DATA ANALYTICS AND R PROGRAMMING

Subject code : 18MIT22C

UNIT-II: Classification and Clustering: Classification and Prediction - Basic Concepts- Decision Tree Induction - Bayesian Classification – Rule Based Classification – Classification by Back Propagation Cluster Analysis - Types of Data – Categorization of Major Clustering Methods–K-means-Partitioning Methods – Hierarchical Methods – Clustering High Dimensional Data- Constraint Based Cluster Analysis – Outlier Analysis – Data Mining Applications.

Text book: 1. K.P. Soman, Shyam Diwakar and V. Ajay, “Insight into Data mining Theory and Practice”, Easter Economy Edition, Prentice Hall of India, 2006.

Prepared by: Dr.M.Soranamageswari

2. Classification and Clustering

- A bank loans officer needs analysis of her data in order to learn which loan applicants are “safe” and which are “risky” for the bank.
- A marketing manager atAllElectronicsneeds data analysis to help guess whether a customer with a given profile will buy a new computer. A medical researcher wants to analyze breast cancer data in order to predict which one of three specific treatments a patient should receive.
- In each of these examples, the data analysis task is **classification,** where a model or classifier is constructed to predict categorical labels, such as “safe” or “risky” for the loan application data; “yes” or “no” for the marketing data; or “treatment A,” “treatment B,” or “treatment C” for the medical data.
- These categories can be represented by discrete values, where the ordering among values has no meaning

Cont…

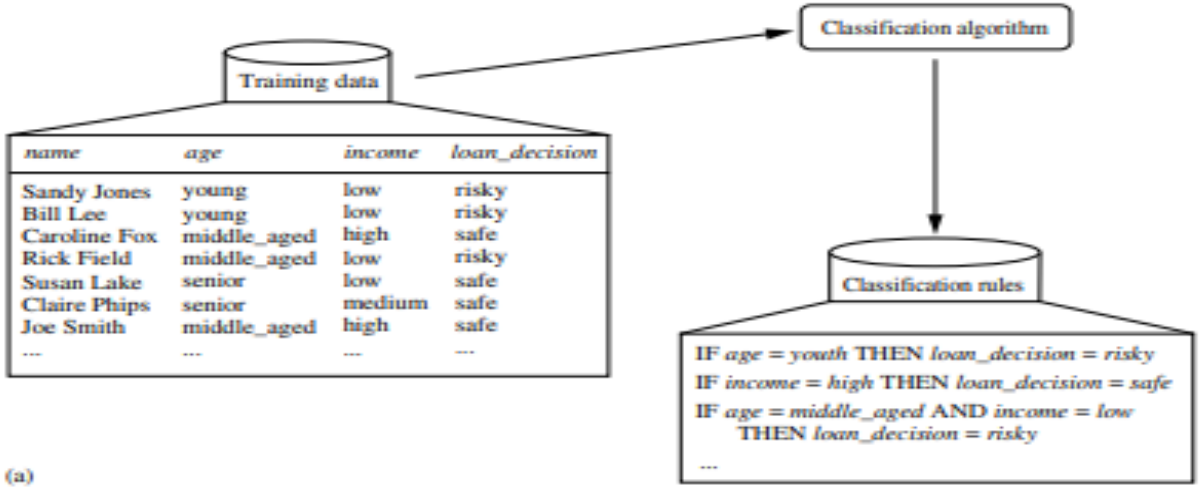
- This data analysis task is an example of numeric prediction, where the model constructed predicts a continuous-valued function, or ordered value, as opposed to a categorical label. This model is a predictor.
- Classification and numeric prediction are the two major types of prediction problems. For simplicity, when there is no ambiguity, we will use the shortened term of prediction to refer to numeric prediction.
- How does classification work? Data classification is a two-step process, as shown for the loan application data of Figure 6.1.

- In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or “learning from” a training set made up of database tuples and their associated class labels.
- A tuple, X, is represented by an n-dimensional attribute vector, X = (x1, x2,..., xn), depicting n measurements made on the tuple from n database attributes, respectively, A1, A2,..., An.

1 Each tuple, X, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. The class label attribute is discrete-valued and unordered. It is categorical in that each value serves as a category or class.

The individual tuples making up the training set are referred to as training tuples and are selected from the database under analysis. In the context of classification, data tuples can be referred to as samples, examples, instances, data points, or objects.

