CHAPTER 3
CLASSIFICATION

Learning Objectives

1. Explain the concept of classification
2. Describe the Decision Tree method
3. Describe the Naïve Bayes method
4. Discuss accuracy of classification methods and how accuracy may be improved

3.1 INTRODUCTION

Classification is a classical problem extensively studied by statisticians and machine learning researchers. The word classification is difficult to define precisely. According to one definition *classification is the separation or ordering of objects (or things) into classes*. If the classes are created without looking at the data (non-empirically), the classification is called *apriori classification.* If however the classes are created empirically (by looking at the data), the classification is called *Posteriori classification.* In most literature on classification it is assumed that the classes have been deemed apriori and classification then consists of training the system so that when a new object is presented to the trained system it is able to assign the object to one of the existing classes. This approach is also called *supervised learning.*

Data mining has generated renewed interest in classification. Since the datasets in data mining are often large, new classification techniques have been developed to deal with millions of objects having perhaps dozens or even hundreds of attributes.

3.2 DECISION TREE

A decision tree is a popular classification method that results in a flow-chart like tree structure where each node denotes a test on an attribute value and each branch represents an outcome of the test. The tree leaves represent the classes.

Let us imagine that we wish to classify Australian animals. We have some training data in Table 3.1 which has already been classified. We want to build a model based on this data.

Table 3.1 Training data for a classification problem

Name	Eggs	Pouch	Flies	Feathers	Class
Cockatoo	Yes	No	Yes	Yes	Bird
Dugong	No	No	No	No	Mammal
Echidna	Yes	Yes	No	No	Marsupial
Emu	Yes	No	No	Yes	Bird
Kangaroo	No	Yes	No	No	Marsupial
Koala	No	Yes	No	No	Marsupial
Kookaburra	Yes	No	Yes	Yes	Bird
Owl	Yes	No	Yes	Yes	Bird
Penguin	Yes	No	No	Yes	Bird
Platypus	Yes	No	No	No	Mammal

Possum Marsupial	No	Yes	No	No
Wombat Marsupial	No	Yes	No	No

_

As an example of a decision tree, we show possible result in Figure 3.1 if classifying the data in Table 3.1.

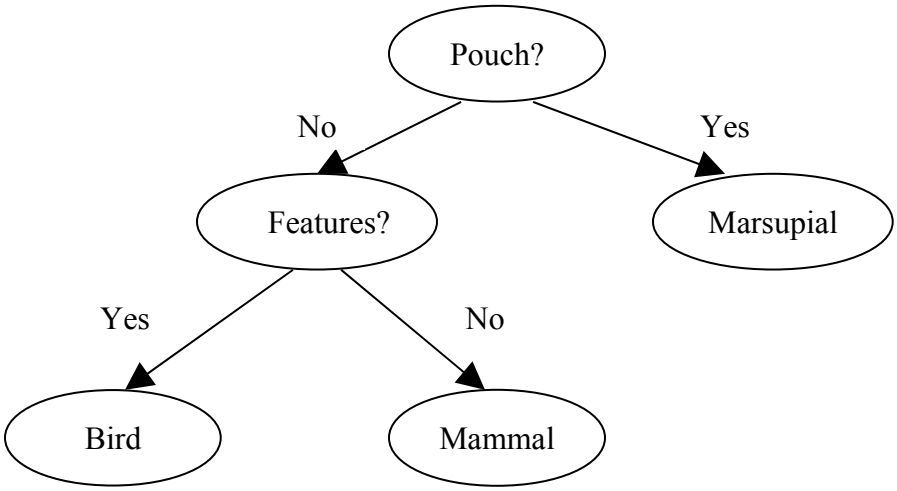


Figure 3.1 A decision tree for the data in Table 3.1

Decision Tree is a model that is both predictive and descriptive. A decision Tree is a tree that displays relationships found in the training data. The tree consists of zero or more internal nodes and one or more leaf nodes with each internal node being a decision node having two or more child nodes. Using the training data, the decision tree method generates a tree that consists of nodes that are rules, very similar to those used in "20 questions" to determine the class of am object after the training is completed. Each node of the tree represents a choice between a number of alternatives and each leaf node represents a classification or a decision. The training process that generates the tree is called *Induction.*

Normally, the complexity of a decision tree increases as the number of attributes increases, although in some situations it has been found that only a small number of

attributes can determine the class to which an object belongs and the rest of the attributes have little or no impact.

3.3 BUILDING A DECISION TREE – THE TREE INDUCTION ALGORITHM

The decision tree algorithm is a relatively simple top-down greedy algorithm. The aim of the algorithm is to build a tree that has leaves that are as homogeneous as possible. The major step of the algorithm is to continue to divide leaves that are not homogeneous into leaves that are as homogeneous as possible until no further division is possible. The decision tree algorithm is given below:

1. Let the set of training data be S. If some of the attributes are continuously-valued, they should be discretized. For example, age values may be binned into the following categories (under 18), (18-40), (41-65) and (over 65) and transformed into A, B, C and D or more descriptive labels may be chosen. Once that is done, put all of S in a single tree node.
2. If all instances in S are in the same class, then stop.
3. Split the next node by selection of an attribute A from amongst the independent attributes that best divides or splits the objects in the node into subsets and create a decision tree node.
4. Split the node according to the values of A.
5. Stop if either of the following conditions is met, otherwise continue with step 3.
    (a) If this partition divides the data into subsets that belong to a single class and no other node needs splitting.
    (b) If there are no remaining attributes on which the sample may be further divided.

In the decision tree algorithm, decisions are made locally and the algorithm at no stage tries to find a globally optimum tree.

3.4 SPLIT ALGORITHM BASED ON INFORMATION THEORY

One of the techniques for selecting an attribute to split a node is based on the concept of information theory or entropy. The concept is quite simple, although often difficult to understand for many. It is based on Claude Shannon's idea that if you have uncertainty then you have information and if there is no uncertainty there is no information. For example, if a coin has a head on both sides, then the result of tossing it does not product any information but if a coin is normal with a head and a tail then the result of the toss provides information.

Essentially, information is defined as $-p_i \log p_i$ where p_i is the probability of some event. Since the probability p_i is always less than 1, $\log p_i$ is always negative and $-p_i \log p_i$ is always positive. For those who cannot recollect their high school mathematics, we note that log of 1 is always zero whatever the base, the log of any number greater than 1 is always positive and the log of any number smaller than 1 is always negative. Also,

$$\log_2(2) = 1$$

$$\log_2(2^n) = n$$

$$\log_2(1/2) = -1$$

$$\log_2(1/2^n) = -n$$

Information of any event that is likely to have several possible outcomes is given by

$$I = \sum_i (-p_i \log p_i)$$

Consider an event that can have one of two possible values. Let the possibilities of the two values be p_1 and p_2. Obviously if p_1 is 1 and p_2 is zero, then there is no information in the outcome and I=0. If p_1=0.5, then the information is

$$I = -0.5 \log(0.5) - 0.5 \log(0.5)$$

This comes out to 1.0 (using log base 2) is the maximum information that you can have for an event with two possible outcomes. This is also called entropy and is in effect a measure of the minimum number of bits required to encode the information.

If we consider the case of a die (singular of dice) with six possible outcomes with equal probability, then the information is given by:

$$I = 6(-1/6) \log(1/6)) = 2.585$$

Therefore three bits are required to represent the outcome of rolling a die. Of course, if the die was loaded so that there was a 50% or a 75% chance of getting a 6, then the information content of rolling the die would be lower as given below. Note that we assume that the probability of getting any of 1 to 5 is equal (that is, equal to 10% for the 50% case and 5% for the 75% case).

50%: $I = 5(-0.1) \log(0.1)) – 0.5 \log(0.5) = 2.16$

75%: $I = 5(-0.05) \log(0.05)) – 0.75 \log(0.75) = 1.39$

Therefore we will need three bits to represent the outcome of throwing a die that has 50% probability of throwing a six but only two bits when the probability is 75%.

Information Gain

Information gain is a measure of how good an attribute is for predicting the class of each of the training data. We will select the attribute with the highest information gain as the next split attribute.

Perhaps the term information gain is somewhat confusing because what we really mean is that information gain is a measure of reduction in uncertainty once the value of an attribute is known. If the uncertainty is reduced by a large amount, knowing the value of the attribute has provided a lot of information and thus we have a large information gain.

Assume there are two classes, P and N, and let the set of training data S (with a total number of objects s) contain p elements of class P and n elements of class N. The amount of information is defined as

$$I = -(n/s) \log(n/s) – (p/s) \log (p/s)$$

Obviously if p=n, I is equal to 1 and if p=s then I=0. Therefore if there was an attribute for which almost all the objects had the same value (for example, gender when most

people are male), using the attribute would lead to no information gain (that is, not reduce uncertainty) because for gender = female there will be almost no objects while gender = male would have almost all the objects even before we knew the gender. If on the other hand, an attribute divided the training sample such that gender = female resulted in objects that all belong to Class A and gender = male all belong to Class B, then uncertainty has been reduced to zero and we have a large information gain.

We define information gain for sample S using attribute A as follows:

$$\text{Gain}(S, A) = I - \sum_{i \text{Evalues}(A)} (t_i/s)\, I_i$$

I is the information before the split and $\sum_{i\text{Evalues}(A)} (t_i/s)\, I_i$ is the sum of information after the split where I_i is the information of node I and t_i is the number of objects in node i. Thus the total information after the split is $\sum_{i\text{Evalues}(A)} (t_i/s)\, I_i$, which is the sum of the information of each node.

Once we have computed the information gain for every remaining attribute, the attribute with the highest information gain is selected.

Example 3.1 – Using the Information Measure

We consider an artificial example of building a decision tree classification model to classify bank loan applications by assigning applications to one of three risk classes (Table 3.2).

Table 3.2 Training data for Example 3.1

Owns Home?	Married	Gender	Employed	Credit Rating	Risk Class
Yes	Yes	Male	Yes	A	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	B	C
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	B	C

Yes	Yes	Female	Yes	B	A
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	A	C
Yes	Yes	Female	Yes	A	C

There are 10 (s =10) samples and three classes. The frequencies of these classes are:

A = 3

B = 3

C = 4

Information in the data due to uncertainty of outcome regarding the risk class each person belongs to is given by

$$I = -(3/10)\log(3/10) - (3/10)\log(3/10) - (4/10)\log(4/10) = 1.57$$

Let us now consider using each attribute in turn as a candidate to split the sample.

