predictors must either be categorical or be coerced into a categorical form via binning (e.g. break up possible people ages into 10 bins from 0-9, 10-19, 20-29 etc.). Though this binning can have deleterious consequences the actual accuracy performances of CART and CHAID have been shown to be comparable in real world direct marketing response models.

2.3. Neural Networks

What is a Neural Network?

When data mining algorithms are talked about these days most of the time people are talking about either decision trees or neural networks. Of the two neural networks have probably been of greater interest through the formative stages of data mining technology. As we will see neural networks do have disadvantages that can be limiting in their ease of use and ease of deployment, but they do also have some significant advantages. Foremost among these advantages is their highly accurate predictive models that can be applied across a large number of different types of problems.

To be more precise with the term “neural network” one might better speak of an “artificial neural network”. True neural networks are biological systems (a k a brains) that detect patterns, make predictions and learn. The artificial ones are computer programs implementing sophisticated pattern detection and machine learning algorithms on a computer to build predictive models from large historical databases. Artificial neural networks derive their name from their historical development which started off with the premise that machines could be made to “think” if scientists found ways to mimic the structure and functioning of the human brain on the computer. Thus historically neural networks grew out of the community of Artificial Intelligence rather than from the discipline of statistics. Despite the fact that scientists are still far from understanding the human brain let alone mimicking it, neural networks that run on computers can do some of the things that people can do.

It is difficult to say exactly when the first “neural network” on a computer was built. During World War II a seminal paper was published by McCulloch and Pitts which first outlined the idea that simple processing units (like the individual neurons in the human brain) could be connected together in large networks to create a system that could solve difficult problems and display behavior that was much more complex than the simple pieces that made it up. Since that time much progress has been made in finding ways to apply artificial neural networks to real world prediction problems and in improving the performance of the algorithm in general. In many respects the greatest breakthroughs in neural networks in recent years have been in their application to more mundane real world

problems like customer response prediction or fraud detection rather than the loftier goals that were originally set out for the techniques such as overall human learning and computer speech and image understanding.

Don't Neural Networks Learn to make better predictions?

Because of the origins of the techniques and because of some of their early successes the techniques have enjoyed a great deal of interest. To understand how neural networks can detect patterns in a database an analogy is often made that they "learn" to detect these patterns and make better predictions in a similar way to the way that human beings do. This view is encouraged by the way the historical training data is often supplied to the network - one record (example) at a time. Neural networks do "learn" in a very real sense but under the hood the algorithms and techniques that are being deployed are not truly different from the techniques found in statistics or other data mining algorithms. It is for instance, unfair to assume that neural networks could outperform other techniques because they "learn" and improve over time while the other techniques are static. The other techniques if fact "learn" from historical examples in exactly the same way but often times the examples (historical records) to learn from a processed all at once in a more efficient manner than neural networks which often modify their model one record at a time.

Are Neural Networks easy to use?

A common claim for neural networks is that they are automated to a degree where the user does not need to know that much about how they work, or predictive modeling or even the database in order to use them. The implicit claim is also that most neural networks can be unleashed on your data straight out of the box without having to rearrange or modify the data very much to begin with.

Just the opposite is often true. There are many important design decisions that need to be made in order to effectively use a neural network such as:

- How should the nodes in the network be connected?
- How many neuron like processing units should be used?
- When should "training" be stopped in order to avoid overfitting?

There are also many important steps required for preprocessing the data that goes into a neural network - most often there is a requirement to normalize numeric data between 0.0 and 1.0 and categorical predictors may need to be broken up into virtual predictors that are 0 or 1 for each value of the original categorical predictor. And, as always, understanding what the data in your database means and a clear definition of the business problem to be solved are essential to ensuring eventual success. The bottom line is that neural networks provide no short cuts.

Applying Neural Networks to Business

Neural networks are very powerful predictive modeling techniques but some of the power comes at the expense of ease of use and ease of deployment. As we will see in this section, neural networks, create very complex models that are almost always impossible to fully understand even by experts. The model itself is represented by numeric values in a complex calculation that requires all of the

predictor values to be in the form of a number. The output of the neural network is also numeric and needs to be translated if the actual prediction value is categorical (e.g. predicting the demand for blue, white or black jeans for a clothing manufacturer requires that the predictor values blue, black and white for the predictor color to be converted to numbers).

Because of the complexity of these techniques much effort has been expended in trying to increase the clarity with which the model can be understood by the end user. These efforts are still in there infancy but are of tremendous importance since most data mining techniques including neural networks are being deployed against real business problems where significant investments are made based on the predictions from the models (e.g. consider trusting the predictive model from a neural network that dictates which one million customers will receive a $1 mailing).

There are two ways that these shortcomings in understanding the meaning of the neural network model have been successfully addressed:

- The neural network is package up into a complete solution such as fraud prediction. This allows the neural network to be carefully crafted for one particular application and once it has been proven successful it can be used over and over again without requiring a deep understanding of how it works.

- The neural network is package up with expert consulting services. Here the neural network is deployed by trusted experts who have a track record of success. Either the experts are able to explain the models or they are trusted that the models do work.

The first tactic has seemed to work quite well because when the technique is used for a well defined problem many of the difficulties in preprocessing the data can be automated (because the data structures have been seen before) and interpretation of the model is less of an issue since entire industries begin to use the technology successfully and a level of trust is created. There are several vendors who have deployed this strategy (e.g. HNC's Falcon system for credit card fraud prediction and Advanced Software Applications ModelMAX package for direct marketing).

Packaging up neural networks with expert consultants is also a viable strategy that avoids many of the pitfalls of using neural networks, but it can be quite expensive because it is human intensive. One of the great promises of data mining is, after all, the automation of the predictive modeling process. These neural network consulting teams are little different from the analytical departments many companies already have in house. Since there is not a great difference in the overall predictive accuracy of neural networks over standard statistical techniques the main difference becomes the replacement of the statistical expert with the neural network expert. Either with statistics or neural network experts the value of putting easy to use tools into the hands of the business end user is still not achieved.

Where to Use Neural Networks

Neural networks are used in a wide variety of applications. They have been used in all facets of business from detecting the fraudulent use of credit cards and credit risk prediction to increasing the hit rate of targeted mailings. They also have a long history of application in other areas such as the military for the automated driving of an unmanned vehicle at 30 miles per hour on paved roads to biological simulations such as learning the correct pronunciation of English words from written text.