2 Because the class label of each training tuple is provided, this step is also known as supervised learning to which class each training tuple belongs). It contrasts with unsupervised learning (or clustering), in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance.



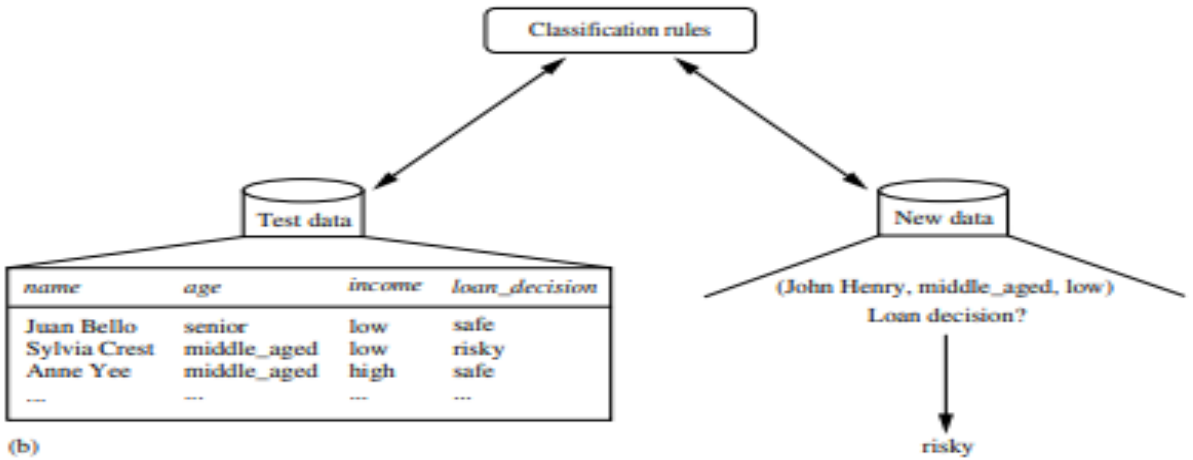


Figure 6.1 The data classification process: (a) *Learning*: Training data are analyzed by a classification algorithm. Here, the class label attribute is *loan_decision*, and the learned model or classifier is represented in the form of classification rules. (b) *Classification*: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

- This first step of the classification process can also be viewed as the learning of a mapping or function, y = f(X), that can predict the associated class label y of a given tuple X.
- In this view, we wish to learn a mapping or function that separates the data classes. Typically, this mapping is represented in the form of classification rules, decision trees, or mathematical formulae. In our example, the mapping is represented as classification rules that identify loan applications as being either safe or risky (Figure 6.1(a)).
- "What about classification accuracy?" In the second step (Figure 6.1(b)), the model is used for classification. First, the predictive accuracy of the classifier is estimated.
- If we were to use the training set to measure the accuracy of the classifier, this estimate would likely be optimistic, because the classifier tends to overfit the data (i.e., during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall).

- Therefore, a test set is used, made up of test tuples and their associated class labels. These tuples are randomly selected from the general data set.
- They are independent of the training tuples, meaning that they are not used to construct the classifier.
- The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.
- “How is (numeric) prediction different from classification?”

Prediction and classification also differ in the methods that are used to build their respective models.

2.1 Issues Regarding Classification and Prediction

Preparing the Data for Classification and Prediction

The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process.

Data cleaning: This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques, for example) and the treatment of missing values

Relevance analysis: Many of the attributes in the data may be redundant. Correlation analysis can be used to identify whether any two given attributes are statistically related. Attribute subset selection can be used in these cases to find a reduced set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.

https://youtu.be/gkagE_fE2sk

Data transformation and reduction: The data may be transformed by normalization, particularly when neural networks or methods involving distance measurements are used in the learning step. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as −1.0 to 1.0, or 0.0 to 1.0. In methods that use distance measurements.

Comparing Classification and Prediction Methods

Accuracy: The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data (i.e., tuples without class label information).

Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data.

Speed: This refers to the computational costs involved in generating and using the given classifier or predictor.

Robustness: This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values. Scalability: This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

Interpretability: This refers to the level of understanding and insight that is provided by the classifier or predictor. Interpretability is subjective and therefore more difficult to assess

2.2 Classification by Decision Tree Induction

- Decision tree induction is the learning of decision trees from class-labeled training tuples.
- A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node.
- A typical decision tree is shown in Figure It represents the concept buys computer, that is, it predicts whether a customer at AllElectronics is likely to purchase a computer. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals. Some decision tree algorithms produce only binary trees (where each internal node branches to exactly two other nodes), whereas others can produce nonbinary trees.