1. Attribute "Owns Home"

Value = Yes. There are five applicants who own their home. They are in classes A=1, B=2, C=2.

Value = No. There are five applicants who do not own their home. They are in classes A=2, B=1, C=2.

Given the above values, it does not appear as if this attribute will reduce the uncertainty by much. Let us compute the information gain by using this attribute. We divide persons into those who own their home and those who do not. Computing information for each of these two subtrees,

$$I(\text{Yes}) = I(y) = -(1/5)\log(1/5) - (2/5)\log(2/5) - (2/5)\log(2/5) = 1.52$$

$$I(\text{No}) = I(n) = -(2/5)\log(2/5) - (1/5)\log(1/5) - (2/5)\log(2/5) = 1.52$$

Total information of the two subtrees = $0.5I(y) + 0.5I(n) = 1.52$

2. Attribute "Married"

There are five applicants who are married and five who are not.

Value = Yes has A = 0, B = 1, C = 4, total 5

Value = No has A = 3, B = 2, C = 0, total 5

Looking at the values above, it appears that this attribute will reduce the uncertainty by more than the last attribute. Computing the information gain by using this attribute, we have

$$I(y) = -(1/5) \log(1/5) – (4/5) \log(4/5) = 0.722$$

$$I(n) = -(3/5) \log(3/5) – (2/5) \log(2/5) = 0.971$$

Information of the subtrees = $0.5I(y) + 0.5I(n) = 0.846$

3. Attribute "Gender"

There are three applicants who are male and seven are female.

Value = Male has A = 0, B = 3, C = 0, total 3

Value = Female has A = 3, B = 0, C = 4, total 7

The values above show that the uncertainty is reduced even more by using this attribute since for Value = Male we have only one class. Let us compute the information gain by using this attribute.

$$I(Male) = 0$$

$$I(Female) = -(3/7) \log(3/7) – (4/7) \log(4/7) = 0.985$$

Total information of the subtrees = $0.3I(Male) + 0.7I(Female) = 0.69$

4. Attribute "Employed"

There are eight applicants who are employed and two that are not.

Value = Yes has A = 3 B = 1, C = 4, total 8

Value = No has A = 0, B = 2, C = 0, total 2

The values above show that this attribute will reduce uncertainty but most attribute values are Yes while the No value leads to only one class. Computing the information gain by using this attribute, we have

$$I(y) = -(3/8) \log(3/8) – (1/8) \log(1/8) – (4/8) \log(4/8) = 1.41$$

$$I(n) = 0$$

Total information of the subtrees = $0.8I(y) + 0.2I(n) = 1.12$

5. Attribute "Credit Rating"

There are five applicants who have credit rating A and five that have B.

Value = A has A = 2, B = 1, C = 2, total 5

Value = B has A = 1, B = 2, C = 2, total 5

Looking at the values above, we can see that this is like the first attribute that does not reduce uncertainty by much. The information gain for this attribute is the same as for the first attribute.

$$I(A) = -(2/5)\log(2/5) - (1/5)\log(1/5) - (2/5)\log(2/5) = 1.52$$

$$I(B) = -(1/5)\log(1/5) - (2/5)\log(2/5) - (2/5)\log(2/5) = 1.52$$

Total information of the subtrees = $0.5I(A) + 0.5I(B) = 1.52$

The values for information gain can now be computed. See Table 3.3.

Table 3.3 Information gain for the five attributes

Potential split attribute	*Information before split*	*Information after split*	*Information gain*
Owns Home	1.57	1.52	0.05
Married	1.57	0.85	0.72
Gender	1.57	0.69	0.88
Employed	1.57	1.12	0.45
Credit Rating	1.57	1.52	0.05

Hence the largest information gain is provided by the attribute "Gender" and that is the attribute that is used for the split.

Now we can reduce the data by removing the attribute Gender and removing the class B since all Class B have Gender = Male. See Table 3.4.

Table 3.4 Data after removing attribute "Gender" and Class B

Owns Home?	*Married*	*Employed*	*Credit Rating*	*Risk*
Class				
No	No	Yes	A	A
Yes	Yes	Yes	B	C
No	Yes	Yes	B	C
No	No	Yes	B	A
Yes	No	Yes	A	A
No	Yes	Yes	A	C
Yes	Yes	Yes	A	C

The information in this data of two classes due to uncertainty of outcome regarding the class each person belongs to is given by

$$I = -(3/7)\log(3/7) - (4/7)\log(4/7) = 1.33$$

Let us now consider each attribute in turn as a candidate to split the sample.

1. Attribute "Owns Home"

Value = Yes. There are three applicants who own their home. They are in classes A=1 and C=2.

Value = No. There are four applicants who do not own their home, who are in classes A=2, and C=2.

Given the above values, it does not appear as if this attribute will reduce the uncertainty by much. Computing information for each of these two subtrees,

$$I(Yes) = I(y) = -(1/3)\log(1/3) - (2/3)\log(2/3) = 0.92$$
$$I(No) = I(n) = -(2/4)\log(2/4) - (2/4)\log(2/4) = 1.00$$

Total information of the two subtrees = $(3/7)I(y) + (4/7)I(n) = 0.96$

2. Attribute "Married"

There are four applicants who are married and three that are not.

Value = Yes has A = 0, C = 4, total 4

Value = No has A = 3, C = 0, total 3

Looking at the values above, it appears that this attribute will reduce the uncertainty by more than the last attribute since for each value the persons belong to only one class and therefore information is zero. Computing the information gain by using this attribute, we have

$$I(y) = -(4/4)\log(4/4) = 0.00$$

$$I(n) = -(3/3)\log(3/3) = 0.00$$

Information of the subtrees = 0.00

There is no need to consider other attributes now, since no other attribute can be better. The split attribute therefore is "Married" and we now obtain the following decision tree in Figure 3.2 which concludes this very simple example. It should be noted that a number of attributes that were available in the data were not required in classifying it.

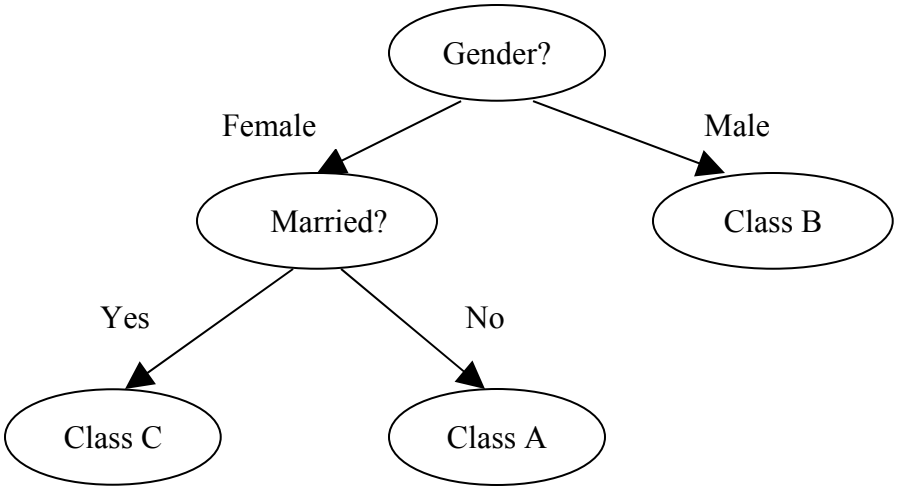


Figure 3.2 Decision tree for Example 3.1

3.5 SPLIT ALGORITHM BASED ON THE GINI INDEX

Another commonly used split approach is called the Gini index which is used in the widely used packages CART and IBM Intelligent Miner.

Figure 3.3 shows the Lorenz curve which is the basis of the Gini Index. The index is the ratio of the area between the Lorenz curve and the 45-degree line to the area under 45-degree line. The smaller the ratio, the less is the area between the two curves and the more evenly distributed is the wealth. When wealth is evently distributed, asking any person about his/her wealth provides no information at all since every person has the same wealth while in a situation where wealth is very unevenly distributed finding out how much wealth a person has provides information because of the uncertainty of wealth distribution.

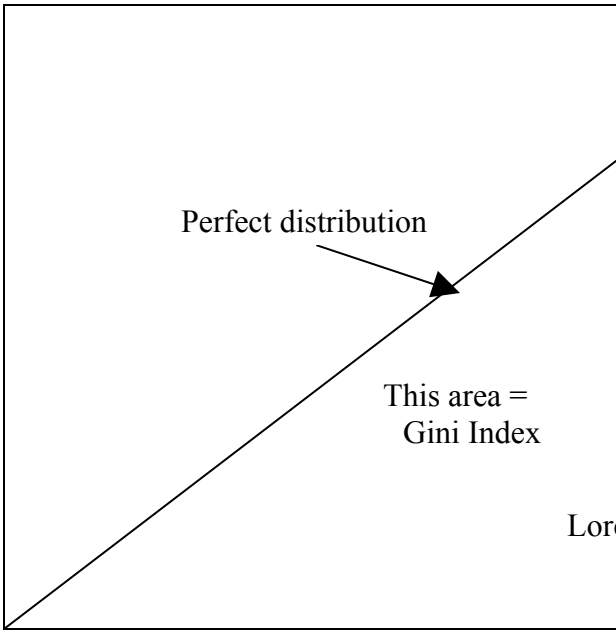


Figure 3.3 Lorenz curve.

Example 3.2 – Using the Gini Index

We use the same example (Table 3.2) as we have used before to illustrate the Gini Index.

Owns Home?	*Married*	*Gender*	*Employed*	*Credit Rating*	*Risk Class*

Yes	Yes	Male	Yes	A	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	B	C
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	B	C
Yes	Yes	Female	Yes	B	A
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	A	C
Yes	Yes	Female	Yes	A	C

There are 10 (s =10) samples and three classes. The frequencies of these classes are:

A = 3

B = 3

C = 4

The Gini index for the distribution of application of applicants in the three classes is

$$G = 1 - (3/10)^2 - (3/10)^2 - (4/10)^2 = 0.66$$

Let us now consider using each of the attributes to split the sample.

1. Attribute "Owns Home"

Value = Yes. There are five applicants who own their home. They are in classes A=1, B=2, C=2.

Value = No. There are five applicants who do not own their home. They are in classes A=2, B=1, C=2.

