Chapter 8
Data Mining and Knowledge Discovery Methods with Case Examples

S. Bandyopadphyay and U. Maulik

8.1 Introduction

This chapter deals with the area of knowledge discovery and data mining that has emerged as an important research direction for extracting useful information from vast repositories of data of various types. The basic concepts, problems, and challenges are first briefly discussed. Some of the major data mining tasks like classification, clustering, and association rule mining are then described in some detail. This is followed by a description of some tools that are frequently used for data mining. Two case examples of supervised and unsupervised classification for satellite image analysis are presented. Finally, an extensive bibliography is provided.

Huge amount of data is generated routinely in banks, telephones, supermarkets, credit card companies, insurance, and other business transactions as well as in scientific domains. For example, AT&T handles billions of calls per day and Google searches more than four billion pages in a day which results in several terabyte orders of data. Similarly, astronomical data of the order of gigabytes per second, as well as large amount of biological data, data from e-commerce transactions, etc., are generated regularly. These data sets are not only huge but also complex and sometimes even unstructured.

Traditionally, manual methods were employed to turn data into knowledge. However, analyzing these data manually and making sense out of it is slow, expensive, subjective, and prone to errors. Hence, the need to automate the process arose, thereby leading to research in the fields of data mining and knowledge discovery. Knowledge discovery from databases (KDD) evolved as a research direction that appears at the intersection of research in databases, machine learning, pattern

S. Bandyopadphyay (✉) • U. Maulik
Indian Statistical Institute, Kolkata, India
e-mail: sanghami@isical.ac.in

R. Jiang et al. (eds.), *Basics of Bioinformatics: Lecture Notes of the Graduate Summer School on Bioinformatics of China*, DOI 10.1007/978-3-642-38951-1_8,

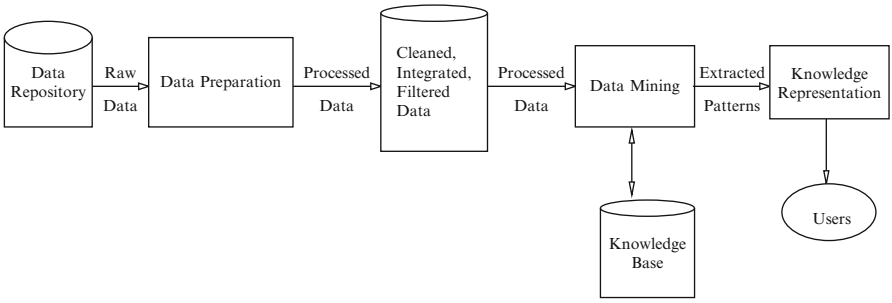

Fig. 8.1 The knowledge discovery process [15]

recognition, statistics, artificial intelligence, reasoning with uncertainty, expert systems, information retrieval, signal processing, high-performance computing, and networking [19, 30].

Data mining and knowledge discovery is the nontrivial process of extraction of valid, previously unknown, potentially useful, and ultimately understandable patterns from data. It is the science of extracting useful information from large data sets or databases. Data mining techniques attempt to use raw data for:

- Increasing business, e.g., focused marketing, inventory logistics
- Improving productivity, e.g., analysis of weather data
- Reducing costs, e.g., fraud detection
- Scientific discovery, e.g., biological applications, drug discovery

The application areas of data mining are very large. Some of these are as follows:

- Science and technology – astronomy, bioinformatics, medicine, drug discovery, etc.
- Business – customer relationship management (CRM), fraud detection, e-commerce, manufacturing, sports/entertainment, telecom, targeted marketing, health care, retail sales, etc.
- Web – search engines, advertising, web and text mining, etc.
- Government – surveillance, crime detection, profiling tax cheaters, etc.
- Natural resource study and estimation – agriculture, forestry, geology, environment, etc.
- Astronomy – mining large astronomical databases
- Image mining – content-based image retrieval from large databases

The task of knowledge discovery can generally be classified into data preparation, data mining, and knowledge presentation. Data mining is the core step where the algorithms for extracting the useful and interesting patterns are applied. In this sense, data preparation and knowledge representation can be considered, respectively, to be preprocessing and postprocessing steps of data mining. Figure 8.1 presents a schematic view of the steps involved in the process of knowledge discovery.

8.2 Different Tasks in Data Mining

Data mining tasks can be classified as descriptive and predictive [30]. While the descriptive techniques provide a summary of the data, the predictive techniques learn from the current data in order to make predictions about the behavior of new data sets. The commonly used tasks in data mining are as follows:

- Classification: predicting an item class
- Clustering: finding groups in data
- Associations: finding associations between different items
- Visualization: proper depiction of the results so as to facilitate knowledge discovery
- Summarization: describing a group of related items
- Deviation detection: finding unexpected changes in the data
- Estimation: predicting a continuous value of a variable
- Link analysis: finding relationships between items/events

The data mining tasks of classification, clustering, and association mining are described in detail in the following subsections.

8.2.1 Classification

The problem of classification involves taking an input pattern that is characterized by a set of features and making a decision about its belongingness to one (or more) of the class (classes). In case the classifier is designed using a set of labelled patterns, it is called a supervised classifier. The classification problem can be modelled in a variety of ways, e.g., by generating a set of rules, learning decision trees, generating class boundaries capable of distinguishing among the different classes. Some well-known classification methods are described below.

8.2.1.1 Nearest Neighbor Rule

A simple and well-known approach of classification is the nearest neighbor rule.