- “How are decision trees used for classification?” Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.
- “Why are decision tree classifiers so popular?” The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data.
- Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy. However, successful use may depend on the data at hand. Decision tree induction algorithms have been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems.

2.2.1 Decision Tree Induction

- During the late 1970s and early 1980s, J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).
- This work expanded on earlier work on concept learning systems, described by E. B. Hunt, J. Marin, and P. T. Stone. Quinlan later presented C4.5 (a successor of ID3), which became a benchmark to which newer supervised learning algorithms are often compared.
- In 1984, a group of statisticians (L. Breiman, J. Friedman, R. Olshen, and C. Stone) published the book Classification and Regression Trees (CART), which described the generation of binary decision trees. ID3 and CART were invented independently of one another at around the same time, yet follow a similar approach for learning decision trees from training tuples. These two cornerstone algorithms spawned a flurry of work on decision tree induction

- ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.
- Most algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built. A basic decision tree algorithm is summarized in Figure 6.3.
- Figure 6.3 Basic algorithm for inducing a decision tree from training tuples

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data partition D.

Input:

Data partition, D, which is a set of training tuples and their associated class labels;

attribute list, the set of candidate attributes;

Attribute selection method, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split point or splitting subset.

Output:

A decision tree. Method:

(1) create a node N;

(2) if tuples in D are all of the same class, C then

(3) return N as a leaf node labeled with the class C;

(4) if attribute list is empty then

(5) return N as a leaf node labeled with the majority class in D; // majority voting

(6) apply Attribute selection method(D, attribute list) to find the “best” splitting criterion;

(7) label node N with splitting criterion;

(8) if splitting attribute is discrete-valued and

multiway splits allowed then // not restricted to binary trees

(9) attribute list ← attribute list − splitting attribute; // remove splitting attribute

(10) for each outcome j of splitting criterion // partition the tuples and grow subtrees for each partition

(11) let Dj be the set of data tuples in D satisfying outcome j;

// a partition

(12) if Dj is empty then

(13) attach a leaf labeled with the majority class in D to node N;

(14) else attach the node returned by Generate decision tree(Dj , attribute list) to node N; endfor

(15) return N;

2.2.2 Attribute Selection Measures

- An attribute selection measure is a heuristic for selecting the splitting criterion that “best” separates a given data partition, D, of class-labeled training tuples into individual classes.
- Attribute selection measures are also known as splitting rules because they determine how the tuples at a given node are to be split. The attribute selection measure provides a ranking for each attribute describing the given training tuples. The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples.
- This section describes three popular attribute selection measures—information gain, gain ratio, and gini index.

  Information gain: highest information gain

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

where pi is the probability that an arbitrary tuple in D belongs to classCi and is estimated by |Ci,D|/|D|.

A log function to the base 2 is used, because the information is encoded in bits.

Info(D) is just the average amount of information needed to identify the class label of a tuple in D. Note that, at this point, the information we have is based solely on the proportions of tuples of each class. Info(D) is also known as the entropy of D.

Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,

$$Gain(A) = Info(D) - Info_A(D).$$

Gain ratio

The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values. It applies a kind of normalization to information gain using a "split information" value defined analogously with Info(D) as

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2 \frac{|D_j|}{|D|}$$

It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning. The gain ratio is defined as

$$\text{GainRatio}(A) = \text{Gain}(A) \; \text{SplitInfo}(A) .$$

Gini index

The Gini index is used in CART. Using the notation described above, the Gini index measures the impurity of D, a data partition or set of training tuples, as

$$\text{Gini}(D) = 1 - \sum_{i=1}^{m} p_i^2 ,$$

where pi is the probability that a tuple in D belongs to class Ci and is estimated by |Ci,D|/|D|. The sum is computed over m classes. The Gini index considers a binary split for each attribute.

Tree Pruning

- Tree pruning methods address this problem of overfitting the data. Such methods typically use statistical measures to remove the least reliable branches.
- “How does tree pruning work?” There are two common approaches to tree pruning: **prepruning and postpruning**
- In the prepruning approach, a tree is “pruned” by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node). Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.
- The second and more common approach is postpruning, which removes subtrees from a “fully grown” tree. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced. For example, notice the subtree at node “A3?” in the unpruned tree of Figure.