Using this attribute will divide objects into those who own their home and those who do not. Computing the Gini index for each of these two subtrees,

$$G(y) = 1 - (1/5)^2 - (2/5)^2 - (2/5)^2 = 0.64$$

$$G(n) = G(y) = 0.64$$

Total value of Gini Index = G = 0.5G(y) + 0.5G(n) = 0.64

2. Attribute "Married"

There are five applicants who are married and five that are not.

Value = Yes has A = 0, B = 1, C = 4, total 5

Value = No has A = 3, B = 2, C = 0, total 5

Looking at the values above, it appears that this attribute will reduce the uncertainty by more than the last attribute. Computing the information gain by using this attribute, we have

$$G(y) = 1 - (1/5)^2 - (4/5)^2 = 0.32$$

$$G(n) = 1 - (3/5)^2 - (2/5)^2 = 0.48$$

Total value of Gini Index = G = 0.5G(y) + 0.5G(n) = 0.40

3. Attribute "Gender"

There are three applicants who are male and seven are female.

Value = Male has A = 0, B = 3, C = 0, total 3

Value = Female has A = 3, B = 0, C = 4, total 7

G(Male) = 1 -1 = 0

$G(Female) = 1 - (3/7)^2 - (4/7)^2 = 0.511$

Total value of Gini Index = G = 0.3G(Male) + 0.7G(Female) = 0.358

4. Attribute "Employed"

There are eight applicants who are employed and two that are not.

Value = Yes has A = 3 B = 1, C = 4, total 8

Value = No has A = 0, B = 2, C = 0, total

$G(y) = 1 - (3/8)^2 - (1/8)^2 - (4/8)^2 = 0.594$

G(n) = 0

Total value of Gini Index = G = 0.8G(y) + 0.2G(n) = 0.475

5. Attribute "Credit Rating"

There are five applicants who have credit rating A and five that have B.

Value = A has A = 2, B = 1, C = 2, total 5

Value = B has A = 1, B = 2, C = 2, total 5

$G(A) = 1 - 2(2/5)^2 - (1/5)^2 = 0.64$

$G(B) = G(A)$

Total value of Gini Index = $G = 0.5G(A) + 0.5G(B) = 0.64$

Table 3.4 summarizes the values of the Gini Index obtained for the following five attributes:

Owns Home	Employed
Married	Credit Rating
Gender	

Table 3.4 Gini Index for the five attributes

Attribute	*Gini Index before split*	*Gini Index after split*	*Gain*
Owns Home	0.66	0.64	0.02
Married	0.66	0.40	0.26
Gender	0.66	0.358	0.302
Employed	0.66	0.475	0.185
Credit Rating	0.66	0.64	0.02

The attribute with the largest reduction in the Gini Index is selected as the split attribute. So the split attribute is Gender.

3.6 OVERFITTING AND PRUNING

The decision tree building algorithm given earlier continues until either all leaf nodes are single class nodes or no more attributes are available for splitting a node that has objects of more than one class. When the objects being classified have a large number of

attributes and a tree of maximum possible depth is built, the tree quality may not be high since the tree is built to deal correctly with the training set. In fact, in order to do so, it may become quite complex, with long and very uneven paths. Some branches of the tree may reflect anomalies due to noise or outliers in the training samples. Such decision trees are a result of overfitting the training data and may result in poor accuracy for unseen samples.

According to the Occam's razor principle (due to the medieval philosopher William of Occam) it is best to posit that the world is inherently simple and to choose the simplest model from similar models since the simplest model is more likely to be a better model. We can therefore "shave off" nodes and branches of a decision tree, essentially replacing a whole subtree by a leaf node, if it can be established that the expected error rate in the subtree is greater than that in the single leaf. This makes the classifier simpler. A simpler model has less chance of introducing inconsistencies, ambiguities and redundancies.

Pruning is a technique to make an overfitted decision tree simpler and more general.

There are a number of techniques for pruning a decision tree by removing some splits and subtrees created by them. One approach involves removing branches from a "fully grown" tree to obtain a sequence of progressively pruned trees. The accuracy of these trees is then computed and a pruned tree that is accurate enough and simple enough is selected. It is advisable to use a set of data different from the training data to decide which is the "best pruned tree".

Another approach is called pre-pruning in which tree construction is halted early. Essentially a node is not split if this would result in the goodness measure of the tree falling below a threshold. It is, however, quite difficult to choose an appropriate threshold.

3.7 DECISION TREE RULES

The decision tree method is a popular and relatively simple supervised classification method that involves each node of the tree specifying a test of some attribute and each branch from the node corresponding to one of the values of the attribute. Each path from

the root to leaf of the decision tree therefore consists of attribute tests, finally reaching a leaf that describes the class. The popularity of decision trees is partly due to the ease of understanding the rules that the nodes specify. One could even use the rules specified by decision tree to retrieve data from a relational database satisfying the rules using SQL.

There are a number of advantages in converting a decision tree to rules. Decision rules make it easier to make pruning decisions since it is easier to see the context of each rule. Also, converting to rules removes the distinction between attribute tests that occur near the roof of the tree and they are easier for people to understand.

IF-THEN rules may be derived based on the various paths from the root to the leaf nodes. Although the simple approach will lead to as many rules as the leaf nodes, rules can often be combined to produce a smaller set of rules. For example:

If Gender = “Male” then Class = B

If Gender = “Female” and Married = “Yes” then Class = C, else Class = A

Once all the rules have been generated, it may be possible to simplify the rules. Rules with only one antecedent (e.g. if Gender=”Male” then Class=B) cannot be further simplified, so we only consider those with two or more antecedents. It may be possible to eliminate unnecessary rule antecedents that have no effect on the conclusion reached by the rule. Some rules may be unnecessary and these may be removed. In some cases a number of rules that lead to the same class may be combined.

3.8 NAÏVE BAYES METHOD

The Naïve Bayes method is based on the work of Thomas Bayes. Bayes was a British minister and his theory was published only after his death. It is a mystery what Bayes wanted to do with such calculations.

Bayesian classification is quite different from the decision tree approach. In Bayesian classification we have a hypothesis that the given data belongs to a particular class. We then calculate the probability for the hypothesis to be true. This is among the most practical approaches for certain types of problems. The approach requires only one scan of the whole data. Also, if at some stage there are additional training data then each

training example can incrementally increase/decrease the probability that a hypothesis is correct.

Before we define the Bayes theorem, we will define some notation. The expression P(A) refers to the probability that event A will occur. P(A|B) stands for the probability that event A will happen, given that event B has already happened. In other words P(A|B) is the conditional probability of A based on the condition that B has already happened. For example, A and B may be probabilities of passing another course B respectively. P(A|B) then is the probability of passing A when we know that B has been passed.

Now here is the Bayes theorem:

$$P(A\backslash B) = P(B|A)P(A)/P(B)$$

Once might wonder where did this theorem come from. Actually it is rather easy to derive since we know the following:

$$P(A|B) = P(A \& B)/P(B)$$

and

$$P(B|A) = P(A \& B)/P(A)$$

Diving the first equation by the second gives us the Bayes' theorem.

Continuing with A and B being courses, we can compute the conditional probabilities if we knew what the probability of passing both courses was, that is P(A & B), and what the probabilities of passing A and B separately were. If an event has already happened then we divide the joint probability P(A & B) with the probability of what has just happened and obtain the conditional probability.

If we consider X to be an object to be classified then Bayes' theorem (3.1) may be read as giving the probability of it belonging to one of the classes C_1, C_2, C_3 etc by calculating P(Ci|X). Once these probabilities have been computed for all the classes, we simply assign X to the class that has the highest conditional probability.

Let us now consider how probabilities P(Ci|X) may be calculated. We have

$$P(Ci|X) = [P(X|Ci)P(Ci)]/P(X)$$

- P(Ci|X) is the probability of the object X belonging to class Ci.
- P(X|Ci) is the probability of obtaining attribute values X if we know that it belongs to class Ci.
- P(Ci) is the probability of any object belonging to class Ci without any other information.
- P(X) is the probability of obtaining attribute values X whatever class the object belongs to.

Given the attribute values X, what probabilities in the formula can we compute? The probabilities we need to compute are P(X|Ci), P(Ci) and P(X). Actually the denominator P(X) is independent of Ci and is not required to be known since we are interested only in comparing probabilities P(Ci|X). Therefore we only need to compute P(X|Ci) and P(Ci) for each class. Comparing P(Ci) is rather easy since we count the number of instances of each class in the training data and divide each by he total number of instances. This may not be the most accurate estimation of P(Ci) but we have very little information, the training sample, and we have no other information to obtain a better estimate. This estimate will be reasonable if the training sample was large and was randomly chosen.

To compute P(X|Ci) we use a naïve approach (that is why it is called the Naïve Bayes model) by assuming that all attributes of X are independent which is often not true.

Using the independence of attributes assumption and based on the training data, we compute an estimate of the probability of obtaining the data X that we have by estimating the probability of each of the attribute values by counting the frequency of those values for class Ci.

We then determine the class allocation of X by computing [P(X|Ci)P(Ci)] for each of the classes and allocating X to the class with the largest value.

The beauty of the Bayesian approach is that the probability of the dependent attribute can be estimated by computing estimates of the probabilities of the independent attributes.

We should also note that it is possible to use this approach even if values of all the independent attributes are not known since we can still estimate the probabilities of the attribute values that we know. This is a significant advantage of the Bayesian approach.

Example 3.3 – Naïve Bayes Method

Once again we go back to the example in Table 3.2 that we have used before.

Owns Home?	*Married*	*Gender*	*Employed*	*Credit Rating*	*Risk Class*
Yes	Yes	Male	Yes	A	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	B	C
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	B	C
Yes	Yes	Female	Yes	B	A
Yes	Yes	Male	No	B	B
Yes	Yes	Female	Yes	A	A
Yes	Yes	Female	Yes	A	C
Yes	Yes	Female	Yes	A	C

There are 10 (s =10) samples and three classes. The frequencies of these classes are:

Credit risk Class A = 3

Credit risk Class B = 3

Credit risk Class C = 4

The prior probabilities are obtained by dividing these frequencies by the total number in the training data (that is, 10)

$$P(A) = 0.3, \quad P(B) = 0.3, \quad \text{and} \quad P(C) = 0.4$$

If the data that is presented to us is {yes, no, female, yes, A} for the five attributes, we can compute the posterior probability for each class as noted earlier. For example:

$$P(X|Ci) = P(\{yes, no, female, yes, A\}|Ci) = P(\text{Owns Home} = yes|Ci) \times P(\text{Married} = no|Ci)$$

$$\times P(\text{Gender} = female|Ci) \times P(\text{Employed} = yes|Ci) \times P(\text{Credit Rating} = A|Ci)$$

Using expressions like that given above, we are able to compute the three posterior probabilities for the three classes, namely that the person with attribute values X

has credit risk class A or class B or class C. We compute P(X|Ci)P(Ci) for each of the three classes given P(A) = 0.3, P(B) = 0.3 and P(C) = 0.4 and these values are the basis for comparing the three classes.