Neural Networks for clustering

Neural networks of various kinds can be used for clustering and prototype creation. The Kohonen network described in this section is probably the most common network used for clustering and segmentation of the database. Typically the networks are used in a unsupervised learning mode to create the clusters. The clusters are created by forcing the system to compress the data by creating prototypes or by algorithms that steer the system toward creating clusters that compete against each other for the records that they contain, thus ensuring that the clusters overlap as little as possible.

Neural Networks for Outlier Analysis

Sometimes clustering is performed not so much to keep records together as to make it easier to see when one record sticks out from the rest. For instance:

Most wine distributors selling inexpensive wine in Missouri and that ship a certain volume of product produce a certain level of profit. There is a cluster of stores that can be formed with these characteristics. One store stands out, however, as producing significantly lower profit. On closer examination it turns out that the distributor was delivering product to but not collecting payment from one of their customers.

A sale on men's suits is being held in all branches of a department store for southern California . All stores with these characteristics have seen at least a 100% jump in revenue since the start of the sale except one. It turns out that this store had, unlike the others, advertised via radio rather than television.

Neural Networks for feature extraction

One of the important problems in all of data mining is that of determining which predictors are the most relevant and the most important in building models that are most accurate at prediction. These predictors may be used by themselves or they may be used in conjunction with other predictors to form "features". A simple example of a feature in problems that neural networks are working on is the feature of a vertical line in a computer image. The predictors, or raw input data are just the colored pixels that make up the picture. Recognizing that the predictors (pixels) can be organized in such a way as to create lines, and then using the line as the input predictor can prove to dramatically improve the accuracy of the model and decrease the time to create it.

Some features like lines in computer images are things that humans are already pretty good at detecting, in other problem domains it is more difficult to recognize the features. One novel way that neural networks have been used to detect features is the idea that features are sort of a compression of the training database. For instance you could describe an image to a friend by rattling off the color and intensity of each pixel on every point in the picture or you could describe it at a higher level in terms of lines, circles - or maybe even at a higher level of features such as trees, mountains etc. In either case your friend eventually gets all the information that they need in order to know what the picture looks like, but certainly describing it in terms of high level features requires much less communication of information than the "paint by numbers" approach of describing the color on each square millimeter of the image.

If we think of features in this way, as an efficient way to communicate our data, then neural networks

can be used to automatically extract them. The neural network shown in Figure 2.2 is used to extract features by requiring the network to learn to recreate the input data at the output nodes by using just 5 hidden nodes. Consider that if you were allowed 100 hidden nodes, that recreating the data for the network would be rather trivial - simply pass the input node value directly through the corresponding hidden node and on to the output node. But as there are fewer and fewer hidden nodes, that information has to be passed through the hidden layer in a more and more efficient manner since there are less hidden nodes to help pass along the information.

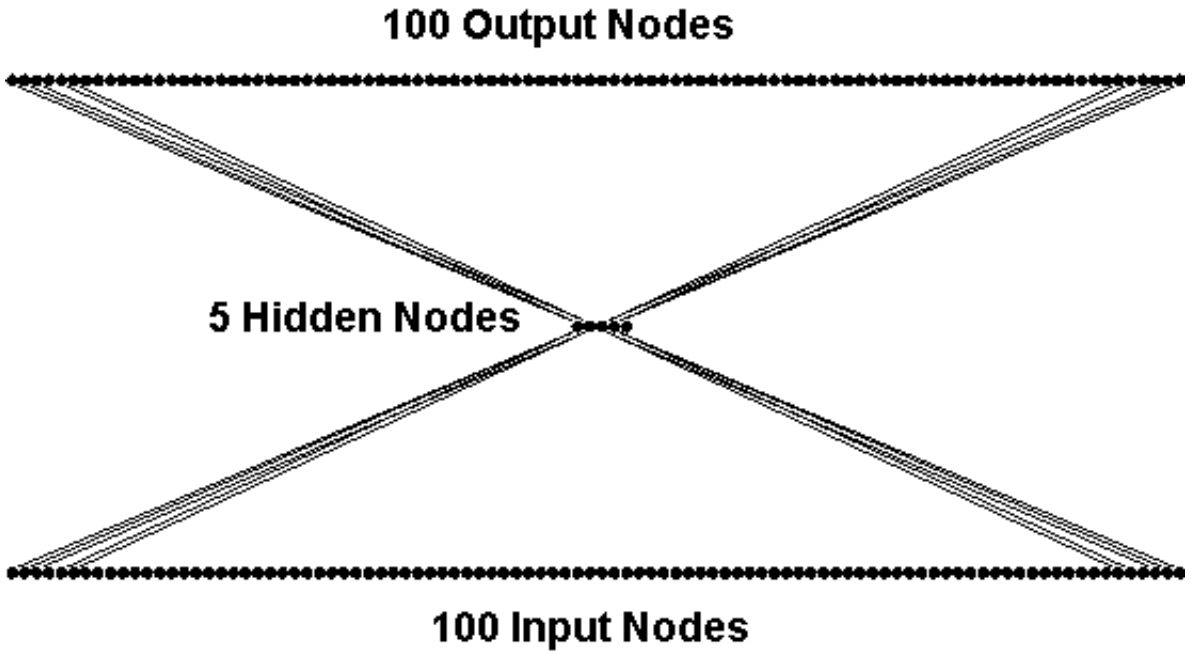

Figure 2.2 *Neural networks can be used for data compression and feature extraction.*

In order to accomplish this the neural network tries to have the hidden nodes extract features from the input nodes that efficiently describe the record represented at the input layer. This forced "squeezing" of the data through the narrow hidden layer forces the neural network to extract only those predictors and combinations of predictors that are best at recreating the input record. The link weights used to create the inputs to the hidden nodes are effectively creating features that are combinations of the input nodes values.

What does a neural net look like?

A neural network is loosely based on how some people believe that the human brain is organized and how it learns. Given that there are two main structures of consequence in the neural network:

The node - which loosely corresponds to the neuron in the human brain.

The link - which loosely corresponds to the connections between neurons (axons, dendrites and synapses) in the human brain.

In Figure 2.3 there is a drawing of a simple neural network. The round circles represent the nodes and the connecting lines represent the links. The neural network functions by accepting predictor

values at the left and performing calculations on those values to produce new values in the node at the far right. The value at this node represents the prediction from the neural network model. In this case the network takes in values for predictors for age and income and predicts whether the person will default on a bank loan.