Let us consider a set of n pattern (or points) of known classification $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where it is assumed that each pattern belongs to one of the classes $C_1, C_2, \ldots, C_k$. The NN classification rule then assigns a pattern $\mathbf{x}$ of unknown classification to the class of its nearest neighbor, where $\mathbf{x}_i \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is defined to be the nearest neighbor of $\mathbf{x}$ if

$$D(\mathbf{x}_i, \mathbf{x}) = \min_l \{D(\mathbf{x}_l, \mathbf{x})\}, \quad l = 1, 2, \ldots, n \tag{8.1}$$

where D is any distance measure definable over the pattern space.

Since the above algorithm uses the class information of only the nearest neighbor to $\mathbf{x}$, it is known as the 1-NN rule. If K neighbors are considered for classification, then the scheme is termed as the K-NN rule. The K-NN rule assigns a pattern $\mathbf{x}$ of unknown classification to class C_i if the majority of the K nearest neighbors belongs to class C_i. The details of the K-NN rule along with the probability of error are available in Duda and Hart [23], Fukunaga [29], and Tou and Gonzalez [56].

8.2.1.2 Bayes Maximum Likelihood Classifier

Bayes maximum likelihood classifier [3, 56] is another well-known and widely used classifier. In most of the real-life problems, the features are usually noisy and the classes in the feature space are overlapping. In order to model such systems, the feature values $x_1, x_2, \ldots, x_j, \ldots, x_N$ are considered as random values in the probabilistic approach. The most commonly used classifier in such probabilistic systems is the Bayes maximum likelihood classifier, which is now described.

Let P_i denote the a priori probability and $p_i(\mathbf{x})$ denote the class conditional density corresponding to the class C_i $(i = 1, 2, \ldots, k)$. If the classifier decides $\mathbf{x}$ to be from the class C_i, when it actually comes from C_l, it incurs a loss equal to L_{li}. The expected loss (also called the conditional average loss or risk) incurred in assigning an observation $\mathbf{x}$ to the class C_i is given by

$$r_i(\mathbf{x}) = \sum_{l=1}^{k} L_{li}\ p\left(\frac{C_l}{\mathbf{x}}\right), \tag{8.2}$$

where $p(C_l/\mathbf{x})$ represents the probability that $\mathbf{x}$ is from C_l. Using Bayes formula, Eq. (8.2) can be written as

$$r_i(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{l=1}^{k} L_{li}\ p_l(\mathbf{x}) P_l, \tag{8.3}$$

where

$$p(\mathbf{x}) = \sum_{l=1}^{k} p_l(\mathbf{x}) P_l.$$

The pattern $\mathbf{x}$ is assigned to the class with the smallest expected loss. The classifier which minimizes the total expected loss is called the *Bayes classifier*.

Let us assume that the loss (L_{li}) is zero for correct decision and greater than zero but the same for all erroneous decisions. In such situations, the expected loss, Eq. (8.3), becomes

$$r_i(\mathbf{x}) = 1 - \frac{P_i\, p_i(\mathbf{x})}{p(\mathbf{x})}. \tag{8.4}$$

Since $p(\mathbf{x})$ is not dependent upon the class, the Bayes decision rule is nothing but the implementation of the decision functions

$$D_i(\mathbf{x}) = P_i\, p_i(\mathbf{x}), \qquad i = 1, 2, \ldots, k, \tag{8.5}$$

where a pattern $\mathbf{x}$ is assigned to class C_i if $D_i(\mathbf{x}) > D_l(\mathbf{x})$, $\forall l \neq i$. This decision rule provides the minimum probability of error. The naive Bayes classifier assumes that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$. Therefore, the decision function in Eq. 8.5 is written as

$$D_i(\mathbf{x}) = P_i \prod_{j=1}^{N} p_i(x_j), \qquad i = 1, 2, \ldots, k. \tag{8.6}$$

It is to be noted that if the a priori probabilities and the class conditional densities are estimated from a given data set, and the Bayes decision rule is implemented using these estimated values (which may be different from the actual values), then the resulting classifier is called the *Bayes maximum likelihood classifier*.

8.2.1.3 Support Vector Machines

Support vector machine (SVM) is considered to be the state-of-the-art classifier. The underlying principle of SVMs is to map the objects to a high dimensional space where the classes become linearly separable. The task then is to estimate the hyperplane that optimally separates the two classes in the high dimensional space. An interesting feature of the SVM is that the mapping to the higher dimensional space is not explicit. Rather, it is done implicitly when the inner product between two vectors $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$ in the higher dimensional space is computed as a kernel function defined over the input feature space. Here $\phi(.)$ is the mapping that transforms the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in the input feature space to the higher dimensional space. In other words,

$$\phi^T(\mathbf{x}_1)\phi(\mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2). \tag{8.7}$$

When used for pattern classification, the SVM basically uses two mathematical operations [32]:

1. Nonlinear mapping of an input vector into a high dimensional feature space. This is in accordance with Cover's theorem that states that a complex pattern-classification problem cast in a high dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space [20].
2. Construction of an optimal hyperplane for separating the transformed patterns computed in the feature space. Construction of the hyperplane is performed in accordance with the principle of structural risk minimization that has its root in the Vapnik–Chervonenkis (VC) dimension theory.

Note that here the input space refers to the original feature space while feature space refers to the transformed higher dimensional space.