To compute P(X|Ci) = P({yes, no, female, yes, A}|Ci) for each of the classes, we need the following probabilities for each:

P(Owns Home = yes|Ci)
P(Married = no|Ci)
P(Gender = female|Ci)
P(Employed = yes|Ci)
P(Credit Rating = A|Ci)

We order the data by risk class to make it convenient.

Given the estimates of the probabilities, we can compute the posterior probabilities as

P(X|A) = 2/9
P(X|B) = 0
P(X|C) = 0

Therefore the values of P(X|Ci)P(Ci) are zero for Classes B and C and 0.3 x 2/9 = 0.0666. Therefore X is assigned to Class A. It is unfortunate that in this example two of the probabilities came out to be zero. This is most unlikely in practice.

Bayes' theorem assumes that all attributes are independent and that the training sample is a good sample to estimate probabilities. These assumptions are not always true in practice, as attributes are often correlated but in spite of this the Naïve Bayes method performs reasonably well. Other techniques have been designed to overcome this limitation. One approach is to use Bayesian networks that combine Bayesian reasoning with causal relationships between attributes.

3.9 ESTIMATING PREDICTIVE ACCURACY OF CLASSIFICATION METHODS

The accuracy of a classification method is the ability of the method to correctly determine the class of a randomly selected data instance. It may be expressed as the probability of correctly classifying unseen data. Estimating the accuracy of a supervised classification method can be difficult if only the training data is available and all of that data has been used in building the model. In such situations, overoptimistic predictions are often made regarding the accuracy of the model.

Methods for estimating the accuracy of a classification method:

1. **Holdout Method**

The holdout method (sometimes called the test sample method) requires a training set and a test set. The sets are mutually exclusive. It may be that only dataset is available which has been divided into two subsets (perhaps 2/3 and 1/3), the training subset and the test or holdout subset. Once the classification method produces the model using the training set, the test set can be used to estimate the accuracy. Interesting questions arise in this estimation since a larger training set would produce a better classifier, while a larger test set would produce a better estimate of the accuracy. A balance must be achieved. Since none of the test data is used in training, the estimate is not biased, but a good estimate is obtained only if the test data is used in training set are large enough and representative of the whole population.

2. **Random Sub-sampling Method**

Random sub-sampling is very much like the holdout method except that it does not rely on a single text set. Essentially, the holdout estimation is repeated several times and the accuracy estimate is obtained by computing the mean of the several trails. Random sub-sampling is likely to produce better error estimates than those by the holdout method.

3. **k-fold Cross-validation Method**

In k-fold cross-validation, the available data is randomly divided into k disjoint subsets of approximately equal size. One of the subsets is then used as the test set and the remaining k-1 sets are used for building the classifier. The test set is then used to estimate the accuracy. This is done repeatedly k times so that each subset is used as a test subset once. The accuracy estimate is then the mean of the estimates for each of the classifiers. Cross-validation has been tested extensively and has been tested extensively and has been found to generally work well when sufficient data is available. A value of 10 for k has been found to be adequate and accurate.

4. Leave-one-out Method

Leave-one-out is a simpler version of k-fold cross-validation. In this method, one of the training samples is taken out and the model is generated using the remaining training data. Once the model is built, the one remaining sample is used for testing and the result is coded as 1 or 0 depending if it was classified correctly or not. The average of such results provides an estimate of the accuracy. The leave-one-out method is useful when the dataset is small. For large training datasets, leave-one-out can become expensive since many iterations are required. Leave-one-out is unbiased but has high variance and is therefore not particularly reliable.

5. Bootstrap Method

In this method, given a dataset of size n, a bootstrap sample is randomly selected uniformly with replacement (that is, a sample may be selected more than once) by sampling n times and used to build a model. It can be shown that only 63.2% of these samples are unique. The error in building the model is estimated by using the remaining 36.8% of objects that are not in the bootstrap sample. The final error is than computed as 0.632 and 0.368 are based on the assumption that if there were n samples available initially from which n samples with replacement were randomly selected for training data then the expected percentage of unique samples in the training data would be 0.632 or 63.2% and the number of remaining unique objects used in the test data would be 0.368 or 36.8% of the initial sample. This is repeated and the average of error estimates is

obtained. The bootstrap method is unbiased and, in contrast to leave-one-out, has low variance but much iteration is needed for good error estimates if the sample is small, say 25 or less.

3.10 IMPROVING ACCURACY OF CLASSIFICATION METHODS

Bootstrapping, bagging and boosting are techniques for improving the accuracy of classification results. They have been shown to be very successful for certain models, for example, decision trees. All three involve combining several classification results from the same training data that has been perturbed in some way. The aim of building several decision trees by using training data that has been perturbed is to find out how these decision trees differ from those that have been obtained earlier.

The bootstrapping method can be shown that all the samples selected are somewhat different from each other since, on the average, only 63.2% of the objects in them are unique. The bootstrap samples are then used for building decision trees which are then combined to form a single decision tree.

Bagging (the name is derived form **B**ootstrap and **agg**regat**ing**) combines classification results from multiple models or results of using the same method on several different sets of training data. Bagging may also be used to improve the stability and accuracy of a complex classification model with limited training data by using sub-samples obtained by resampling, with replacement, for generating models.

Bagging essentially involves simple voting (with no weights), so the final model is the one that is predicted by the majority of the trees. The different decision trees obtained during bagging should not be so different if the training data is good and large enough. If the trees obtained are very different, it only indicates instability, perhaps due to the training data being random. In such a situation, bagging will often provide a better result than using the training data only once to build a decision tree.

There is a lot of literature available on bootstrapping, bagging, and boosting. This brief introduction only provides a glimpse into these techniques but some of the points made in the literature regarding the benefits of these methods are:

- These techniques can provide a level of accuracy that usually cannot be obtained by a large single-tree model.
- Creating a single decision tree from a collection of trees in bagging and boosting is not difficult.
- These methods can often help in avoiding the problem of overfitting since a number of trees based on random samples are used.
- Boosting appears to be on the average better than bagging although it is not always so. On some problems bagging does better than boosting.

3.11 OTHER EVALUATION CRITERIA FOR CLASSIFICATION METHODS

The criteria for evaluation of classification methods are as follows:

1. Speed
2. Robustness
3. Scalability
4. Interpretability
5. Goodness of the model
6. Flexibility
7. Time complexity

Speed

Speed involves not just the time or computation cost of constructing a model (e.g. a decision tree), it also includes the time required to learn to use the model. Obviously, a user wishes to minimize both times although it has to be understood that any significant data mining project will take time to plan and prepare the data. If the problem to be

solved is large, a careful study of the methods available may need to be carried out so that an efficient classification method may be chosen.

Robustness

Data errors are common, in particular when data is being collected from a number of sources and errors may remain even after data cleaning. It is therefore desirable that a method be able to produce good results in spite of some errors and missing values in datasets.

Scalability

Many data mining methods were originally designed for small datasets. Many have been modified to deal with large problems. Given that large datasets are becoming common, it is desirable that a method continues to work efficiently for large disk-resident databases as well.

Interpretability

A data mining professional is to ensure that the results of data mining are explained to the decision makers. It is therefore desirable that the end-user be able to understand and gain insight from the results produced by the classification method.

Goodness of the Model

For a model to be effective, it needs to fit the problem that is being solved. For example, in a decision tree classification, it is desirable to find a decision tree of the "right" size and compactness with high accuracy.

3.12 CLASSIFICATION SOFTWARE

A more comprehensive classification software list is available at kdnuggets site: (http://www.kdnuggets.com/software/classification.html).

- C4.5, version 8 of the "classic" decision-tree tool, developed by J. R. Quinlan (free, restricted distribution) is available at: http://www.rulequest.com/Personal/

- CART 5.0 and TreeNet from Salford Systems are the well-known decision tree software packages. TreeNet provides boosting. CART is the decision tree software. The packages incorporate facilities for data pre-processing and predictive modeling including bagging and arcing. For more details visit: http://www.salford-systems.com/

- DTREG, from a company with the same name, generates classification trees when the classes are categorical, and regression decision trees when the classes are numerical intervals, and finds the optimal tree size. In both cases, the attribute values may be discrete or numerical. Software modules TreeBoost and Decision Tree Forest generate an ensemble of decision trees. In TreeBoost, each tree is generated based on input from a previous tree while Decision Tree Forest generates an ensemble independently of each other. The ensemble is then combined. For more details visit: http://www.dtreg.com/

- SMILES provides new splitting criteria, non-greedy search, new partitions, extraction of several and different solutions. It offers a quite effective handling of (misclassification and test) costs. SMILES also uses boosting and cost-sensitive learning. For more details visit: http://www.dsic.upv.es/~flip/smiles/

- NBC: a Simple Naïve Bayes Classifier. Written in awk. For more details visit: http://scant.org/nbc/nbc.html

CONCLUSION

In this chapter we discussed supervised classification, which is an extensively studied problem in statistics and machine learning. Classification is probably the most widely used data mining technique.

We described the decision tree approach to classification using information measure and Gini index for splitting attributes. In the decision tree approach, decisions are made locally by considering one attribute at a time thus considering the most important attribute first. We discussed decision tree pruning and testing. Another classification method, the Naïve Bayes method, has been described.