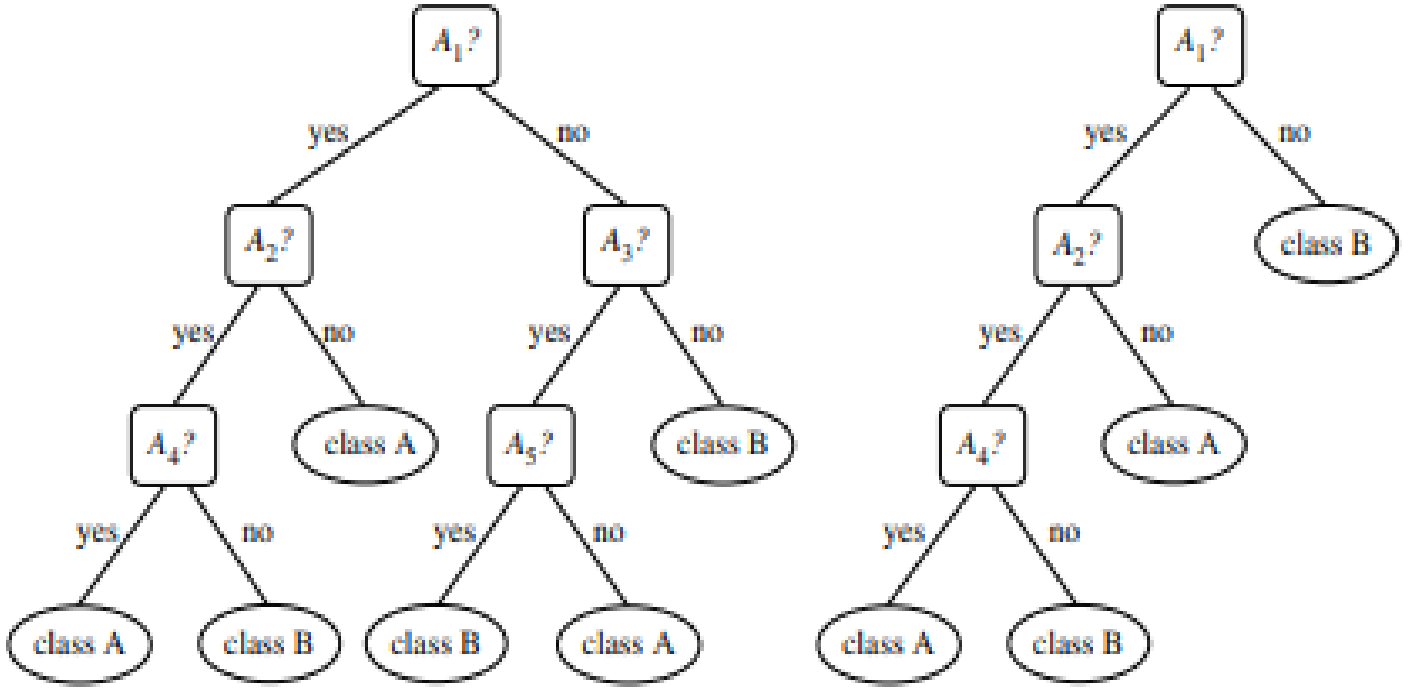


Figure 6.6 An unpruned decision tree and a pruned version of it.

Scalability and Decision Tree Induction

- “What if D, the disk-resident training set of class-labeled tuples, does not fit in memory?
- In other words, how scalable is decision tree induction?” The efficiency of existing decision tree algorithms, such as ID3, C4.5, and CART, has been well established for relatively small data sets

2.3 Bayesian Classification

- “What are Bayesian classifiers?” Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes’ theorem, described below. Studies comparing classification algorithms have found a simple Bayesian classifier known as the naive Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.
- Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered “naïve.”
- Bayesian belief networks are graphical models, which unlike naïve Bayesian classifiers, allow the representation of dependencies among subsets of attributes. Bayesian belief networks can also be used for classification.
- https://youtu.be/hPQiUk66bLM

2.3.1 Bayes' Theorem

- Bayes' theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let X be a data tuple.
- In Bayesian terms, X is considered "evidence." As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis, such as that the data tuple X belongs to a specified class C. For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the "evidence" or observed data tuple X.
- P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X.
- Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X). Bayes' theorem is

$$P(H|X) = P(X|H)P(H) / P(X)$$

2.3.2 Naïve Bayesian Classification

- The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2,..., xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2,..., An.