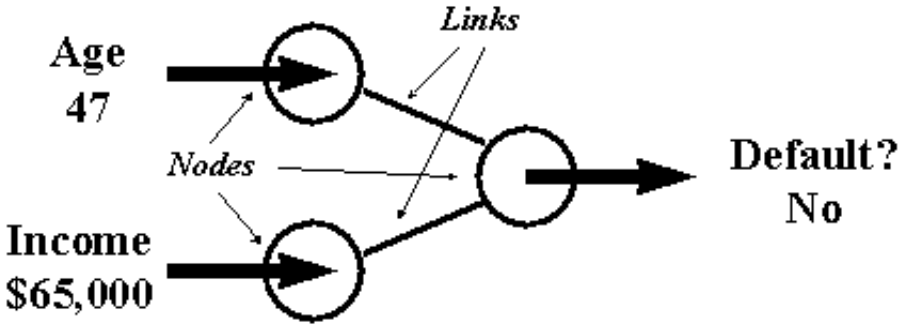

Figure 2.3 *A simplified view of a neural network for prediction of loan default.*

How does a neural net make a prediction?

In order to make a prediction the neural network accepts the values for the predictors on what are called the input nodes. These become the values for those nodes those values are then multiplied by values that are stored in the links (sometimes called links and in some ways similar to the weights that were applied to predictors in the nearest neighbor method). These values are then added together at the node at the far right (the output node) a special thresholding function is applied and the resulting number is the prediction. In this case if the resulting number is 0 the record is considered to be a good credit risk (no default) if the number is 1 the record is considered to be a bad credit risk (likely default).

A simplified version of the calculations made in Figure 2.3 might look like what is shown in Figure 2.4. Here the value age of 47 is normalized to fall between 0.0 and 1.0 and has the value 0.47 and the income is normalized to the value 0.65. This simplified neural network makes the prediction of no default for a 47 year old making $65,000. The links are weighted at 0.7 and 0.1 and the resulting value after multiplying the node values by the link weights is 0.39. The network has been trained to learn that an output value of 1.0 indicates default and that 0.0 indicates non-default. The output value calculated here (0.39) is closer to 0.0 than to 1.0 so the record is assigned a non-default prediction.

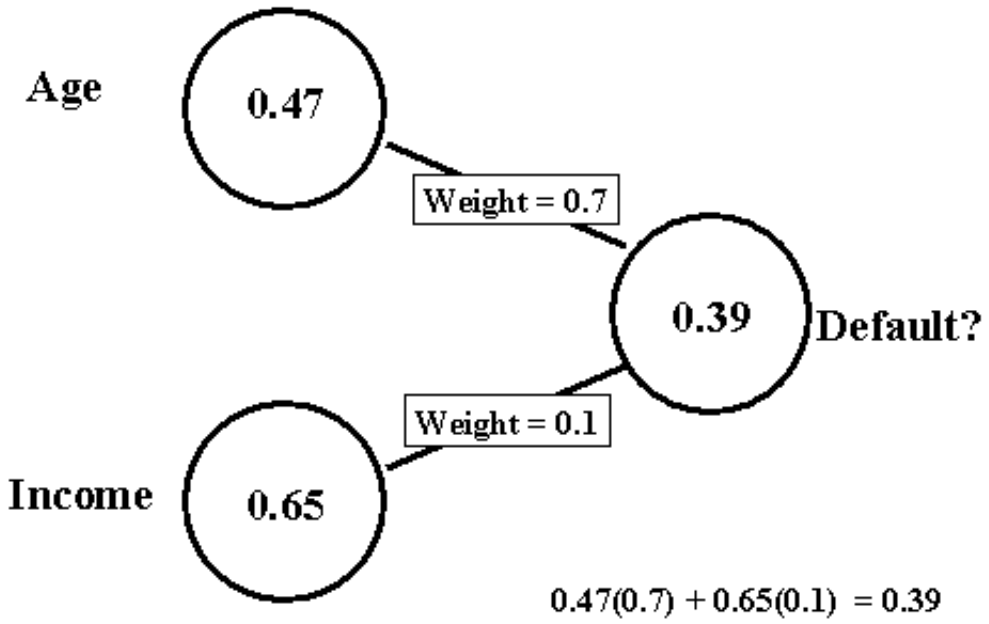

Figure 2.4 *The normalized input values are multiplied by the link weights and added together at the output.*

How is the neural net model created?

The neural network model is created by presenting it with many examples of the predictor values from records in the training set (in this example age and income are used) and the prediction value from those same records. By comparing the correct answer obtained from the training record and the predicted answer from the neural network it is possible to slowly change the behavior of the neural network by changing the values of the link weights. In some ways this is like having a grade school teacher ask questions of her student (a.k.a. the neural network) and if the answer is wrong to verbally correct the student. The greater the error the harsher the verbal correction. So that large errors are given greater attention at correction than are small errors.

For the actual neural network it is the weights of the links that actually control the prediction value for a given record. Thus the particular model that is being found by the neural network is in fact fully determined by the weights and the architectural structure of the network. For this reason it is the link weights that are modified each time an error is made.

How complex can the neural network model become?

The models shown in the figures above have been designed to be as simple as possible in order to make them understandable. In practice no networks are as simple as these. Networks with many more links and many more nodes are possible. This was the case in the architecture of a neural network system called NETtalk that learned how to pronounce written English words. Each node in this network was connected to every node in the level above it and below it resulting in 18,629 link weights that needed to be learned in the network.

In this network there was a row of nodes in between the input nodes and the output nodes. These

are called hidden nodes or the hidden layer because the values of these nodes are not visible to the end user the way that the output nodes are (that contain the prediction) and the input nodes (which just contain the predictor values). There are even more complex neural network architectures that have more than one hidden layer. In practice one hidden layer seems to suffice however.

Hidden nodes are like trusted advisors to the output nodes

The meaning of the input nodes and the output nodes are usually pretty well understood - and are usually defined by the end user based on the particular problem to be solved and the nature and structure of the database. The hidden nodes, however, do not have a predefined meaning and are determined by the neural network as it trains. Which poses two problems:

- It is difficult to trust the prediction of the neural network if the meaning of these nodes is not well understood.
- ince the prediction is made at the output layer and the difference between the prediction and the actual value is calculated there, how is this error correction fed back through the hidden layers to modify the link weights that connect them?

The meaning of these hidden nodes is not necessarily well understood but sometimes after the fact they can be looked at to see when they are active and when they are not and derive some meaning from them.

The learning that goes on in the hidden nodes.

The learning procedure for the neural network has been defined to work for the weights in the links connecting the hidden layer. A good metaphor for how this works is to think of a military operation in some war where there are many layers of command with a general ultimately responsible for making the decisions on where to advance and where to retreat. The general probably has several lieutenant generals advising him and each lieutenant general probably has several major generals advising him. This hierarchy continuing downward through colonels and privates at the bottom of the hierarchy.