Let the training set be denoted by $(\mathbf{x}_i, d_i)_{i=1}^n$, where $\mathbf{x}_i$ is the ith input pattern and d_i is its class. The SVM first learns n parameters, $\alpha_1, \alpha_2, \ldots, \alpha_n$, the Lagrange multipliers, by maximizing the objective

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{8.8}$$

subject to the constraints

$$\sum_{i=1}^{n} \alpha_i d_i = 0 \tag{8.9}$$

and

$$0 \leq \alpha_i \leq C, \text{ for } i = 1, 2, \ldots, n, \tag{8.10}$$

where C is a user-specified positive parameter. Thereafter, given an unlabelled vector $\mathbf{x}$, the SVM classifies it based on the decision function

$$f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{n} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) + b), \tag{8.11}$$

where b is a bias term computed from the already-learned Lagrange multipliers and the support vectors. Here, the support vectors are the vectors that are closest to the optimal hyperplane, and hence the most difficult to classify.

8.2.2 Clustering

Clustering [2, 22, 31, 34, 56] is an important unsupervised classification technique where a set of patterns, usually vectors in a multidimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. For this purpose, a measure of similarity which will establish a rule for assigning patterns to a particular cluster is defined. One such measure of similarity may be the Euclidean distance $\mathbf{D}$ between two patterns $\mathbf{x}$ and $\mathbf{z}$ defined by $\mathbf{D} = \|\mathbf{x} - \mathbf{z}\|$. The smaller the distance between $\mathbf{x}$ and $\mathbf{z}$, the greater is the similarity between the two and vice versa.

An alternative measure of similarity is the dot product between $\mathbf{x}$ and $\mathbf{z}$, which, physically, is a measure of the cosine of the angle between these two vectors. Formally,

$$d(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}. \tag{8.12}$$

Clearly, the smaller the Euclidean distance between these two vectors, the more similar they are, and therefore, the larger will be the inner product between them. It can be shown that minimization of the Euclidean distance corresponds to maximization of the inner product.

Let us assume that the n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, represented by the set P, are grouped into K clusters $C_1, C_2, \ldots, C_K$. Then, in general, for crisp clustering,

$$\begin{aligned} &C_i \neq \emptyset \qquad \text{for } i = 1, \ldots, K, \\ &C_i \bigcap C_j = \emptyset \quad \text{for } i = 1, \ldots, K, \quad j = 1, \ldots, K \text{ and } i \neq j, \text{ and} \\ &\bigcup_{i=1}^{K} C_i = P. \end{aligned}$$

Clustering techniques may be partitional or hierarchical [2]. Among the partitional clustering techniques, where a partitioning of the data is obtained only on termination of the algorithm, the K-means technique [56] has been one of the more widely used ones. In hierarchical clustering, the clusters are generated in a hierarchy, where every level of the hierarchy provides a particular clustering of the data, ranging from a single cluster to n clusters. A clustering technique is said to be crisp if it assigns a point to exactly one cluster. An alternate clustering strategy is fuzzy clustering where a point can have nonzero membership to more than one cluster simultaneously. Fuzzy c-means is a well-known partitional clustering technique that belongs to this category. The basic steps of the K-means, fuzzy c-means, and single-linkage hierarchical clustering algorithms are described below.

8.2.2.1 *K*-Means Algorithm

The K-means clustering algorithm essentially consists of an alternating sequence of cluster assignment followed by center update. The steps of the algorithm are as follows:

1. Choose K initial cluster centers $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K$ randomly from the n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.
2. Assign point $\mathbf{x}_m$, $m = 1, 2, \ldots, n$ to cluster C_j, $j \in \{1, 2, \ldots, K\}$ iff

$$\|\mathbf{x}_m - \mathbf{z}_j\| < \|\mathbf{x}_m - \mathbf{z}_p\|, \quad p = 1, 2, \ldots, K, \text{ and } j \neq p.$$

   Ties are resolved arbitrarily.
3. Compute new cluster centers $\mathbf{z}_1^*, \mathbf{z}_2^*, \ldots, \mathbf{z}_K^*$ as follows:

$$\mathbf{z}_i^* = \frac{1}{n_i} \Sigma_{x_j \in C_i} \mathbf{x}_j \quad i = 1, 2, \ldots, K,$$

   where n_i is the number of elements belonging to cluster C_i.
4. If $\mathbf{z}_i^* = \mathbf{z}_i$, $i = 1, 2, \ldots, K$ then terminate. Otherwise, continue from Step 2.

Note that in case the K-means algorithm does not terminate normally, it is executed for a predefined maximum number of iterations.

Sometimes, the K-means algorithm may converge to some local optima [54]. Moreover, global solutions of large problems cannot be found within a reasonable amount of computation effort [55]. As a result, several approximate methods, including genetic algorithms and simulated annealing [8, 14, 41], are developed to solve the underlying optimization problem. These methods have also been extended to the case where the number of clusters is variable [7, 42], and to fuzzy clustering [43].