2. Suppose that there are m classes, C1, C2,..., Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(Ci|X) > P(Cj|X) \text{ for } 1 \leq j \leq m, j \mathrel{!}= i.$$

Thus we maximize P(Ci |X). The classCi for which P(Ci |X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

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

3. As P(X) is constant for all classes, only P(X|Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = ⋯ = P(Cm), and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci). Note that the class prior probabilities may be estimated by P(Ci) = |Ci,D|/|D|, where |Ci,D| is the number of training tuples of class Ci in D.

4. Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci). In order to reduce computation in evaluating P(X|Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$
$$= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i).$$

5. In order to predict the class label of X, P(X|Ci)P(Ci) is evaluated for each class Ci . The classifier predicts that the class label of tuple X is the class Ci if and only if

$$P(X|Ci)P(Ci) > P(X|Cj)P(Cj) \text{ for } 1 \leq j \leq m, j \mathrel{!}= i$$

2.4 Rule-Based Classification

- Rules are a good way of representing information or bits of knowledge. A rule-based classifier uses a set of IF-THEN rules for classification.

An IF-THEN rule is an expression of the form

IF condition THEN conclusion

An example is rule R1,

R1: IF age = youth AND student = yes THEN buys computer = yes.

https://youtu.be/3VMYspfRKZY

The “IF”-part (or left-hand side) of a rule is known as the rule antecedent or precondition. The “THEN”-part (or right-hand side) is the rule consequent.

In the rule antecedent, the condition consists of one or more attribute tests (such as age = youth, and student = yes) that are logically ANDed.

The rule’s consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). R1 can also be written as

$$R1: (age = youth) \wedge (student = yes) \Rightarrow (buys\ computer = yes).$$

If the condition (that is, all of the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent issatisfied (or simply, that the rule is satisfied) and that the rule covers the tuple.

We can define the coverage and accuracy of R as

coverage(R) = ncovers/ |D|

accuracy(R) = ncorrect/ ncovers

That is, a rule's coverage is the percentage of tuples that are covered by the rule

2.4.1Rule Extraction from a Decision Tree

- Decision tree classifiers are a popular method of classification—it is easy to understand how decision trees work and they are known for their accuracy.
- Decision trees can become large and difficult to interpret. In this subsection, we look at how to build a rulebased classifier by extracting IF-THEN rules from a decision tree.
- In comparison with a decision tree, the IF-THEN rules may be easier for humans to understand, particularly if the decision tree is very large. To extract rules from a decision tree, one rule is created for each path from the root to a leaf node.
- Each splitting criterion along a given path is logically ANDed to form the rule antecedent (“IF” part). The leaf node holds the class prediction, forming the rule consequent (“THEN” part).

Rule Induction Using a Sequential Covering Algorithm

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.

Input:

D, a data set class-labeled tuples;

Att vals, the set of all attributes and their possible values. Output: A set of IF-THEN rules.

Method:

```
(1)  Rule set = {}; // initial set of rules learned is empty
(2)  for each class c do
(3)    repeat
(4)      Rule = Learn One Rule(D, Att vals, c);
(5)      remove tuples covered by Rule from D;
(6)    until terminating condition;
(7)    Rule set = Rule set +Rule; // add new rule to rule set
(8)  endfor
(9)  return Rule Set;
```

Figure 6.12 Basic sequential covering algorithm.

2.5 Classification by Backpropagation

“What is backpropagation?”

- Backpropagation is a neural network learning algorithm. The field of neural networks was originally kindled by psychologists and neurobiologists who sought to develop and test computational analogues of neurons. Roughly speaking, a neural network is a set of connected input/output units in which each connection has a weight associated with it.
- During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples.
- Neural network learning is also referred to as connectionist learning due to the connections between units.
- There are many different kinds of neural networks and neural network algorithms. The most popular neural network algorithm is backpropagation, which gained repute in the 1980s.

A Multilayer Feed-Forward Neural Network

- The backpropagation algorithm performs learning on a multilayer feed-forward neural network. It iteratively learns a set of weights for prediction of the class label of tuples.
- A multilayer feed-forward neural network consists of an input layer, one or more hidden layers, and an output layer. An example of a multilayer feed-forward network is shown in Figure.
- Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple.
- The inputs are fed simultaneously into the units making up the input layer. These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of “neuronlike” units, known as a hidden layer.