This is not too far from the structure of a neural network with several hidden layers and one output node. You can think of the inputs coming from the hidden nodes as advice. The link weight corresponds to the trust that the general has in his advisors. Some trusted advisors have very high weights and some advisors may no be trusted and in fact have negative weights. The other part of the advice from the advisors has to do with how competent the particular advisor is for a given situation. The general may have a trusted advisor but if that advisor has no expertise in aerial invasion and the question at hand has to do with a situation involving the air force this advisor may be very well trusted but the advisor himself may not have any strong opinion one way or another.

In this analogy the link weight of a neural network to an output unit is like the trust or confidence that a commander has in his advisors and the actual node value represents how strong an opinion this particular advisor has about this particular situation. To make a decision the general considers how trustworthy and valuable the advice is and how knowledgeable and confident each advisor is in making their suggestion and then taking all of this into account the general makes the decision to advance or retreat.

In the same way the output node will make a decision (a prediction) by taking into account all of the input from its advisors (the nodes connected to it). In the case of the neural network this decision is reach by multiplying the link weight by the output value of the node and summing these values across all nodes. If the prediction is incorrect the nodes that had the most influence on making the decision have their weights modified so that the wrong prediction is less likely to be made the next time.

This learning in the neural network is very similar to what happens when the wrong decision is made by the general. The confidence that the general has in all of those advisors that gave the wrong recommendation is decreased - and all the more so for those advisors who were very confident and vocal in their recommendation. On the other hand any advisors who were making the correct recommendation but whose input was not taken as seriously would be taken more seriously the next time. Likewise any advisor that was reprimanded for giving the wrong advice to the general would then go back to his advisors and determine which of them he had trusted more than he should have in making his recommendation and who he should have listened more closely to.

Sharing the blame and the glory throughout the organization

This feedback can continue in this way down throughout the organization - at each level giving increased emphasis to those advisors who had advised correctly and decreased emphasis to those who had advised incorrectly. In this way the entire organization becomes better and better and supporting the general in making the correct decision more of the time.

A very similar method of training takes place in the neural network. It is called “back propagation” and refers to the propagation of the error backwards from the output nodes (where the error is easy to determine the difference between the actual prediction value from the training database and the prediction from the neural network) through the hidden layers and to the input layers. At each level the link weights between the layers are updated so as to decrease the chance of making the same mistake again.

Different types of neural networks

There are literally hundreds of variations on the back propagation feedforward neural networks that have been briefly described here. Most having to do with changing the architecture of the neural network to include recurrent connections where the output from the output layer is connected back as input into the hidden layer. These recurrent nets are some times used for sequence prediction where the previous outputs from the network need to be stored someplace and then fed back into the network to provide context for the current prediction. Recurrent networks have also been used for decreasing the amount of time that it takes to train the neural network.

Another twist on the neural net theme is to change the way that the network learns. Back propagation is effectively utilizing a search technique called gradient descent to search for the best possible improvement in the link weights to reduce the error. There are, however, many other ways of doing search in a high dimensional space including Newton’s methods and conjugate gradient as well as simulating the physics of cooling metals in a process called simulated annealing or in simulating the search process that goes on in biological evolution and using genetic algorithms to optimize the weights of the neural networks. It has even been suggested that creating a large number of neural networks with randomly weighted links and picking the one with the lowest error rate would be the best learning procedure.

Despite all of these choices, the back propagation learning procedure is the most commonly used. It is well understand, relatively simple, and seems to work in a large number of problem domains. There are, however, two other neural network architectures that are used relatively often. Kohonen feature maps are often used for unsupervised learning and clustering and Radial Basis Function networks are used for supervised learning and in some ways represent a hybrid between nearest neighbor and neural network classification.

Kohonen Feature Maps

Kohonen feature maps were developed in the 1970's and as such were created to simulate certain brain function. Today they are used mostly to perform unsupervised learning and clustering.

Kohonen networks are feedforward neural networks generally with no hidden layer. The networks generally contain only an input layer and an output layer but the nodes in the output layer compete amongst themselves to display the strongest activation to a given record. What is sometimes called "winner take all".

The networks originally came about when some of the puzzling yet simple behaviors of the real neurons were taken into effect. Namely that physical locality of the neurons seems to play an important role in the behavior and learning of neurons.

When these networks were run, in order to simulate the real world visual system it became that the organization that was automatically being constructed on the data was also very useful for segmenting and clustering the training database. Each output node represented a cluster and nearby clusters were nearby in the two dimensional output layer. Each record in the database would fall into one and only one cluster (the most active output node) but the other clusters in which it might also fit would be shown and likely to be next to the best matching cluster.

How much like a human brain is the neural network?

Since the inception of the idea of neural networks the ultimate goal for these techniques has been to have them recreate human thought and learning. This has once again proved to be a difficult task - despite the power of these new techniques and the similarities of their architecture to that of the human brain. Many of the things that people take for granted are difficult for neural networks - like avoiding overfitting and working with real world data without a lot of preprocessing required. There have also been some exciting successes.

Combatting overfitting - getting a model you can use somewhere else

As with all predictive modeling techniques some care must be taken to avoid overfitting with a neural network. Neural networks can be quite good at overfitting training data with a predictive model that does not work well on new data. This is particularly problematic for neural networks because it is difficult to understand how the model is working. In the early days of neural networks the predictive accuracy that was often mentioned first was the accuracy on the training set and the vaulted or validation set database was reported as a footnote.

This is in part due to the fact that unlike decision trees or nearest neighbor techniques, which can quickly achieve 100% predictive accuracy on the training database, neural networks can be trained

forever and still not be 100% accurate on the training set. While this is an interesting fact it is not terribly relevant since the accuracy on the training set is of little interest and can have little bearing on the validation database accuracy.

Perhaps because overfitting was more obvious for decision trees and nearest neighbor approaches more effort was placed earlier on to add pruning and editing to these techniques. For neural networks generalization of the predictive model is accomplished via rules of thumb and sometimes in a more methodically way by using cross validation as is done with decision trees.

One way to control overfitting in neural networks is to limit the number of links. Since the number of links represents the complexity of the model that can be produced, and since more complex models have the ability to overfit while less complex ones cannot, overfitting can be controlled by simply limiting the number of links in the neural network. Unfortunately there is no god theoretical grounds for picking a certain number of links.

Test set validation can be used to avoid overfitting by building the neural network on one portion of the training database and using the other portion of the training database to detect what the predictive accuracy is on vaulted data. This accuracy will peak at some point in the training and then as training proceeds it will decrease while the accuracy on the training database will continue to increase. The link weights for the network can be saved when the accuracy on the held aside data peaks. The NeuralWare product, and others, provide an automated function that saves out the network when it is best performing on the test set and even continues to search after the minimum is reached.