The K-means algorithm is known to be sensitive to outliers, since such points can significantly affect the computation of the centroids, and hence the resultant partitioning. K-medoid attempts to alleviate this problem by using the medoid, the most centrally located object, as the representative of the cluster. Partitioning around medoid (PAM) [36] was one of the earliest K-medoid algorithms introduced. PAM finds K clusters by first finding a representative object for each cluster, the medoid. The algorithm then repeatedly tries to make a better choice of medoids by analyzing all possible pairs of objects such that one object is a medoid and the other is not. PAM is computationally quite inefficient for large data sets and a large number of clusters. The Clustering LARge Applications (CLARA) algorithm was proposed in Kaufman and Rousseeuw [36] to tackle this problem. CLARA is based on data sampling, where only a small portion of the real data is chosen as a representative of the data, and medoids are chosen from this sample using PAM. CLARA draws multiple samples and outputs the best clustering from these samples. As expected, CLARA can deal with larger data sets than PAM. However, if the best set of medoids is never chosen in any of the data samples, CLARA will never find the best clustering. The CLARANS algorithm proposed in Ng and Han [44] tries to mix both PAM and CLARA by searching only a subset of the data set. However, unlike CLARA, CLARANS does not confine itself to any sample at any given time, but draws it randomly at each step of the algorithm. Based upon CLARANS, two spatial data mining algorithms, the spatial dominant approach, SD(CLARANS), and the nonspatial dominant approach, NSD(CLARANS), were developed. In order to make CLARANS applicable to large data sets, use of efficient spatial access methods, such as R*-tree, was proposed [24]. CLARANS has a limitation that it can provide good clustering only when the clusters are mostly equisized and convex. DBSCAN [25], another popularly used density clustering technique that was proposed by Ester et al., could handle nonconvex and nonuniformly sized clusters. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), proposed in Zhang et al. [61], is another algorithm for clustering large data sets. It uses two concepts, the clustering feature and the clustering feature tree, to summarize cluster representations which help the method achieve good speed and scalability in large databases. Discussion on several other clustering algorithms may be found in Han and Kamber [30].

8.2.2.2 Fuzzy c-Means Clustering Algorithm

Fuzzy c-means (FCM) [17] is a widely used technique that uses the principles of fuzzy sets to evolve a fuzzy partition matrix for a given data set. The set of all $c \times n$, where c is equal to the number of clusters, nondegenerate constrained fuzzy partition matrices, denoted by M_{fcn}, is defined as

$$M_{fcn} = \{U \in R^{c \times n} \mid \sum_{i=1}^{c} u_{ik} = 1, \quad \sum_{k=1}^{n} u_{ik} > 0, \ \forall i \ \text{ and } \\ u_{ik} \in [0, 1]; 1 \leq i \leq c; 1 \leq k \leq n\}. \tag{8.13}$$

Here, u_{ik} is the membership of the kth point to the ith cluster. The minimizing criterion used to define good clusters for fuzzy c-means partitions is the FCM function defined as

$$J_{\mu}(U, Z) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^{\mu} D_{ik}^{2}. \tag{8.14}$$

Here, $U \in M_{fcn}$ is a fuzzy partition matrix; $\mu \in [1, \infty]$ is the weighting exponent on each fuzzy membership; $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_c]$ represents c cluster centers; $\mathbf{z}_i \in IR^N$; and D_{ik} is the distance of $\mathbf{x}_k$ from the ith cluster center. The fuzzy c-means theorem [17] states that if $D_{ik} > 0$, for all i and k, then (U, Z) may minimize J_{μ}, only if when $\mu > 1$

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D_{ik}}{D_{jk}}\right)^{\frac{2}{\mu - 1}}}, \quad \text{for } 1 \leq i \leq c, \ \ 1 \leq k \leq n, \tag{8.15}$$

and

$$\mathbf{z}_i = \frac{\sum_{k=1}^{n} (u_{ik})^{\mu} \mathbf{x}_k}{\sum_{k=1}^{n} (u_{ik})^{\mu}}, \quad 1 \leq i \leq c. \tag{8.16}$$

A common strategy for generating the approximate solutions of the minimization problem in Eq. (8.14) is by iterating through Eqs. (8.15) and (8.16) (also known as the Picard iteration technique). A detailed description of the FCM algorithm may be found in Bezdek [17].

Note that in fuzzy clustering, although the final output is generally a crisp clustering, the users are free to still utilize the information contained in the partition matrix. The FCM algorithm shares the problems of the K-means algorithm in that it also gets stuck at local optima depending on the choice of the initial clusters, and requires the number of clusters to be specified a priori.

8.2.2.3 Single Linkage Hierarchical Clustering Technique

The single-linkage clustering scheme is a noniterative method based on a local connectivity criterion [34]. Instead of an object data set X, single linkage processes sets of n^2 numerical relationships, say $\{r_{jk}\}$, between pairs of objects represented by the data. The value r_{jk} represents the extent to which object j and k are related in the sense of some binary relation ρ. It starts by considering each point in a cluster of its own. The single-linkage algorithm computes the distance between two clusters C_i and C_j as

$$\delta_{SL}(C_i, C_j) = \underbrace{\min}_{x \in C_i, y \in C_j} \{d(x, y)\}, \tag{8.17}$$

where $d(x, y)$ is some distance measure defined between objects x and y. Based on these distances, it merges the two closest clusters, replacing them by the merged cluster. The distance of the remaining clusters from the merged one is recomputed as above. The process continues until a single cluster, comprising all the points, is formed. The advantages of this algorithm are that (1) it is independent of the shape of the cluster, and (2) it works for any kind of attributes, both categorical and numeric, as long as a similarity of the data objects can be defined. However, the disadvantages of this method are its computational complexity and its inability to handle overlapping classes.

8.2.3 Discovering Associations

The root of the association rule mining (ARM) [30] problem lies in the market basket or transaction data analysis. Association analysis is the discovery of rules showing attribute–value associations that occur frequently.