- The outputs of the hidden layer units can be input to another hidden layer, and so on. The number of hidden layers is arbitrary, although in practice, usually only one is used.
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network's prediction for given tuples.
- The units in the input layer are called input units. The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.
- The multilayer neural network shown in Figure 6.15 has two layers of output units. Therefore, we say that it is a two-layer neural network. Similarly, a network containing two hidden layers is called a three-layer neural network, and so on.
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer. It is fully connected in that each unit provides input to each unit in the next forward layer.

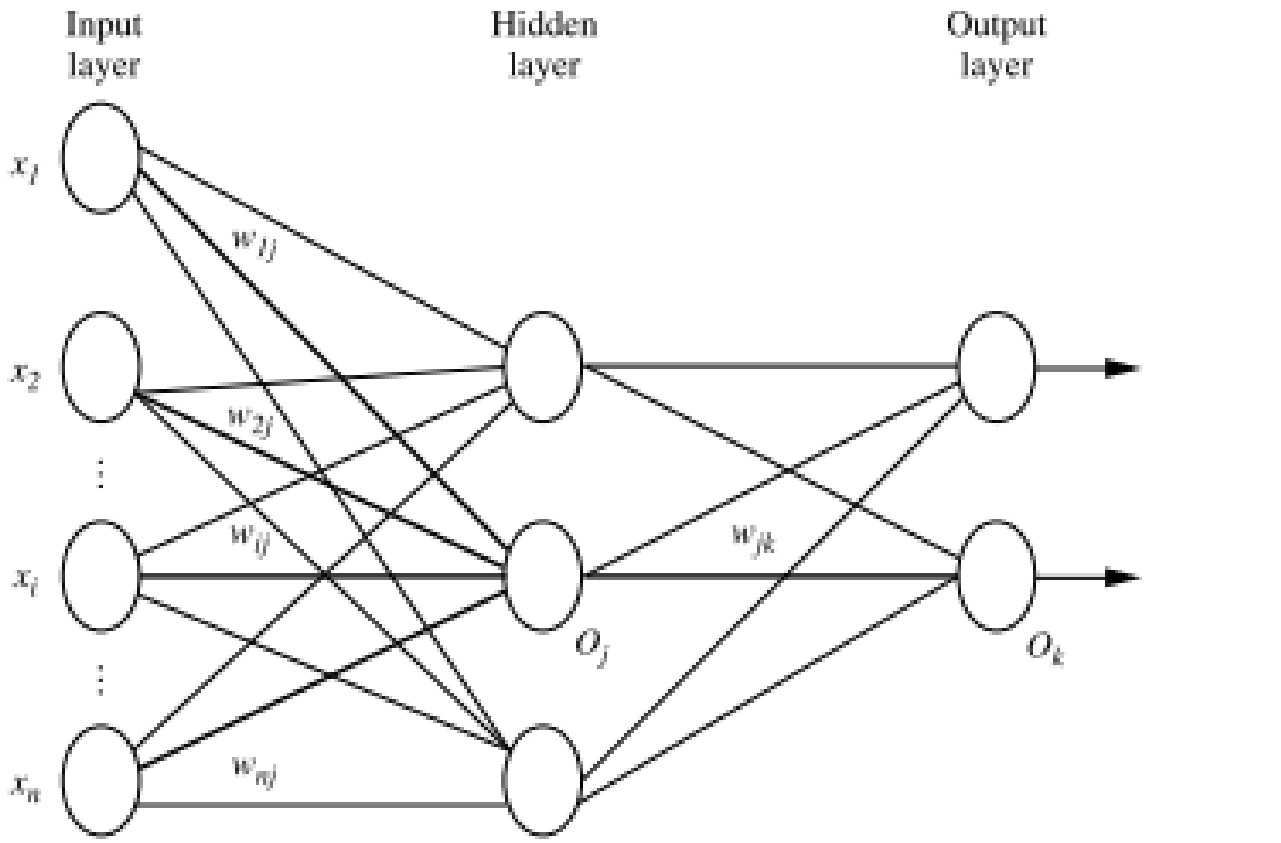


Figure 6.15 A multilayer feed-forward neural network.

Defining a Network Topology

"How can I design the topology of the neural network?" Before training can begin, the user must decide on the network topology by specifying the number of units in the input layer, the number of hidden layers (if more than one), the number of units in each hidden layer, and the number of units in the output layer.

Backpropagation

"How does backpropagation work?" Backpropagation learns by iteratively processing a data set of training tuples, comparing the network's prediction for each tuple with the actual known target value.