Explaining the network

One of the indictments against neural networks is that it is difficult to understand the model that they have built and also how the raw data effects the output predictive answer. With nearest neighbor techniques prototypical records are provided to "explain" why the prediction is made, and decision trees provide rules that can be translated in to English to explain why a particular prediction was made for a particular record. The complex models of the neural network are captured solely by the link weights in the network which represent a very complex mathematical equation.

There have been several attempts to alleviate these basic problems of the neural network. The simplest approach is to actually look at the neural network and try to create plausible explanations for the meanings of the hidden nodes. Some times this can be done quite successfully. In the example given at the beginning of this section the hidden nodes of the neural network seemed to have extracted important distinguishing features in predicting the relationship between people by extracting information like country of origin. Features that it would seem that a person would also extract and use for the prediction. But there were also many other hidden nodes, even in this particular example that were hard to explain and didn't seem to have any particular purpose. Except that they aided the neural network in making the correct prediction.

2.4. Rule Induction

Rule induction is one of the major forms of data mining and is perhaps the most common form of knowledge discovery in unsupervised learning systems. It is also perhaps the form of data mining that most closely resembles the process that most people think about when they think about data mining, namely "mining" for gold through a vast database. The gold in this case would be a rule that

is interesting - that tells you something about your database that you didn't already know and probably weren't able to explicitly articulate (aside from saying "show me things that are interesting").

Rule induction on a data base can be a massive undertaking where all possible patterns are systematically pulled out of the data and then an accuracy and significance are added to them that tell the user how strong the pattern is and how likely it is to occur again. In general these rules are relatively simple such as for a market basket database of items scanned in a consumer market basket you might find interesting correlations in your database such as:

- If bagels are purchased then cream cheese is purchased 90% of the time and this pattern occurs in 3% of all shopping baskets.

- If live plants are purchased from a hardware store then plant fertilizer is purchased 60% of the time and these two items are bought together in 6% of the shopping baskets.

The rules that are pulled from the database are extracted and ordered to be presented to the user based on the percentage of times that they are correct and how often they apply.

The bane of rule induction systems is also its strength - that it retrieves all possible interesting patterns in the database. This is a strength in the sense that it leaves no stone unturned but it can also be viewed as a weaknes because the user can easily become overwhelmed with such a large number of rules that it is difficult to look through all of them. You almost need a second pass of data mining to go through the list of interesting rules that have been generated by the rule induction system in the first place in order to find the most valuable gold nugget amongst them all. This overabundance of patterns can also be problematic for the simple task of prediction because all possible patterns are culled from the database there may be conflicting predictions made by equally interesting rules. Automating the process of culling the most interesting rules and of combing the recommendations of a variety of rules are well handled by many of the commercially available rule induction systems on the market today and is also an area of active research.

Applying Rule induction to Business

Rule induction systems are highly automated and are probably the best of data mining techniques for exposing all possible predictive patterns in a database. They can be modified to for use in prediction problems but the algorithms for combining evidence from a variety of rules comes more from rules of thumbs and practical experience.

In comparing data mining techniques along an axis of explanation neural networks would be at one extreme of the data mining algorithms and rule induction systems at the other end. Neural networks are extremely proficient and saying exactly what must be done in a prediction task (e.g. who do I give credit to / who do I deny credit to) with little explanation. Rule induction systems when used for prediction on the other hand are like having a committee of trusted advisors each with a slightly different opinion as to what to do but relatively well grounded reasoning and a good explanation for why it should be done.

The business value of rule induction techniques reflects the highly automated way in which the rules are created which makes it easy to use the system but also that this approach can suffer from an overabundance of interesting patterns which can make it complicated in order to make a prediction that is directly tied to return on investment (ROI).

What is a rule?

In rule induction systems the rule itself is of a simple form of "if this and this and this then this". For example a rule that a supermarket might find in their data collected from scanners would be: "if pickles are purchased then ketchup is purchased'. Or

- If paper plates then plastic forks
- If dip then potato chips
- If salsa then tortilla chips

In order for the rules to be useful there are two pieces of information that must be supplied as well as the actual rule:

- Accuracy - How often is the rule correct?
- Coverage - How often does the rule apply?

Just because the pattern in the data base is expressed as rule does not mean that it is true all the time. Thus just like in other data mining algorithms it is important to recognize and make explicit the uncertainty in the rule. This is what the accuracy of the rule means. The coverage of the rule has to do with how much of the database the rule "covers" or applies to. Examples of these two measure for a variety of rules is shown in Table 2.2.

In some cases accuracy is called the confidence of the rule and coverage is called the support. Accuracy and coverage appear to be the preferred ways of naming these two measurements.

Rule	Accuracy	Coverage
If breakfast cereal purchased then milk purchased.	85%	20%
If bread purchased then swiss cheese purchased.	15%	6%
If 42 years old and purchased pretzels and purchased dry roasted peanuts then beer will be purchased.	95%	0.01%

Table 2.2 *Examples of Rule Accuracy and Coverage*

The rules themselves consist of two halves. The left hand side is called the antecedent and the right hand side is called the consequent. The antecedent can consist of just one condition or multiple conditions which must all be true in order for the consequent to be true at the given accuracy. Generally the consequent is just a single condition (prediction of purchasing just one grocery store item) rather than multiple conditions. Thus rules such as: "if x and y then a and b and c".

What to do with a rule

When the rules are mined out of the database the rules can be used either for understanding better the business problems that the data reflects or for performing actual predictions against some predefined prediction target. Since there is both a left side and a right side to a rule (antecedent and consequent) they can be used in several ways for your business.

Target the antecedent. In this case all rules that have a certain value for the antecedent are gathered and displayed to the user. For instance a grocery store may request all rules that have nails, bolts or screws as the antecedent in order to try to understand whether discontinuing the sale of these low margin items will have any effect on other higher margin. For instance maybe people who buy nails also buy expensive hammers but wouldn't do so at the store if the nails were not available.

Target the consequent. In this case all rules that have a certain value for the consequent can be used to understand what is associated with the consequent and perhaps what affects the consequent. For instance it might be useful to know all of the interesting rules that have "coffee" in their consequent. These may well be the rules that affect the purchases of coffee and that a store owner may want to put close to the coffee in order to increase the sale of both items. Or it might be the rule that the coffee manufacturer uses to determine in which magazine to place their next coupons.