Let us assume $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n items and X be an itemset where $X \subset I$. A k-itemset is a set of k items. Let $T = \{(t_1, X_1), (t_2, X_2) \ldots, (t_m, X_m)\}$ be a set of m transactions, where t_i and X_i, $i = 1, 2, \ldots, m$, are the transaction identifier and the associated itemset, respectively. The *cover* of an itemset X in T is defined as follows:

$$cover(X, T) = \{t_i | (t_i, X_i) \in T, X \subset X_i\}. \tag{8.18}$$

The *support* of an itemset X in T is

$$support(X, T) = |cover(X, T)|, \tag{8.19}$$

and the *frequency* of an itemset is

$$frequency(X, T) = \frac{support(X, T)}{|T|}. \tag{8.20}$$

In other words, support of an itemset X is the number of transactions where all the items in X appear in each transaction. The frequency of an itemset represents the probability of its occurrence in a transaction in T. An itemset is called frequent if its support in T is greater than some threshold min_sup. The collection of frequent itemsets with respect to a minimum support min_sup in T, denoted by $\mathcal{F}(T, min\_sup)$, is defined as

$$\mathcal{F}(T, min\_sup) = \{X \subset I, support\ (X, T) > min\_sup\}. \tag{8.21}$$

The objective in association rule mining is to find all rules of the form $X \Rightarrow Y$, $X \bigcap Y = \emptyset$ with probability $c\,\%$, indicating that if itemset X occurs in a transaction, the itemset Y also occurs with probability $c\,\%$. X is called the *antecedent* of the rule and Y is called the *consequent* of the rule. Support of a rule denotes the percentage of transactions in T that contains both X and Y. This is taken to be the probability $P(X \bigcup Y)$. An association rule is called *frequent* if its support exceeds a minimum value min_sup.

The confidence of a rule $X \Rightarrow Y$ in T denotes the percentage of the transactions in T containing X that also contains Y. It is taken to be the conditional probability $P(X|Y)$. In other words,

$$confidence(X \Rightarrow Y, T) = \frac{support(X \bigcup Y, T)}{support(X, T)}. \tag{8.22}$$

A rule is called *confident* if its confidence value exceeds a threshold min_conf. The problem of association rule mining can therefore be formally stated as follows: *Find the set of all rules* R *of the form* $X \Rightarrow Y$ *such that*

$$R = \{X \Rightarrow Y | X, Y \subset I, X \bigcap Y = \emptyset, X \bigcup Y \in \mathcal{F}(T, min\_sup), \\ confidence(X \Rightarrow Y, T) > min\_conf\}. \tag{8.23}$$

The association rule mining process, in general, consists of the following two steps:

1. Finding all frequent itemsets
2. Generating strong association rules from the frequent itemsets

It may be noted that the number of itemsets grows exponentially with the number of items, $|I|$, and therefore, generating all the frequent itemsets is a challenging problem. *Apriori* algorithm [1] is commonly used for this purpose. The *Apriori* algorithm finds frequent itemsets of length k from frequent itemsets of length $k-1$. The important concept, utilized for pruning many unnecessary searches, is that if an itemset I of length $k-1$ is not frequent, then all itemsets I' such that $I \in I'$ cannot be frequent, and hence, this branch of the tree can be effectively pruned from the search space.

Table 8.1 A set of six transactions defined over a set of four items $I = \{i_1, i_2, i_3, i_4\}$

$I = \{i_1, i_2, i_3, i_4\}$

Transaction	Items
t_1	i_1, i_3
t_2	i_1, i_2, i_3
t_3	i_1, i_2
t_4	i_1, i_4
t_5	i_1, i_2, i_3, i_4
t_6	i_2, i_4

Consider the example shown in Table 8.1. Here, there are four items and six transactions. Let the problem be to identify the associations, if any, that can be extracted from this example. Let the $min\_sup = 2$, and min_conf of a rule be 0.9. From the example, it can be seen that

$$cover(i_1, i_3) = \{t_1, t_2, t_5\} \text{ and } cover(i_1, i_2) = \{t_2, t_3, t_5\}.$$

Hence,

$$support(i_1, i_3) = 3 \text{ and } support(i_1, i_2) = 3,$$

both of which exceed the min_sup, and hence, they are frequent itemsets. In fact, it can be verified that only these two are the largest frequent itemsets. Therefore, the rules to be evaluated are

$$(1)\ R_1 :\ i_1 \Rightarrow i_3 \quad (2)\ R_2 :\ i_3 \Rightarrow i_1 \quad (3)\ R_3 :\ i_1 \Rightarrow i_2 \quad (4)\ R_4 :\ i_2 \Rightarrow i_1$$

It can be easily shown that

$$(1)\ confidence(R_1) = \tfrac{3}{5} = 0.6 \quad (2)\ confidence(R_2) = \tfrac{3}{3} = 1.0$$
$$(3)\ confidence(R_3) = \tfrac{3}{5} = 0.6 \quad (4)\ confidence(R_4) = \tfrac{3}{4} = 0.75.$$

Hence, only rule R_2, i.e., $i_3 \Rightarrow i_1$, is above the confidence threshold. So ARM will return R_2 as the discovered association rule.