REVIEW QUESTIONS

1. What kind of data is the decision tree method most suitable for?
2. Briefly outline the major steps of the algorithm to construct a decision tree.
3. Assume that we have 10 training samples. There are four classes A, B, C and D. Compute the information in the samples for the five training datasets given below (each row is a dataset and each dataset has 10 objects) when the number of samples in each class are:

Class	A	B	C	D
Dataset 1	1	1	1	7
Dataset 2	2	2	2	4
Dataset 3	3	3	3	1
Dataset 4	1	2	3	4
Dataset 5	0	0	1	9

Use the log values from the following table:

p	log(p)	-p log(p)
0.1	-3.32	0.332
0.2	-2.32	0.464
0.3	-1.74	0.521
0.4	-1.32	0.529

0.5	-1.00	0.500
0.6	-0.74	0.442
0.7	-0.51	0.360
0.8	-0.32	0.258
0.9	-0.15	0.137

---

4. What is overfitting of a decision tree? What problems can overfitting lead to?
5. List five criteria for evaluating the classification methods. Discuss them briefly.
6. Describe three methods of estimating accuracy of a classification method.
7. Explain the terms bootstrapping, bagging, and boosting for improving the accuracy of a classification method.
8. What is the difference between bootstrapping, bagging and boosting?

CHAPTER 4
CLUSTER ANALYSIS

Learning Objectives

1. Explain what cluster analysis is
2. Describe some desirable features of a cluster analysis method
3. Describe the types of cluster analysis techniques available
4. Describe the K-means method, a partitioning technique
5. Describe two hierarchical techniques – the Agglomerative method and the Divisive method.

4.1 WHAT IS CLUSTER ANALYSIS?

We like to organize observations or objects or things (e.g. plants, animals, chemicals) into meaningful groups so that we are able to make comments about the groups rather than individual objects. Such groupings are often rather convenient since we can talk about a small number of groups rather than a large number of objects although certain details are necessarily lost because objects in each group are not identical. A classical example of a grouping is the chemical periodic table where chemical elements are grouped into rows and columns such that elements adjacent to each other within a group have similar physical properties. For example, the elements in the periodic table are grouped as:

1. Alkali metals

2. Actinide series
3. Alkaline earth metals
4. Other metals
5. Transition metals
6. Nonmetals
7. Lanthanide series
8. Noble gases

Elements in each of these groups are similar but dissimilar to elements in other groups.

The aim of cluster analysis is exploratory, to find if data naturally falls into meaningful groups with small within-group variations and large between-group variation. Often we may not have a hypothesis that we are trying to test. The aim is to find any interesting grouping of the data. It is possible to define cluster analysis as an optimization problem in which a given function consisting of within cluster (intra-cluster) similarity and between clusters (inter-cluster) dissimilarly needs to be optimized. This function can be difficult to define and the optimization of any such function is a challenging task, clustering methods therefore only try to find an approximate or local optimum solution.

4.2 DESIRED FEATURES OF CLUSTER ANALYSIS

Given that there are a large number of cluster analysis methods on offer, we make a list of desired features that an ideal cluster analysis method should have. The list is given below:

1. ***(For large datasets) Scalability:*** Data mining problems can be large and therefore it is desirable that a cluster analysis method be able to deal with small as well as large problems gracefully. Ideally, the performance should be linear with the size of the data. The method should also scale well to datasets in which the number of attributes is large.

2. ***(For large datasets) Only one scan of the dataset:*** For large problems, the data must be stored on the disk and the cost of I/O from the disk can then

become significant in solving the problem. It is therefore desirable that a cluster analysis method not require more than one scan of the disk-resident data.

3. ***(For large datasets) Ability to stop and resume:*** When the dataset is very large, cluster analysis may require considerable processor time to complete the task. In such cases, it is desirable that the task be able to be stopped and then resumed when convenient.

4. ***Minimal input parameters:*** The cluster analysis method should not expect too much guidance from the user. A data mining analyst may be working with a dataset about which his/her knowledge is limited. It is therefore desirable that the user not be expected to have domain knowledge of the data and not be expected to possess insight into clusters that might exist in the data.

5. ***Robustness:*** Most data obtained from a variety of sources has errors. It is therefore desirable that a cluster analysis method be able to deal with noise, outliers and missing values gracefully.

6. ***Ability to discover different cluster shapes:*** Clusters come in different shapes and not all clusters are spherical. It is therefore desirable that a cluster analysis method be able to discover cluster shapes other than spherical. Some applications require that various shapes be considered.

7. ***Different data types:*** Many problems have a mixture of data types, for example, numerical, categorical and even textual. It is therefore desirable that a cluster analysis method be able to deal with not only numerical data but also Boolean and categorical data.

8. ***Result independent of data input order:*** Although this is a simple requirement, not all methods satisfy it. It is therefore desirable that a cluster

analysis method not be sensitive to data input order, whatever the order, the result of cluster analysis of the same data should be the same.

4.3 TYPES OF DATA

Datasets come in a number of different forms. The data may be quantitative, binary, nominal or ordinal.

1. **Quantitative (or numerical) data** is quite common, for example, weight, marks, height, price, salary, and count. There are a number of methods for computing similarity between quantitative data.
2. **Binary data** is also quite common, for example, gender, and marital status. Computing similarity or distance between categorical variables is not as simple as for quantitative data but a number of methods have been proposed. A simple method involves counting how many attribute values of the two objects are different amongst n attributes and using this as an indication of distance.
3. **Qualitative nominal data** is similar to binary data which may take more than two values but has no natural order, for example, religion, food or colours. For nominal data too, an approach similar to that suggested for computing distance for binary data may be used.
4. **Qualitative ordinal (or ranked) data** is similar to nominal data except that the data has an order associated with it, for example, grades A, B, C, D, sizes S, M, L, and XL. The problem of measuring distance between ordinal variables is different than for nominal variables since the order of the values is important. One method of computing distance involves transferring the values to numeric values according to their rank. For example grades, A, B, C, D could be transformed to 4.0, 3.0, 2.0 and 1.0 and then one of the methods in the next section may be used.

4.4 COMPUTING DISTANCE

Distance is well understood concept that has a number of simple properties.

1. Distance is always positive,

2. Distance from point x to itself is always zero.

3. Distance from point x to point y cannot be greater than the sum of the distance from x to some other point z and distance from z to y.

4. Distance from x to y is always the same as from y to x.

Let the distance between two points x and y (both vectors) be D(x,y). We now define a number of distance measures.

Euclidean distance

Euclidean distance or the L_2 norm of the difference vector is most commonly used to compute distances and has an intuitive appeal but the largest valued attribute may dominate the distance. It is therefore essential that the attributes are properly scaled.

$$\mathbf{D(x,y) = (\Sigma\ (x_i - y_i)^2\ )^{1/2}}$$

It is possible to use this distance measure without the square root if one wanted to place greater weight on differences that are large.

A Euclidean distance measure is more appropriate when the data is not standardized, but as noted above the distance measure can be greatly affected by the scale of the data.

Manhattan distance

Another commonly used distance metric is the Manhattan distance or the L_1 norm of the difference vector. In most cases, the results obtained by the Manhattan distance are similar to those obtained by using the Euclidean distance. Once again the largest-valued attribute can dominate the distance, although not as much as in the Euclidean distance.

$$\mathbf{D(x,y) = \Sigma\ |x_i - y_i\ |}$$

Chebychev distance

This distance metric is based on the maximum attribute difference. It is also called the L_∞ norm of the difference vector.

$$\mathbf{D(x,y) = Max\ |x_i - y_i\ |}$$

Categorical data distance

This distance measure may be used if many attributes have categorical values with only a small number of values (e.g. binary values). Let N be the total number of categorical attributes.

$$D(x,y) = (\text{ number of } x_i - y_i)/N$$

4.5 TYPES OF CLUSTER ANALYSIS METHODS

The cluster analysis methods may be divided into the following categories:

Partitional methods

Partitional methods obtain a single level partition of objects. These methods usually are based on greedy heuristics that are used iteratively to obtain a local optimum solution. Given n objects, these methods make $k \leq n$ clusters of data and use an iterative relocation method. It is assumed that each cluster has at least one object belongs to only one cluster. Objects may be relocated between clusters as the clusters are refined. Often these methods require that the number of clusters be specified apriori and this number usually does not change during the processing.

Hierarchical methods

Hierarchical methods obtain a nested partition of the objects resulting in a tree of clusters. These methods either start with one cluster and then split into smaller and smaller clusters (called divisive or top down) or start with each object in an individual cluster and then try to merge similar clusters into larger and larger clusters (called agglomerative or bottom up). In this approach, in contrast to partitioning, tentative clusters may be merged or split based on some criteria.

Density-based methods

In this class of methods, typically for each data point in a cluster, at lease a minimum number of points must exist within a given radius. Density-based methods can deal with

arbitrary shape clusters since the major requirement of such methods is that each cluster be a dense region of points surrounded by regions of low density.

Grid-based methods

In this class of methods, the object space rather than the data is divided into a grid. Grid partitioning is based on characteristics of the data and such methods can deal with non-numeric data more easily. Grid-based methods are not affected by data ordering.

Model-based methods

A model is assumed, perhaps based on a probability distribution. Essentially, the algorithm tries to build clusters with a high level of similarity within them and a low of similarity between them. Similarity measurement is based on the mean values and the algorithm tries to minimize the squared-error function.

A simple taxonomy of cluster analysis methods is presented in Figure 4.1

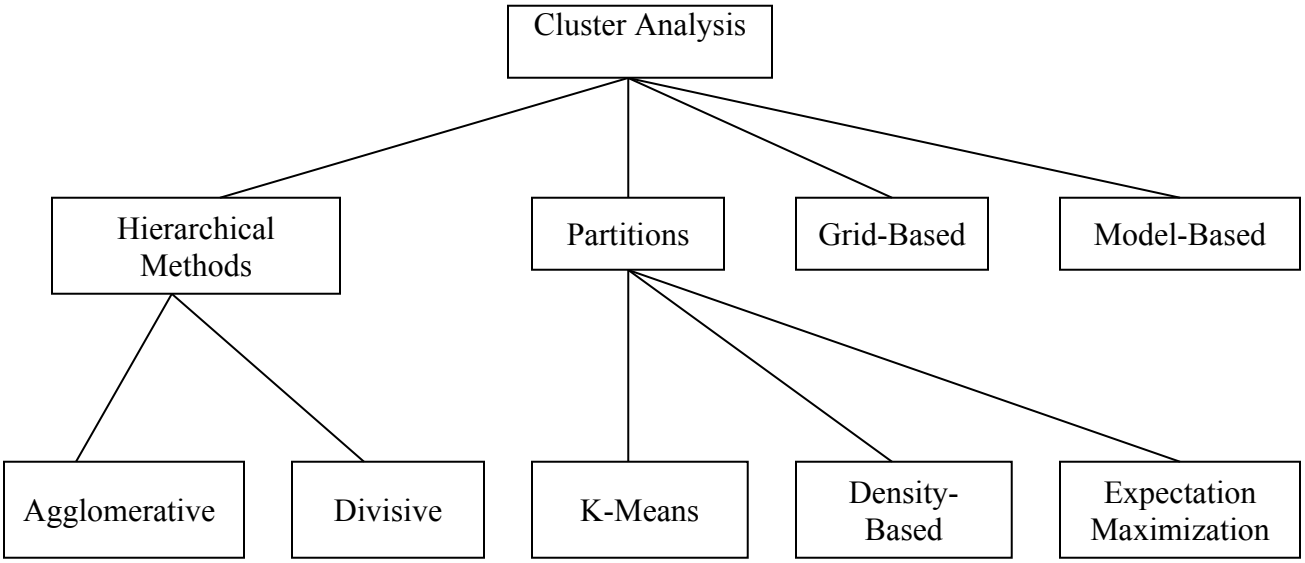


Figure 4.1 Taxonomy of cluster analysis methods

4.6 PARTITIONAL METHODS

Partitional methods are popular since they tend to be computationally efficient and are more easily adapted for very large datasets. The hierarchical methods tend to be computationally more expensive.