Target based on accuracy. Some times the most important thing for a user is the accuracy of the rules that are being generated. Highly accurate rules of 80% or 90% imply strong relationships that can be exploited even if they have low coverage of the database and only occur a limited number of times. For instance a rule that only has 0.1% coverage but 95% can only be applied one time out of one thousand but it will very likely be correct. If this one time is highly profitable that it can be worthwhile. This, for instance, is how some of the most successful data mining applications work in the financial markets - looking for that limited amount of time where a very confident prediction can be made.

Target based on coverage. Some times user want to know what the most ubiquitous rules are or those rules that are most readily applicable. By looking at rules ranked by coverage they can quickly get a high level view of what is happening within their database most of the time.

Target based on "interestingness". Rules are interesting when they have high coverage and high accuracy and deviate from the norm. There have been many ways that rules have been ranked by some measure of interestingness so that the trade off between coverage and accuracy can be made.

Since rule induction systems are so often used for pattern discovery and unsupervised learning it is less easy to compare them. For example it is very easy for just about any rule induction system to generate all possible rules, it is, however, much more difficult to devise a way to present those rules (which could easily be in the hundreds of thousands) in a way that is most useful to the end user. When interesting rules are found they usually have been created to find relationships between many different predictor values in the database not just one well defined target of the prediction. For this reason it is often much more difficult to assign a measure of value to the rule aside from its interestingness. For instance it would be difficult to determine the monetary value of knowing that if people buy breakfast sausage they also buy eggs 60% of the time. For data mining systems that are more focused on prediction for things like customer attrition, targeted marketing response or risk it is much easier to measure the value of the system and compare it to other systems and other methods for solving the problem.

Caveat: Rules do not imply causality

It is important to recognize that even though the patterns produced from rule induction systems are delivered as if then rules they do not necessarily mean that the left hand side of the rule (the "if" part) causes the right hand side of the rule (the "then" part) to happen. Purchasing cheese does not cause the purchase of wine even though the rule if cheese then wine may be very strong.

This is particularly important to remember for rule induction systems because the results are presented as if this then that as many causal relationships are presented.

Types of databases used for rule induction

Typically rule induction is used on databases with either fields of high cardinality (many different values) or many columns of binary fields. The classical case of this is the super market basket data from store scanners that contains individual product names and quantities and may contain tens of thousands of different items with different packaging that create hundreds of thousands of SKU identifiers (Stock Keeping Units).

Sometimes in these databases the concept of a record is not easily defined within the database - consider the typical Star Schema for many data warehouses that store the supermarket transactions as separate entries in the fact table. Where the columns in the fact table are some unique identifier of the shopping basket (so all items can be noted as being in the same shopping basket), the quantity, the time of purchase, whether the item was purchased with a special promotion (sale or coupon). Thus each item in the shopping basket has a different row in the fact table. This layout of the data is not typically the best for most data mining algorithms which would prefer to have the data structured as one row per shopping basket and each column to represent the presence or absence of a given item. This can be an expensive way to store the data, however, since the typical grocery store contains 60,000 SKUs or different items that could come across the checkout counter. This structure of the records can also create a very high dimensional space (60,000 binary dimensions) which would be unwieldy for many classical data mining algorithms like neural networks and decision trees. As we'll see several tricks are played to make this computationally feasible for the data mining algorithm while not requiring a massive reorganization of the database.

Discovery

The claim to fame of these ruled induction systems is much more so for knowledge discovers in unsupervised learning systems than it is for prediction. These systems provide both a very detailed view of the data where significant patterns that only occur a small portion of the time and only can be found when looking at the detail data as well as a broad overview of the data where some systems seek to deliver to the user an overall view of the patterns contained n the database. These systems thus display a nice combination of both micro and macro views:

- Macro Level - Patterns that cover many situations are provided to the user that can be used very often and with great confidence and can also be used to summarize the database.

- Micro Level - Strong rules that cover only a very few situations can still be retrieved by the system and proposed to the end user. These may be valuable if the situations that are covered are highly valuable (maybe they only apply to the most profitable customers) or represent a small but growing subpopulation which may indicate a market shift or the emergence of a new competitor (e.g. customers are only being lost in one particular area of the country where a new competitor is emerging).

Prediction

After the rules are created and their interestingness is measured there is also a call for performing prediction with the rules. Each rule by itself can perform prediction - the consequent is the target and the accuracy of the rule is the accuracy of the prediction. But because rule induction systems produce many rules for a given antecedent or consequent there can be conflicting predictions with different accuracies. This is an opportunity for improving the overall performance of the systems by combining the rules. This can be done in a variety of ways by summing the accuracies as if they were weights or just by taking the prediction of the rule with the maximum accuracy.

Table 2.3 shows how a given consequent or antecedent can be part of many rules with different accuracies and coverages. From this example consider the prediction problem of trying to predict whether milk was purchased based solely on the other items that were in the shopping basket. If the shopping basket contained only bread then from the table we would guess that there was a 35% chance that milk was also purchased. If, however, bread and butter and eggs and cheese were purchased what would be the prediction for milk then? 65% chance of milk because the relationship between butter and milk is the greatest at 65%? Or would all of the other items in the basket increase even further the chance of milk being purchased to well beyond 65%? Determining how to combine evidence from multiple rules is a key part of the algorithms for using rules for prediction.

Antecedent	Consequent	Accuracy	Coverage
bagels	cream cheese	80%	5%
bagels	orange juice	40%	3%
bagels	coffee	40%	2%
bagels	eggs	25%	2%
bread	milk	35%	30%
butter	milk	65%	20%
eggs	milk	35%	15%
cheese	milk	40%	8%

Table 2.3 *Accuracy and Coverage in Rule Antecedents and Consequents*

The General Idea

The general idea of a rule classification system is that rules are created that show the relationship between events captured in your database. These rules can be simple with just one element in the antecedent or they might be more complicated with many column value pairs in the antecedent all joined together by a conjunction (item1 and item2 and item3 … must all occur for the antecedent to be true).

The rules are used to find interesting patterns in the database but they are also used at times for prediction. There are two main things that are important to understanding a rule:

Accuracy - Accuracy refers to the probability that if the antecedent is true that the precedent will be true. High accuracy means that this is a rule that is highly dependable.

Coverage - Coverage refers to the number of records in the database that the rule applies to. High

coverage means that the rule can be used very often and also that it is less likely to be a spurious artifact of the sampling technique or idiosyncrasies of the database.

The business importance of accuracy and coverage

From a business perspective accurate rules are important because they imply that there is useful predictive information in the database that can be exploited - namely that there is something far from independent between the antecedent and the consequent. The lower the accuracy the closer the rule comes to just random guessing. If the accuracy is significantly below that of what would be expected from random guessing then the negation of the antecedent may well in fact be useful (for instance people who buy denture adhesive are much less likely to buy fresh corn on the cob than normal).