8.2.4 Issues and Challenges in Data Mining

The earlier mining algorithms were usually applied on data that had a fixed structure and were relatively simple. With the advent of new technologies, the data being collected nowadays is increasingly unstructured, complex, and high dimensional data [15]. Typical domains from where such data are routinely collected are:

1. Biometrics data, e.g., fingerprints, iris, face
2. Biological data, e.g., DNA and RNA sequences, 3-D protein structure, gene regulatory networks
3. Web data, e.g., hyperlink structure, user profiles, access profiles

4. Business data, e.g., credit card information, mobile company data, stock market data
5. Meteorological data, e.g., weather patterns, cloud cover

Because of the sheer quantity of data collected, it is often the case that some data might not be collected and/or noise might be inadvertently introduced. For example, the weather information of a particular day might not be collected simply because of unavailability of a technician, or noise might be introduced while sequencing a genome. Performing data mining under such circumstances necessitates the use of sophisticated data cleaning, integration, and estimation methods. The mining algorithms should also be extremely robust to noise and outliers, and scalable and adaptive to new information.

The purpose of knowledge discovery is to identify interesting patterns. The definition of "interestingness" is itself subjective and dependent on the application domain. Thus, automatic identification of interesting patterns becomes extremely difficult. Moreover, if the data and the environment are frequently changing, e.g., stock market time series data and weather data, then designing suitable algorithms that can dynamically adapt their goals is an important issue.

While huge amount of information presents a challenge to data mining in many domains, lack of it throws an equally important challenge to algorithm designers in several application areas. For example, in gene expression data analysis, an important task is to predict the cancer type of an individual from the expression level of thousands of genes. The training data in many such situations might be the expression values of thousands of genes for a few hundreds of patients, or even less. Such applications demand highly sophisticated feature selection methods that can identify only those features that are relevant for the task in hand. Another important issue in data mining arises due to the fact that some events are much rarer than some others making the different classes in the data highly unbalanced. Intrusion detection is one such area where intrusions are relatively rare, and hence, learning their characteristics and identifying them accurately are difficult tasks. Moreover, in many domains, the cost of error is not the same irrespective of the class. For example, in particular cases, the cost of a false-positive prediction might be less than a false-negative prediction. Designing algorithms that can suitably weight the errors is therefore an important issue.

Distributed data mining [35], where the data is distributed over several sites, has become extremely important in recent times. The data might be distributed horizontally in such a way that the schema viewed in every site is the same. Alternatively, the data might be distributed vertically where each site has a different schema and a different view of the data. Though collecting all the data at a central site and then executing the algorithms is a possible way of dealing with such situations, it is evidently highly inefficient. Moreover, privacy and security are nowadays top priority in many organizations, e.g., credit card companies and pharmaceutical organizations, and hence, sharing local data is not a feasible proposition for them. Thus, designing algorithms that can carry out the computation

in a distributed, secure, and privacy-preserving manner is of serious concern and a major research thrust worldwide. Other issues of importance are designing mining algorithms over peer-to-peer networks [16], grids, and cluster computers.

8.3 Some Common Tools and Techniques

In this section, some commonly used optimization, learning, and representation tools are discussed in brief.

8.3.1 Artificial Neural Networks

Artificial neural networks (ANN) [21, 32, 49] are a parallel and layered interconnected structure of a large number of artificial neurons, each of which constitutes an elementary computational primitive. According to Kohonen [37], ANNs are massively parallel adaptive networks of simple nonlinear computing elements called neurons which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strengths. The distributed representation of the interconnections through massive parallelism achieved out of the inherent network structure bestows upon such networks properties of graceful degradation and fault tolerance.

In the most general form, an ANN is a layered structure of neurons. The neurons can be of three types, namely, input, hidden, and output. The input neurons are designated to accept stimuli from the external world. The output neurons generate the network outputs. The hidden neurons, which are shielded from the external world, are entrusted with the computation of intermediate functions necessary for the operation of the network. A signal function operates within the neurons that generates its output signal based on its activation. In general, these activation functions take an input as an infinite range of activations $(-\infty, +\infty)$ and transform them in the finite range $[0, 1]$ or $[-1, +1]$.

The neurons are connected based on an interconnection topology which basically house the memory of the network. These connections may be excitatory (+), inhibitory (−) or absent (0). Based on the signals received on its input connection and the signal function applicable for the neuron, its output is computed. Neural networks posses the ability to learn from examples. The learning rule provides the basis for modifying the network dynamics with an aim to improve its performance. Learning rules/algorithms define an architecture-dependent procedure to encode pattern information into interneuron interconnections. Learning in a neural network is data driven and proceeds by modifying these connection weights. Some well-known models of ANNs, distinguished by the interconnection topology, activation function, and learning rules, are the multilayer perceptron (MLP), self-organizing map (SOM), and Hopfield network [32].

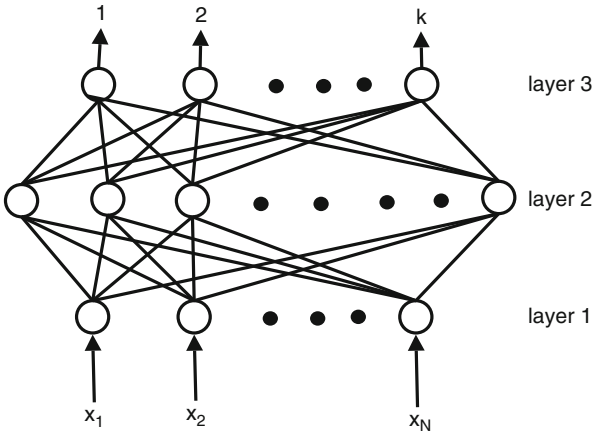

Fig. 8.2 A typical MLP [10]

A multilayer perceptron (MLP) consists of several layers of simple neurons with full connectivity existing between neurons of adjacent layers. Figure 8.2 shows an example of a three-layer MLP which consists of an input layer (*layer 1*), one hidden layer (*layers 2*), and an output layer (*layer 3*).