The aim of partitional methods is to reduce the variance within each cluster as much as possible and have large variance between the clusters. Since the partitional methods do not normally explicitly control the inter-cluster variance, heuristics (e.g. choosing seeds as far apart as possible) may be used for ensuring large inter-cluster variance. One may therefore consider the aim to be minimizing a ratio like a/b where a is some measure of within cluster variance and b is some measure of between cluster variation.

The K-Means Method

K-Means is the simplest and most popular classical clustering method that is easy to implement. The classical method can only be used if the data about all the objects is

located in the main memory. The method is called K-Means since each of the K clusters is represented by the mean of the objects (called the centroid) within it. It is also called the centroid method since at each step the centroid point of each cluster is assumed to be known and each of the remaining points are allocated to the cluster whose centroid is closest to it. Once this allocation is completed, the centroids of the clusters are recomputed using simple means and the process of allocating points to each cluster is repeated until there is no change in the clusters (or some other stopping criterion, e.g. no significant reduction in the squared error, is met). The method may also be liked as a search problem where the aim is essentially to find the optimum clusters given the number of clusters and seeds specified by the user. Obviously, we cannot use a brute-force or exhaustive search method to find the optimum, so we consider solutions that may not be optimal but may be computed efficiently.

The K-means method uses the Euclidean distance measure, which appears to work well with compact clusters. If instead of the Euclidean distance, the Manhattan distance is used the method is called the K-median method. The K-median method can be less sensitive to outliers.

The K-means method may be described as follows:

1. Select the number of clusters. Let this number be k.
2. Pick k seeds as controids of the k clusters. The seeds may be picked randomly unless

   the user has some insight into the data.
3. Compute the Euclidean distance of each object in the dataset from each of the

   centroids.
4. Allocate each object to the cluster it is nearest to based on the distances computed in

   the previous step.

   5. Compute the centroids of the clusters by computing the means of the attribute values of the objects in each cluster.

   6. Check if the stopping criterion has been met (e.g. the cluster membership is unchanged). If yes, go to Step 7. If not, go to Step 3.

7. [Optional] One may decide to stop at this stage or to split a cluster or combine two

   clusters heuristically until a stopping criterion is met.

The method is scalable and efficient (the time complexity is of O(n)) and is guaranteed to find a local minimum.

Scaling and weighting

For clustering to be effective, all attributes should be converted to a similar scale unless we want to give more weight to some attributes that are relatively large in scale. There are a number of ways to transform the attributes. One possibility is to transform them all to a normalized score or to a range (0,1). Such transformations are called scaling. Some other approaches to scaling are given below:

1. Divide each attribute by the mean value of that attribute. This reduces the mean of each attribute to 1. It does not control the variation; some values may still be large, others small.
2. Divide each attribute by the difference between the largest value and increases the mean of attributes that have a small range of values but does not reduce each attribute's mean to the same value. The scaling reduces the difference between the largest value and the smallest value to 1 and therefore does control the variation.
3. Convert the attribute values to "standardized scores" by subtracting the mean of the attribute from each attribute value and dividing it by the standard deviation. Now the mean of each attribute will be zero and standard deviation one. This not only scales the magnitude of each attribute to a similar range but also scales the standard deviation.

Starting values for the K-means method

Often the user has little basis for specifying the number of clusters and starting seeds. This problem may be overcome by using an iterative approach. For example, one may first three clusters and choose three starting seeds randomly. Once the final clusters have been obtained, the process may be repeated with a different set of seeds. Attempts should

be made to select seeds that are as far away from each other as possible. Also, during the iterative process if two clusters are found to be close together, it may be desirable to merge them. Also, a large cluster may be split to in two if the variance within the cluster is above some threshold value.

Another approach involves finding the centroid of the whole dataset and then perturbing this centroid value to find seeds. Yet another approach recommends using a hierarchical method like the agglomerative method on the data first, since that method does not require staring values, and then using the results of that method as the basis for specifying the number of clusters and starting seeds.

Summary of the K-means method

K-means is an iterative-improvement greedy method. A number of iterations are normally needed for convergence and therefore the dataset is processed a number of times. If the data is very large and cannot be accommodated in the main memory the process may become inefficient.

Although the K-means method is most widely known and used, there are a number of issues related to the method that should be understood:

1. The K-means method needs to compute Euclidean distances and means of the attribute values of objects within a cluster. The classical algorithm therefore is only suitable for continuous data. K-means variations that deal with categorical data those are available but not widely used.
2. The K-means method implicitly assumes spherical probability distributions.
3. The results of the K-means method depend strongly on the initial guesses of the seeds.
4. The K-means method can be sensitive to outliers. If an outlier is picked as a starting seed, it may end up in a cluster of its own. Also, if an outlier moves from one cluster to another during iterations, it can have a major impact on the clusters because the means of the two clusters are likely to change significantly.
5. Although some local optimum solutions discovered by the K-means method are satisfactory, often the local optimum is not as good as the global optimum.

6. The K-means method does not consider the size of the clusters. Some clusters may be large and some very small.
7. The K-means method does not deal with overlapping clusters.

Expectation Maximization Method

The K-means method does not explicitly assume any probability distribution for the attribute values. It only assumes that the dataset consists of groups of objects that are similar and the groups can be discovered because the user has provided cluster seeds.

In contrast to the K-means method, the Expectation Maximization (EM) method is based on the assumption that the objects in the dataset have attributes whose values are distributed according to some (unknown) linear combination (or mixture) of simple probability distributions. While the K-means method involves assigning objects to clusters to minimize within-group variation, the EM method assigns objects to different clusters with certain probabilities in an attempt to maximize expectation (or likelihood) of assignment.

The simplest situation is when there are only two distributions. For every individual we may assume that it comes from distribution 1 with probability p and therefore from distribution 2 with probability 1-p. Such mixture models appear to be widely used because they provide more parameters and therefore more flexibility in modeling.

The EM method consists of a two-step iterative algorithm. The first step, called the Estimation step or the E-step, involves estimating the probability distributions of the clusters given the data. The second step, called the Maximization step or the M-step, involves finding the model parameters that maximize the likelihood of the solution.

The EM method assumes that all attributes are independent random variables. In a simple case of just two clusters with objects having only a single attribute, we may assume that the attribute values vary according to a normal distribution. The EM method requires that we now estimate the following parameters:

1. The mean and standard deviation of the normal distribution for cluster 1

2. The mean and standard deviation of the normal distribution for cluster 2
3. The probability p of a sample belonging to cluster 1 and therefore probability 1-p of belonging to cluster 2

The EM method then works as follows:

1. Guess the initial values of the five parameters (the two means, the two standard deviations and the probability p) given above.
2. Use the two normal distributions (given the two guesses of means and two guesses of standard deviations) and compute the probability of each object belonging to each of the two clusters.
3. Compute the likelihood of data coming from these two clusters by multiplying the sum of probabilities of each object.
4. Re-estimate the five parameters and go to Step 2 until a stopping criterion has been met.

4.7 HIERARCHICAL METHODS

Hierarchical methods produce a nested series of clusters as opposed to the partitional methods which produce only a flat set of clusters. Essentially the hierarchical methods attempt to capture the structure of the data by constructing a tree of clusters. This approach allows clusters to be found at different levels of granularity.

There are two types of hierarchical approaches possible. In one approach, called the *agglomerative* approach for merging groups (or bottom-up approach), each object at the start is a cluster by itself and the nearby clusters are repeatedly merged resulting in larger and larger clusters until some stopping criterion (often a given number of clusters) is met or all the objects are merged into a single large cluster which is the highest level of the hierarchy. In the second approach, called the *divisive* approach (or the top-down approach), all the objects are put in a single cluster to start. The method then repeatedly performs splitting of clusters resulting in smaller and smaller clusters until a stopping criterion is reached or each cluster has only one object in it.

Distance Between Clusters

The hierarchical clustering methods require distances between clusters to be computed. These distance metrics are often called *linkage* metrics.

Computing distances between large clusters can be expensive. Suppose one cluster has 50 objects and another has 100, then computing most of the distance metrics listed below would require computing distances between each object in the first cluster with every object in the second. Therefore 5000 distances would need to be computed just to compute a distance between two clusters. This can be expensive if each object has many attributes.

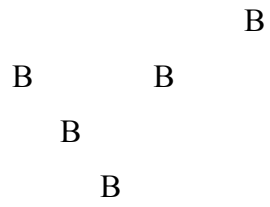
We will discuss the following methods for computing distances between clusters:

1. Single-link algorithm
2. Complete-link algorithm
3. Centroid algorithm
4. Average-link algorithm
5. Ward's minimum-variance algorithm

Single-link

The single-link (or the nearest neighbour) algorithm is perhaps the simplest algorithm for computing distance between two clusters. The algorithm determines the distance between two clusters as the minimum of the distances between all pairs of points (a,x) where a is from the firs cluster and x is from the second. The algorithm therefore requires that all pairwise distances be computed and the smallest distance (or the shortest kink) found. The algorithm can form chains and can form elongated clusters.

Figure 4.2 shows two clusters A and B the single-link distance between them.