The target value may be the known class label of the training tuple (for classification problems) or a continuous value (for prediction). For each training tuple, the weights are modified so as to minimize the mean squared error between the network's prediction and the actual target value.

These modifications are made in the "backwards" direction, that is, from the output layer, through each hidden layer down to the first hidden layer (hence the name backpropagation).

Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.

Input:

D, a data set consisting of the training tuples and their associated target values;

l, the learning rate; network, a multilayer feed-forward network.

Output: A trained neural network.

Method:

(1) Initialize all weights and biases in network;

(2) while terminating condition is not satisfied {

(3) for each training tuple X in D {

(4) // Propagate the inputs forward:

(5) for each input layer unit j {

(6) Oj = Ij ; // output of an input unit is its actual input value

(7) for each hidden or output layer unit j {

(8) Ij = ∑iwi jOi +θj ; //compute the net input of unit j with respect to the previous layer, I

(9) Oj = 1 1+e −I j ; } // compute the output of each unit j

(10) // Backpropagate the errors:

(11) for each unit j in the output layer

(12) Errj = Oj(1−Oj)(Tj −Oj); // compute the error

(13) for each unit j in the hidden layers, from the last to the first hidden layer

(14) Errj = Oj(1−Oj)∑k Errkwjk; // compute the error with respect to the next higher layer, k

(15) for each weight wi j in network {

(16) Δwi j = (l)ErrjOi ; // weight increment

(17) wi j = wi j +Δwi j; } // weight update

(18) for each bias θj in network {

(19) Δθj = (l)Errj ; // bias increment

(20) θj = θj +Δθj ; } // bias update

(21) } }

2.6 What Is Cluster Analysis?

- The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.
- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.
- A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression.
- Although classification is an effective means for distinguishing groups or classes of objects, it requires the often costly collection and labeling of a large set of training tuples or patterns, which the classifier uses to model each group.
- It is often more desirable to proceed in the reverse direction: First partition the set of data into groups based on data similarity (e.g., using clustering), and then assign labels to the relatively small number of groups.
- Additional advantages of such a clustering-based process are that it is adaptable to changes and helps single out useful features that distinguish different groups.

2.7 Types of Data in Cluster Analysis

1. Data matrix (or object-by-variable structure):

This represents n objects, such as persons, with p variables (also called measurements or attributes), such as age, height, weight, gender, and so on. The structure is in the form of a relational table, or n-by-p matrix (n objects ×p variables):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \qquad (7.1)$$

2. Dissimilarity matrix (or object-by-object structure):

This stores a collection of proximities that are available for all pairs of n objects. It is often represented by an n-by-n table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \qquad (7.2)$$

- The rows and columns of the data matrix represent different entities, while those of the dissimilarity matrix represent the same entity. Thus, the data matrix is often called a two-mode matrix, whereas the dissimilarity matrix is called a one-mode matrix.
- Many clustering algorithms operate on a dissimilarity matrix. If the data are presented in the form of a data matrix, it can first be transformed into a dissimilarity matrix before applying such clustering algorithms.

2.8 A Categorization of Major Clustering Methods

- In general, the major clustering methods can be classified into the following categories.

Partitioning methods:

Given a database of n objects or data tuples, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$.

That is, it classifies the data into k groups, which together satisfy the following requirements: (1) each group must contain at least one object, and (2) each object must belong to exactly one group.

Hierarchical methods:

A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed.

The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group.

Density-based methods:

Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.

Other clustering methods have been developed based on the notion of density. Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the “neighborhood” exceeds some threshold;

Grid-based methods:

Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure (i.e., on the quantized space).

The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

Model-based methods:

Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model.

A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points.

It also leads to a way of automatically determining the number of clusters based on standard statistics, taking “noise” or outliers into account and thus yielding robust clustering methods.

- Aside from the above categories of clustering methods, there are two classes of clustering tasks that require special attention. One is **clustering high-dimensional data, and the other is constraint-based clustering**

Clustering high-dimensional data is a particularly important task in cluster analysis because many applications require the analysis of objects containing a large number of features or dimensions.

For example, text documents may contain thousands of terms or keywords as features, and DNA microarray data may provide information on the expression levels of thousands of genes under hundreds of conditions. Clustering high-dimensional data is challenging due to the curse of dimensionality. Many dimensions may not be relevant.