The neurons in the input layer serve the purpose of fanning out the input values to the neurons of *layer 2*. Let

$$w_{ji}^{(l)}, \quad l = 2,\ 3 \tag{8.24}$$

represent the connection weight on the link from the ith neuron in layer $l-1$ to the jth neuron in layer l. Let $\theta_j^{(l)}$ represent the threshold or the bias of the jth neuron in layer l. The total input, $x_j^{(l)}$, received by the jth neuron in layer l is given by

$$x_j^{(l)} = \sum_i y_i^{(l-1)}\, w_{ji}^{(l)} + \theta_j^{(l)}, \quad l = 2,\ 3, \tag{8.25}$$

where $y_i^{(l-1)}$ is the output of the ith neuron in layer $l-1$. For the input layer,

$$y_i^{(1)} = x_i, \tag{8.26}$$

where x_i is the ith component of the input vector. For the other layers,

$$y_i^{(l)} = f(x_i^{(l)}) \qquad l = 2,\ 3. \tag{8.27}$$

Several functional forms like threshold logic, hard limiter, and sigmoid can be used for $f(.)$.

There are several algorithms for training the network in order to learn the connection weights and the thresholds from a given training data set. Backpropagation (BP)

is one such learning algorithm, where the least mean square error of the network output is computed, and this is propagated in a top-down manner (i.e., from the output side) in order to update the weights. The error is computed as the difference between the actual and the desired output when a known input pattern is presented to the network. A gradient descent method along the error surface is used in BP. More information on MLP and other neural methods are available in Dayhoff [21], Haykin [32], Lippmann [39], and Pao [49].

8.3.2 Fuzzy Sets and Fuzzy Logic

Much of the information available in the real world are not numeric in nature and hence cannot be precisely measured. The means of human communication is inherently vague, imprecise, and uncertain. Fuzzy logic was developed in order to mathematically model this vagueness and imprecision. The fuzzy set theory, introduced by Zadeh [60], explains the varied nature of ambiguity and uncertainty that exist in the real world. This is in sheer contradiction to the concept of crisp sets, where information is more often expressed in quantifying propositions. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth, i.e., truth values between completely true and completely false.

A generic fuzzy system comprises the following modules. A fuzzification interface fuzzifies the numeric crisp inputs by assigning grades of membership using fuzzy sets defined for the input variable. A fuzzy rule base/knowledge base comprises a data-derived or heuristic rule base. The data-derived rule base is usually generated by clustering techniques or neural networks using sensor databases. The heuristic rule base, on the other hand, is generated by human experts through some intuitive mechanisms. A fuzzy inference engine infers fuzzy outputs by employing the fuzzy implications and the rules of inference of fuzzy logic. Finally, a defuzzification interface is present that yields a non-fuzzy crisp control action from an inferred fuzzy control action.

Fuzzy set theory has found a lot of applications in data mining [4, 50, 59]. Examples of such applications may be found in clustering [38], association rules [57], time series [18], and image retrieval [28].

8.3.3 Genetic Algorithms

Genetic algorithms (GAs) [33] are randomized search and optimization technique guided by the principles of natural genetic systems. These algorithms are characterized by a population of encoded trial solutions and a collection of operators to act on the population. The basic philosophy behind these algorithms is to encode the parameters of the problems and then parallely search the space of

the encoded solutions by the application of the embedded operators so as to arrive at an optimal solution. Generally, two types of operators are used, namely, reproduction and evolution. The reproduction operator is guided by a selection mechanism. The evolution operator includes the crossover and mutation operators. The search technique is implemented through a series of iterations, whereby the different operators are applied in a loop on the initial population. Each iteration is referred to as a generation. Each generation produces a new solution space, which are selectively chosen for participating in the next generation of the optimization procedure. The selection of the participating solutions for the next generation is decided by a figure of merit, often referred to as the fitness function. The essential components of GAs are the following:

- A representation strategy that determines the way in which potential solutions will be coded to form string-like structures called *chromosomes*
- A population of *chromosomes*
- Mechanism for evaluating each chromosome
- Selection/reproduction procedure
- Genetic operators
- Probabilities of performing genetic operations

It operates through a simple cycle of

1. evaluation of each chromosome in the population to get the fitness value,
2. selection of chromosomes, and
3. genetic manipulation to create a new population of chromosomes,

over a number of iterations (or generations) till one or more of the following termination criteria is satisfied:

- The average fitness value of a population becomes more or less constant over a specified number of generations.
- A desired objective function value is attained by at least one string in the population.
- The number of iterations is greater than some predefined threshold.

A schematic diagram of the basic structure of a genetic algorithm is shown in Fig. 8.3.

Applications of genetic algorithms and related techniques in data mining include extraction of association rules [40], predictive rules [26, 27], clustering [7, 8, 14, 41–43], program evolution [53], and web mining [45, 46, 51, 52].

8.4 Case Examples

In this section, we provide two case examples of the application of the above-mentioned tools and techniques for solving two real-world problems. They deal with the applications of genetic algorithms for supervised classification and clustering, respectively, both dealing with analysis of remote sensing imagery.