From a business perspective coverage implies how often you can use a useful rule. For instance you may have a rule that is 100% accurate but is only applicable in 1 out of every 100,000 shopping baskets. You can rearrange your shelf space to take advantage of this fact but it will not make you much money since the event is not very likely to happen. Table 2.4. Displays the trade off between coverage and accuracy.

	Accuracy Low	Accuracy High
Coverage High	Rule is rarely correct but can be used often.	Rule is often correct and can be used often.
Coverage Low	Rule is rarely correct and can be only rarely used.	Rule is often correct but can be only rarely used.

Table 2.4 *Rule coverage versus accuracy*.

Trading off accuracy and coverage is like betting at the track

An analogy between coverage and accuracy and making money is the following from betting on horses. Having a high accuracy rule with low coverage would be like owning a race horse that always won when he raced but could only race once a year. In betting, you could probably still make a lot of money on such a horse. In rule induction for retail stores it is unlikely that finding that one rule between mayonnaise, ice cream and sardines that seems to always be true will have much of an impact on your bottom line.

How to evaluate the rule

One way to look at accuracy and coverage is to see how they relate so some simple statistics and how they can be represented graphically. From statistics coverage is simply the a priori probability of the antecedent and the consequent occurring at the same time. The accuracy is just the probability of the consequent conditional on the precedent. So, for instance the if we were looking at the following database of super market basket scanner data we would need the following information in order to calculate the accuracy and coverage for a simple rule (let's say milk purchase implies eggs purchased).

T = 100 = Total number of shopping baskets in the database.

E = 30 = Number of baskets with eggs in them.

M = 40 = Number of baskets with milk in them.

B = 20 = Number of baskets with both eggs and milk in them.

Accuracy is then just the number of baskets with eggs and milk in them divided by the number of baskets with milk in them. In this case that would be 20/40 = 50%. The coverage would be the number of baskets with milk in them divided by the total number of baskets. This would be 40/100 = 40%. This can be seen graphically in Figure 2.5.

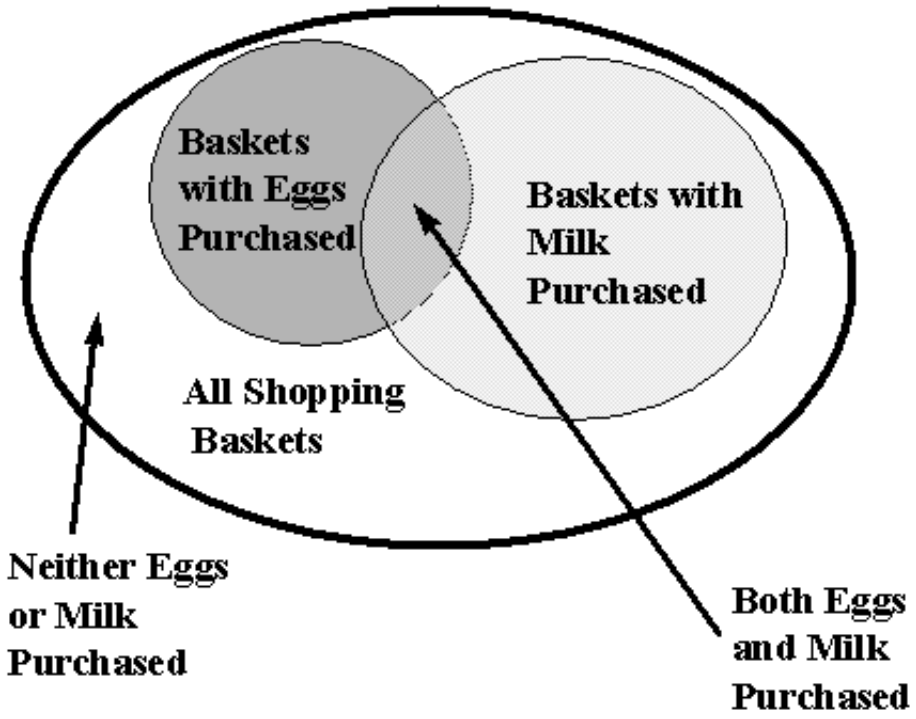

Figure 2.5 *Graphically the total number of shopping baskets can be represented in a space and the number of baskets containing eggs or milk can be represented by the area of a circle. The coverage of the rule "If Milk then Eggs" is just the relative size of the circle corresponding to milk. The accuracy is the relative size of the overlap between the two to the circle representing milk purchased.*

Notice that we haven't used E the number of baskets with eggs in these calculations. One way that eggs could be used would be to calculate the expected number of baskets with eggs and milk in them based on the independence of the events. This would give us some sense of how unlikely and how special the event is that 20% of the baskets have both eggs and milk in them. Remember from the statistics section that if two events are independent (have no effect on one another) that the product of their individual probabilities of occurrence should equal the probability of the occurrence of them both together.

If the purchase of eggs and milk were independent of each other one would expect that 0.3 x 0.4 = 0.12 or 12% of the time we would see shopping baskets with both eggs and milk in them. The fact that this combination of products occurs 20% of the time is out of the ordinary if these events were

independent. That is to say there is a good chance that the purchase of one effects the other and the degree to which this is the case could be calculated through statistical tests and hypothesis testing.

Defining “interestingness”

One of the biggest problems with rule induction systems is the sometimes overwhelming number of rules that are produced. Most of which have no practical value or interest. Some of the rules are so inaccurate that they cannot be used, some have so little coverage that though they are interesting they have little applicability, and finally many of the rules capture patterns and information that the user is already familiar with. To combat this problem researchers have sought to measure the usefulness or interestingness of rules.

Certainly any measure of interestingness would have something to do with accuracy and coverage. We might also expect it to have at least the following four basic behaviors:

- Interestingness = 0 if the accuracy of the rule is equal to the background accuracy (a priori probability of the consequent). The example in Table 2.5 shows an example of this. Where a rule for attrition is no better than just guessing the overall rate of attrition.
- Interestingness increases as accuracy increases (or decreases with decreasing accuracy) if the coverage is fixed.
- Interestingness increases or decreases with coverage if accuracy stays fixed
- Interestingness decreases with coverage for a fixed number of correct responses (remember accuracy equals the number of correct responses divided by the coverage).