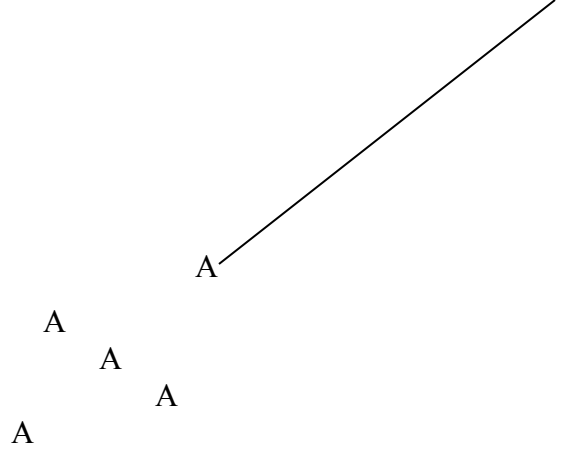


Figure 4.2 Single-link distance between two clusters.

Complete-link

The complete-link algorithm is also called the *farthest neighbour* algorithm. In this algorithm, the distance between two clusters is defined as the maximum of the pairwise distances (a,x). Therefore if there are *m* elements in one cluster and *n* in the other, all *mn* pairwise distances therefore must be computed and the largest chosen.

Compute link is strongly biased towards compact clusters. Figure 4.3 shows two clusters A and B and the complete-link distance between them. Complete-link can be distorted by moderate outliers in one or both of the clusters.

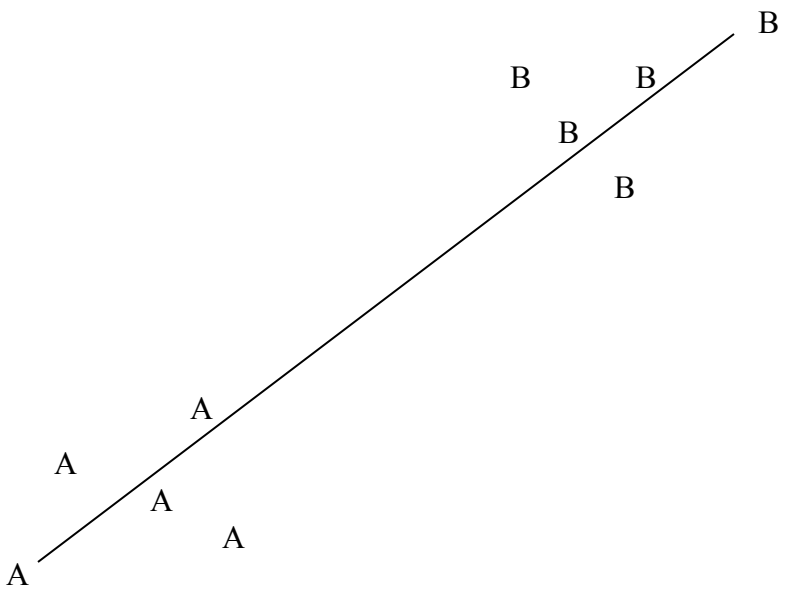


Figure 4.3 Complete-link distance between two clusters.

Both single-link and complete-link measures have their difficulties. In the single-link algorithm, each cluster may have an outlier and the two outliers may be nearby and so the distance between the two clusters would be computed to be small. Single-link can form a chain of objects as clusters are combined since there is no constraint on the distance between objects that are far away from each other.

Centroid

In the centroid algorithm the distance between two clusters is determined as the distance between the centroids of the clusters as shown below. The centroid algorithm computes the distance between two clusters as the distance between the average point of each of the two clusters. Usually the squared Euclidean distance between the centroids is used. This approach is easy and generally works well and is more tolerant of somewhat longer clusters than the complete-link algorithm. Figure 4.4 shows two clusters A and B and the centroid distance between them.

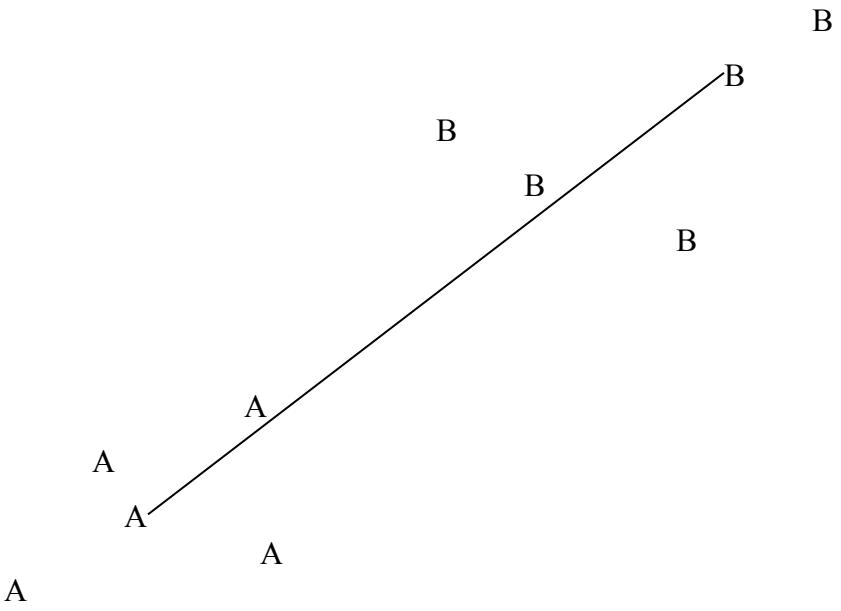


Figure 4.4 The Centroid distance between two clusters.

Average-link

The average-link algorithm on the other hand computes the distance between two clusters as the average of all pairwise distances between an object from one cluster and another from the other cluster. Therefore if there are m elements in one cluster and n in the other, there are mn distances to be computed, added and divided by mn. This approach also generally works well. It tends to join clusters with small variances although it is more tolerant of somewhat longer clusters than the complete-link algorithm. Figure 4.5 shows two clusters A and B and the average-link distance between them.

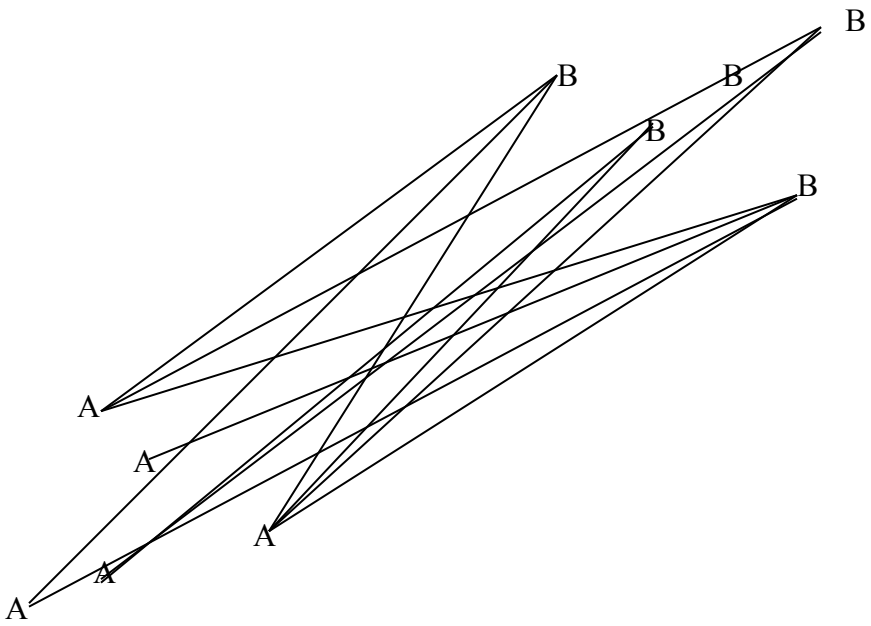


Figure 4.5 The Average-link distance between two clusters.

Ward's minimum-variance method

Ward's minimum-variance distance measure on the other hand is different. The method generally works well and results in creating small tight clusters. Ward's distance is the difference between the total within the cluster sum of squares for the two clusters separately and within the cluster sum of squares resulting from merging the two clusters. An example for ward's distance may be derived. It may be expressed as follows:

$$D_W(A,B) = N_A N_B D_C(A,B)/(N_A + N_B)$$

Where $\mathbf{D_W(A,B)}$ is the Ward's minimum-variance distance between clusters A and B with $\mathbf{N_A}$ and $\mathbf{N_B}$ objects in them respectively. $\mathbf{D_C(A,B)}$ is the centroid distance between the two clusters computed as squared Euclidean distance between the centroids. It has been observed that the Ward's method tends to join clusters with a small number of objects and is biased towards producing clusters with roughly the same number of objects. The distance measure can be sensitive to outliers.

Agglomerative Method

Some applications naturally have a hierarchical structure. For example, the world's fauna and flora have a hierarchical structure. The agglomerative clustering method tries to discover such structure given a dataset.

The basic idea of the agglomerative method is to start out with n clusters for n data points, that is, each cluster consisting of a single data point. Using a measure of distance, at each step of the method, the method merges two nearest clusters, thus reducing the number of clusters and building obtained or all the data points are in one cluster. The agglomerative method leads to hierarchical clusters in which at each step we build larger and larger clusters that include increasingly dissimilar objects.

The agglomerative method is basically a bottom-up approach which involves the following steps.

1. Allocate each point to a cluster of its own. Thus we start with n clusters for n objects.
2. Create a distance matrix by computing distances between all pairs of clusters either using, for example, the single-link metric or the complete-link metric. Some other metric may also be used. Sort these distances in ascending order.
3. Find the two clusters that have the smallest distance between them.
4. Remove the pair of objects and merge them.
5. If there is only one cluster left then stop.
6. Compute all distances from the new cluster and update the distance matrix after the merger and go to Step 3.

Divisive Hierarchical Method

The divisive method is the opposite of the agglomerative method in that the method starts with the whole dataset as one cluster and then proceeds to recursively divide the cluster into two sub-clusters and continues until each cluster has only one object or some other stopping criterion has been reached. There are two types of divisive methods:

1. ***Monothetic:*** It splits a cluster using only one attribute at a time. An attribute that has the most variation could be selected.
2. ***Polythetic:*** It splits a cluster using all of the attributes together. Two clusters far apart could be built based on distance between objects.

A typical polythetic divisive method works like the following:

1. Decide on a method of measuring the distance between two objects. Also decide a threshold distance.
2. Create a distance matrix by computing distances between all pairs of objects within the cluster. Sort these distances in ascending order.
3. Find the tow objects that have the largest distance between them. They are the most dissimilar objects.
4. If the distance between the two objects is smaller than the per-specified threshold and there is no other cluster that needs to be divided then stop, otherwise continue.
5. Use the pair of objects as seeds of a **K-**means method to create two new clusters.
6. If there is only one object in each cluster then stop otherwise continue with Step 2.