As the number of dimensions increases, the data become increasingly sparse so that the distance measurement between pairs of points become meaningless and the average density of points anywhere in the data is likely to be low.

Therefore, a different clustering methodology needs to be developed for high-dimensional data. CLIQUE and PROCLUS are two influential subspace clustering methods, which search for clusters in subspaces (or subsets of dimensions) of the data, rather than over the entire data space.

Frequent pattern–based clustering, another clustering methodology, extracts distinct frequent patterns among subsets of dimensions that occur frequently.

It uses such patterns to group objects and generate meaningful clusters. pCluster is an example of frequent pattern–based clustering that groups objects based on their pattern similarity.

Constraint-based clustering is a clustering approach that performs clustering by incorporation of user-specified or application-oriented constraints.

A constraint expresses a user's expectation or describes "properties" of the desired clustering results, and provides an effective means for communicating with the clustering process.

Various kinds of constraints can be specified, either by a user or as per application requirements. Our focus of discussion will be on spatial clustering with the existence of obstacles and clustering under user-specified constraints. In addition, semi-supervised clustering is described.

4.9 Outlier Analysis

- “What is an outlier?” Very often, there exist data objects that do not comply with the general behavior or model of the data.
- Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called outliers.
- Outliers can be caused by measurement or execution error.
- Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information because one person’s noise could be another person’s signal.
- In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity. Thus, outlier detection and analysis is an interesting data mining task, referred to as outlier mining.

4.10 Data Mining Applications

Data Mining for Financial Data Analysis

- Most banks and financial institutions offer a wide variety of banking services (such as checking and savings accounts for business or individual customers), credit (such as business, mortgage, and automobile loans), and investment services (such as mutual funds).
- Some also offer insurance services and stock investment services. Financial data collected in the banking and financial industry are often relatively complete, reliable, and of high quality, which facilitates systematic data analysis and data mining.

Here we present a few typical cases:

Design and construction of data warehouses for multidimensional data analysis and data mining:

Like many other applications, data warehouses need to be constructed for banking and financial data. Multidimensional data analysis methods should be used to analyze the general properties of such data.

Loan payment prediction and customer credit policy analysis:

Loan payment prediction and customer credit analysis are critical to the business of a bank. Many factors can strongly or weakly influence loan payment performance and customer credit rating.

Classification and clustering of customers for targeted marketing:

Classification and clustering methods can be used for customer group identification and targeted marketing.

Detection of money laundering and other financial crimes:

To detect money laundering and other financial crimes, it is important to integrate information from multiple databases (like bank transaction databases, and federal or state crime history databases), as long as they are potentially related to the study.

Data Mining for the Retail Industry

- The retail industry is a major application area for data mining, since it collects huge amounts of data on sales, customer shopping history, goods transportation, consumption, and service.
- A few examples of data mining in the retail industry are outlined as follows.

1. Design and construction of data warehouses based on the benefits of data mining
2. Multidimensional analysis of sales, customers, products, time, and region
3. Analysis of the effectiveness of sales campaigns
4. Customer retention—analysis of customer loyalty
5. Product recommendation and cross-referencing of items

Data Mining for the Telecommunication Industry

- The telecommunication industry has quickly evolved from offering local and long distance telephone services to providing many other comprehensive communication services, including fax, pager, cellular phone, Internet messenger, images, e-mail, computer and Web data transmission, and other data traffic.
- The following are a few scenarios for which data mining may improve telecommunication services:

1. Multidimensional analysis of telecommunication data
2. Fraudulent pattern analysis and the identification of unusual patterns
3. Multidimensional association and sequential pattern analysis
4. Mobile telecommunication services
5. Use of visualization tools in telecommunication data analysis

Data Mining for Biological Data Analysis

- Biological data mining has become an essential part of a new research field called bioinformatics.
- Since the field of biological data mining is broad, rich, and dynamic, it is impossible to cover such an important and flourishing theme in one subsection.
- Here we outline only a few interesting topics in this field, with an emphasis on genomic and proteomic data analysis. Data mining may contribute to biological data analysis in the following aspects

1. Semantic integration of heterogeneous, distributed genomic and proteomic databases

2. Alignment, indexing, similarity search, and comparative analysis of multiple nucleotide/protein sequences

3. Discovery of structural patterns and analysis of genetic networks and protein pathways

4. Visualization tools in genetic data analysis

5. Association and path analysis: identifying co-occurring gene sequences and linking genes to different stages of disease development