Antecedent	Consequent	Accuracy	Coverage
<no constraints>	then customer will attrite	10%	100%
If customer balance > $3,000	then customer will attrite	10%	60%
If customer eyes = blue	then customer will attrite	10%	30%
If customer social security number = 144 30 8217	then customer will attrite	100%	0.000001%

Table 2.5 *Uninteresting rules*

There are a variety of measures of interestingness that are used that have these general characteristics. They are used for pruning back the total possible number of rules that might be generated and then presented to the user.

Other measures of usefulness

Another important measure is that of simplicity of the rule. This is an important solely for the end

user. As complex rules, as powerful and as interesting as they might be, may be difficult to understand or to confirm via intuition. Thus the user has a desire to see simpler rules and consequently this desire can be manifest directly in the rules that are chosen and supplied automatically to the user.

Finally a measure of novelty is also required both during the creation of the rules - so that rules that are redundant but strong are less favored to be searched than rules that may not be as strong but cover important examples that are not covered by other strong rules. For instance there may be few historical records to provide rules on a little sold grocery item (e.g. mint jelly) and they may have low accuracy but since there are so few possible rules even though they are not interesting they will be "novel" and should be retained and presented to the user for that reason alone.

Rules vs. Decision trees

Decision trees also produce rules but in a very different way than rule induction systems. The main difference between the rules that are produced by decision trees and rule induction systems is as follows:

Decision trees produce rules that are mutually exclusive and collectively exhaustive with respect to the training database while rule induction systems produce rules that are not mutually exclusive and might be collectively exhaustive.

In plain English this means that for an given record there will be a rule to cover it and there will only be one rule for rules that come from decision trees. There may be many rules that match a given record from a rule induction system and for many systems it is not guaranteed that a rule will exist for each and every possible record that might be encountered (though most systems do create very general default rules to capture these records).

The reason for this difference is the way in which the two algorithms operate. Rule induction seeks to go from the bottom up and collect all possible patterns that are interesting and then later use those patterns for some prediction target. Decisions trees on the other hand work from a prediction target downward in what is known as a "greedy" search. Looking for the best possible split on the next step (i.e. greedily picking the best one without looking any further than the next step). Though the greedy algorithm can make choices at the higher levels of the tree which are less than optimal at the lower levels of the tree it is very good at effectively squeezing out any correlations between predictors and the prediction. Rule induction systems on the other hand retain all possible patterns even if they are redundant or do not aid in predictive accuracy.

For instance, consider that in a rule induction system that if there were two columns of data that were highly correlated (or in fact just simple transformations of each other) they would result in two rules whereas in a decision tree one predictor would be chosen and then since the second one was redundant it would not be chosen again. An example might be the two predictors annual charges and average monthly charges (average monthly charges being the annual charges divided by 12). If the amount charged was predictive then the decision tree would choose one of the predictors and use it for a split point somewhere in the tree. The decision tree effectively "squeezed" the predictive value out of the predictor and then moved onto the next. A rule induction system would on the other hand create two rules. Perhaps something like:

If annual charges > 12,000 then default = true 90% accuracy

If average monthly charges > 1,000 the default = true 90% accuracy.

In this case we've shown an extreme case where two predictors were exactly the same, but there can also be less extreme cases. For instance height might be used rather than shoe size in the decision tree whereas in a rule induction system both would be presented as rules.

Neither one technique or the other is necessarily better though having a variety of rules and predictors helps with the prediction when there are missing values. For instance if the decision tree did choose height as a split point but that predictor was not captured in the record (a null value) but shoe size was the rule induction system would still have a matching rule to capture this record. Decision trees do have ways of overcoming this difficulty by keeping "surrogates" at each split point that work almost as well at splitting the data as does the chosen predictor. In this case shoe size might have been kept as a surrogate for height at this particular branch of the tree.

Another commonality between decision trees and rule induction systems

One other thing that decision trees and rule induction systems have in common is the fact that they both need to find ways to combine and simplify rules. In a decision tree this can be as simple as recognizing that if a lower split on a predictor is more constrained than a split on the same predictor further up in the tree that both don't need to be provided to the user but only the more restrictive one. For instance if the first split of the tree is age <= 50 years and the lowest split for the given leaf is age <= 30 years then only the latter constraint needs to be captured in the rule for that leaf.

Rules from rule induction systems are generally created by taking a simple high level rule and adding new constraints to it until the coverage gets so small as to not be meaningful. This means that the rules actually have families or what is called "cones of specialization" where one more general rule can be the parent of many more specialized rules. These cones then can be presented to the user as high level views of the families of rules and can be viewed in a hierarchical manner to aid in understanding.

2.5. Which Technique and When?

Clearly one of the hardest things to do when deciding to implement a data mining system is to determine which technique to use when. When are neural networks appropriate and when are decision trees appropriate? When is data mining appropriate at all as opposed to just working with relational databases and reporting? When would just using OLAP and a multidimensional database be appropriate?

Some of the criteria that are important in determining the technique to be used are determined by trial and error. There are definite differences in the types of problems that are most conducive to each technique but the reality of real world data and the dynamic way in which markets, customers and hence the data that represents them is formed means that the data is constantly changing. These dynamics mean that it no longer makes sense to build the "perfect" model on the historical data since whatever was known in the past cannot adequately predict the future because the future is so unlike what has gone before.

In some ways this situation is analogous to the business person who is waiting for all information to come in before they make their decision. They are trying out different scenarios, different formulae

and researching new sources of information. But this is a task that will never be accomplished - at least in part because the business the economy and even the world is changing in unpredictable and even chaotic ways that could never be adequately predicted. Better to take a robust model that perhaps is an under-performer compared to what some of the best data mining tools could provide with a great deal of analysis and execute it today rather than to wait until tomorrow when it may be too late.

Balancing exploration and exploitation

There is always the trade off between exploration (learning more and gathering more facts) and exploitation (taking immediate advantage of everything that is currently known). This theme of exploration versus exploitation is echoed also at the level of collecting data in a targeted marketing system: from a limited population of prospects/customers to choose from how many to you sacrifice to exploration (trying out new promotions or messages at random) versus optimizing what you already know.

There was for instance no reasonable way that Barnes and Noble bookstores could in 1995 look at past sales figures and foresee the impact that Amazon books and others would have based on the internet sales model.

Compared to historic sales and marketing data the event of the internet could not be predicted based on the data alone. Instead perhaps data mining could have been used to detect trends of decreased sales to certain customer sub-populations - such as to those involved in the high tech industry that were the first to begin to buy books online at Amazon.

So caveat emptor - use the data mining tools well but strike while the iron is hot. The performance of predictive model provided by data mining tools have a limited half life of decay. Unlike a good bottle of wine they do not increase in value with age.

[Data Mining Page] [White Papers] [Data Mining Tutorial]