In the above method, we need to resolve the following two issues:

Which cluster to split next?

How to split a cluster?

Which cluster to split next?

There are a number of possibilities when selecting the next cluster to split:

1. Split the clusters in some sequential order.
2. Split the cluster that has the largest number of objects.
3. Split the cluster that has the largest variation within it.

How to split a cluster?

A distance matrix is created and the two most dissimilar objects are selected as seeds of two new clusters. The **K-**means method is then used to split the cluster.

Advantages of the hierarchical approach

1. The hierarchical approach can provide more insight into the data by showing a hierarchy of clusters than a flat cluster structure created by a partitioning method like the **K-**means method.
2. Hierarchical methods are comceptually simpler and can be implemented easily.
3. In some applications only proximity data is available and then the hierarchical approach may be better.
4. Hierarchical methods can provide clusters at different levels of granularity.

Disadvantages of the hierarchical approach

1. The hierarchical methods do not include a mechanism by which objects that have been incorrectly put in a cluster may be reassigned to another cluster.
2. The time complexity of hierarchical methods can be shown to be $\mathbf{O}(n^3)$.
3. The distance matrix requires $\mathbf{O}(n^3)$ space and becomes very large for a large number of objects.
4. Different distance metrics and scaling of data can significantly change the results.

4.8 DENSITY-BASED METHODS

The density-based methods are based on the assumption that clusters are high density collections of data of arbitrary shape that are separated by a large space of low density data (which is assumed to be noise).

DBSCAN (density based spatial clustering of applications with noise) is one example of a density-based method for clustering. The method was designed for spatial databases but can be used in other applications. It requires two input parameters: the size of the neighbourhood (R) and the minimum points in the neighbourhood (N). Essentially these two parameters determine the density within the clusters the user is willing to accept since they specify how many points must be in a region. The number of points not only determines the density of acceptable clusters but it also determines which objects will be labeled outliers or noise. Objects are declared to be outliers if there are few other objects in their neighbourhood. The size parameter R determines the size of the clusters found. If R is big enough, there would be one big cluster and no outliers. If R is small, there will be small dense clusters and there might be many outliers.

We now define a number of concepts that are required in the DBSCAN method:

1. ***Neighbourhood****:* The neighbourhood of an object y is defined as all the objects that are within the radius R from y.
2. ***Core object*****:** An object y is called a core object if there are N objects within its neighbourhood.
3. ***Proximity:*** Two objects are defined to be in proximity to each other if they belong to the same cluster. Object x_1 is in proximity to object x_2 if two conditions are satisfied:
    (a) The objects are close enough to each other, i.e. within a distance of R.
    (b) x_2 is a core object as defined above.
4. ***Connectivity:*** Two objects x_1 and x_n are connected if there is a path or chain of objects $x_1, x_2, \ldots\ldots\ldots\ldots, x_n$ from x_1 to x_n such that each x_{i+1} is in proximity to object x_i.

We now outline the basic algorithm for density based clustering:

1. Select values of R and N.

2. Arbitrarily select an object p.
3. Retrieve all objects that are connected to p, given R and N.
4. If p is a core object, a cluster is formed.
5. If p is border object, no objects are in its proximity. Choose another object. Go to Step 3.
6. Continue the process until all of the objects have been processed.

4.9 DEALING WITH LARGE DATABASES

Most clustering methods implicitly assume that all data is accessible in the main memory. Often the size of the database is not considered but a method requiring multiple scans of data that is disk-resident could be quite inefficient for large problems.

One possible approach to deal with large datasets that could be used with any type of clustering method is to draw as large a sample from the large dataset as could be accommodated in the main memory. The sample is then clustered. Each remaining object is then assigned to the nearest cluster obtained from the sample. This process could be repeated several times and the clusters that lead to the smallest within clusters variance could be chosen.

K-Means Method for Large Databases

This method first picks the number of clusters and their seed centroids and then attempts to classify each object to belong to one of the following three groups:

(a) Those that are certain to belong to a cluster. These objects together are called the discard set. Some information about these objects is computed and saved. This includes the number of objects n, a vector sum of all attribute values of the n objects (a vector S) and a vector sum of squares of all attribute values of the n objects (a vector Q). These values are sufficient to recompute the centroid of the new cluster and its variance.

(b) Those that are sufficiently close to each other to be replaced by their summary. The objects are however sufficiently far away from each cluster's centroid that

they cannot yet be put in the discard set of objects. These objects together are called the compression set.

(c) The remaining objects are too difficult to assign to either of the two groups above. These objects are called the retained set and are stored as individual objects. They cannot be replaced by a summary.

Hierarchical Method for Large Databases – Concept of Fractionation

Dealing with large datasets is difficult using hierarchical methods since the methods require an $N \times N$ distance matrix to be computed for N objects. If N is large, say 100,000, the matrix has 10^{10} elements making it impractical to use a classical hierarchical method.

A modification of classical hierarchical methods that deals with large datasets was proposed in 1992. It is based on the idea of splitting the data into manageable subsets called "fractions" and then applying a hierarchical method to each fraction. The concept is called fractionation. The basic algorithm used in the method is as follows, assuming that M is the largest number of objects that the hierarchical method may be applied to. The size M may be determined perhaps based on the size of the main memory.

Now the algorithm:

1. Split the large dataset into fractions of size M.
2. The hierarchical clustering technique being used is applied to each fraction. Let the number of clusters so obtained from all the fractions be C.
3. For each of the C clusters, compute the mean of the attribute values of the objects in it. Let this mean vector be m_i, i= 1,…..,C. These cluster means are called meta-observations. The meta-observations now become the data values that represent the fractions.
4. If the C meta-observations are too large (greater than M), go to step 1, otherwise apply the same hierarchical clustering technique to the meta-observations obtained in step 3.
5. Allocate each object of the original dataset to the cluster with the nearest mean obtained in step 4.

4.10 QUALITY OF CLUSTER ANALYSIS METHODS

The quality of the clustering methods or results of a cluster analysis is a challenging task. The quality of a method involves a number of criteria:

1. Efficiency of the method.
2. Ability of the method to deal with noisy and missing data.
3. Ability of the method to deal with large problems.
4. Ability of the method to deal with a variety of attribute types and magnitudes.

4.11 CLUSTER ANALYSIS SOFTWARE

A more comprehensive list of cluster analysis software is available at http://www.kdnuggets.com/software/clustering.html.

- ClustanGraphics7 from Clustan offers a variety of clustering methods including K-means, density-based and hierarchical cluster analysis. The software provides facilities to display results of clustering including dendograms and scatterplots. For more details visit: http://www.clustan.com/index.html
- CViz Cluster Visualization from IBM is a visualization tool designed for analyzing high-dimensional data in large, complex data sets. It displays the most important factors relating clusters of records. For more details visit: http://www.alphaworks.ibm.com/tech/cviz

- AutoClass, an unsupervised Bayesian classification system, based on SNOB mentioned below, is available from NASA. (Free). For more details visit: http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/

- Cluster 3.0, open source software originally developed by Michael Eisen at Stanford. It uses the K-means method, which includes multiple trials to find the best clustering solution. This is crucial for the K-means method to be reliable. For

more details visit: http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm

- CLUTO provides a set of clustering methods including partitional, agglomerative, and graph-partitioning based on a variety of similarity/distance metrics. For more details visit: http://www-users.cs.umn.edu/~karypis/cluto/ (Free)

CONCLUSION

Cluster analysis is a collection of methods that assists the user in putting different objects from a collection of objects into different groups. In some ways one could say that cluster analysis is best used as an exploratory data analysis exercise when the user has no hypothesis to test. Cluster analysis, therefore, can be used to uncover hidden structure which may assist further exploration.

We have discussed a number of clustering methods. In the K-means method it was required that the user specifies the number of clusters and starting seeds for each cluster. This may be difficult to do without some insight into the data that the user may not have. One possible approach is that might combine a partitioning method like K-means with a hierarchical method like the agglomerative method. The agglomerative method can then be used to understand better the data and help in estimating the number of clusters and the starting seeds. The strength of cluster analysis is that it works well with numeric data. Techniques that work well are available with categorical and textual data as well. Cluster analysis is easy to use.

We have noted that the performance of cluster analysis methods can be dependent on the choice of the distance metric. It can be difficult to devise a suitable distance metric for data that contains a mixture of variable types. It can also be difficult to determine a proper weighting scheme for disparate variable type. Furthermore, since cluster analysis is exploratory, the results of cluster analysis sometimes can be difficult to interpret. On the other hand, quite unexpected results may be obtained. For example, at NASA two subgroups of stars were distinguished, where previously no difference was suspected.

REVIEW QUESTIONS

1. List four desirable features of a cluster analysis method. Which of them are important for large databases? Discuss.

2. Discuss the different types of data which one might encounter in practice. What a data type is clustering most suitable for?

3.Given two objects represented by the attribute values (1, 6, 2, 5, 3) and (3, 5, 2, 6, 6)

a) Compute the Euclidean distance between the two objects

b) Compute the Manhattan distance between the two objects.

4. Suppose that a data mining task is to cluster the following eight points (with (x, y) representing location) into three clusters.

$A_1(4,6)$, $A_2(2, 5)$, $A_3(9,3)$, $A_4(6,9)$, $A_5(7,5)$, $A_6(5,7)$, $A_7(2,2)$, $A_8(6,6)$

Suppose initially we assign A_1, A_2 and A_3 as the seeds of three clusters that we wish to find. Use the K-means method to show:

a) The three cluster centroids after the first iteration using the Manhattan distance

b) The final three clusters

c) Repeat the exercise using the Euclidean distance.

5. What kind of shapes would the different distance metrics deliver? Devise a new distance measure that could lead to sausage type clusters.

6. The following six objects, each with two attributes, are to be clustered:

$A_1(4,6)$, $A_2(2, 5)$, $A_3(9,3)$, $A_4(6,9)$, $A_5(7,5)$, $A_6(5,7)$

a) Show the distance matrix for the six objects using the Manhattan distance

b) Using the devisive method, determine the two objects that should form the basis for splitting the above dataset

c) Now split the dataset using the two objects identified in part (b) using K-means method.