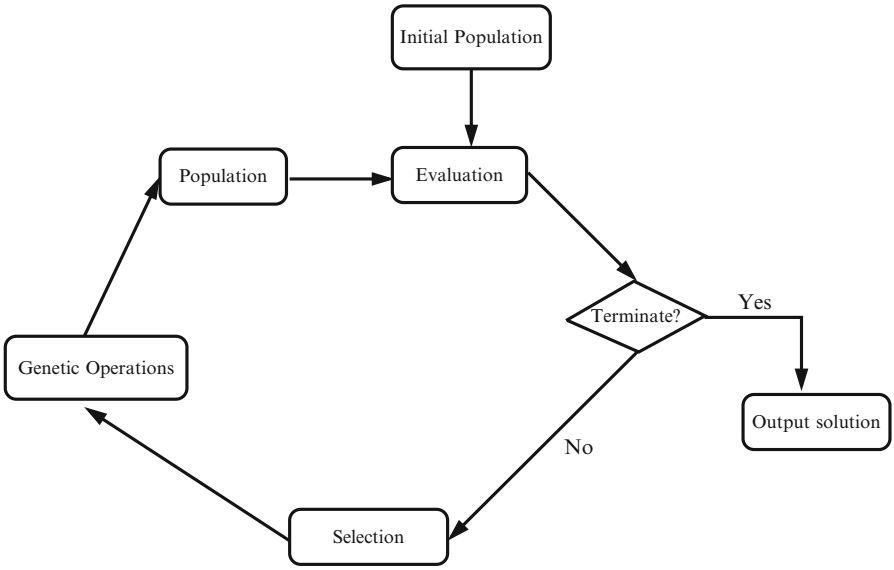

Fig. 8.3 Basic steps of a genetic algorithm [10]

8.4.1 Pixel Classification

The problem of classification can be viewed as one of generating decision boundaries that can successfully distinguish the various classes in the feature space. In real-life problems, the boundaries between the different classes are usually nonlinear. In this section, a classifier, called *GA-classifier*, that utilizes the characteristics of GAs in searching for a number of linear segments which can approximate the nonlinear boundaries while providing minimum misclassification of training sample points is described [11, 48].

The *GA-classifier* attempts to place H hyperplanes in the feature space appropriately such that the number of misclassified training points is minimized. From elementary geometry, it can be derived that a hyperplane in N-dimensional space can be represented by $N-1$ angle variables and a perpendicular distance variable. These are encoded in a chromosome in the GA. Thus, each chromosome encoding H hyperplanes is of length $l = H((N-1) * b_1 + b_2)$, where b_1 and b_2 are the numbers of bits used for encoding an angle and the perpendicular distance, respectively.

The computation of the fitness is done for each string in the population. The fitness of a string is characterized by the number of points it misclassifies. A string with the lowest misclassification is therefore considered to be the fittest among the population of strings. If the number of misclassified points for a string is denoted by $miss$, then the fitness of the string is computed as $(n - miss)$, where n is the number of training data points. The best string of each generation or iteration is the one

which has the fewest misclassifications. This string is stored after each iteration. If the best string of the previous generation is found to be better than the best string of the current generation, then the previous best string replaces the worst string of the current generation. This implements the *elitist strategy*, where the best string seen up to the current generation is propagated to the next generation.

Since it is difficult to estimate the proper number of hyperplanes a priori, in Bandyopadhyay et al. [13], the concept of variable H and hence variable chromosome length was introduced. This resulted in a classifier called *VGA-classifier*. The chromosomes in the *VGA-classifier* are represented by strings of 1, 0, and # (don't care), encoding the parameters of variable number of hyperplanes. The fitness of string i, encoding H_i hyerplanes, is defined as

$$\text{fit}_i = (n - miss_i) - \alpha H_i,\ 1 \leq H_i \leq H_{\max}, \tag{8.28}$$

$$= \quad 0, \qquad \text{otherwise}, \tag{8.29}$$

where n = size of the training data set and $\alpha = \frac{1}{H_{\max}}$. Here, $H_{\max}$ is the maximum number of hyperplanes that are considered for approximating the decision boundary. Note that the maximization of the fitness function leads to minimization of the number of misclassified points and also minimization of the number of hyperplanes. The genetic operators of crossover and mutation are redefined appropriately so as to tackle the variable length chromosomes [13].

The *VGA-classifier* was further enhanced in Bandyopadhyay and Pal [9] by the incorporation of the concept of chromosome differentiation proposed in Bandyopadhyay et al. [12]. This resulted in a classifier called *VGACD-classifier*. Here, two classes of chromosomes exist in the population. Crossover (mating) is allowed only between individuals belonging to these categories. A schema analysis of GAs with chromosome differentiation (GACD), vis á vis that of conventional GA (CGA), was conducted in Bandyopadhyay et al. [12] which showed that in certain situations, the lower bound of the number of instances of a schema sampled by GACD is greater than or equal to that of CGA.

The *VGACD-classifier* was used in Bandyopadhyay and Pal [9] to classify a 512×512 *SPOT* image of a part of the city of Calcutta that is available in three bands. The image in the near-infrared band is shown in Fig. 8.4. The design set comprises 932 points belonging to 7 classes that are extracted from the above image. A two-dimensional scatter plot of the training data is shown in Fig. 8.5. The seven classes are *turbid water* (TW), *pond water* (PW), *concrete* (Concr.), *vegetation* (Veg), *habitation* (Hab), *open space* (OS), and *roads* (including bridges) (B/R). The classifier trained using the design set is then utilized for classifying the pixels in the 512×512 image.

Figure 8.6 shows the full Calcutta image classified using the *VGACD-classifier* [9]. As can be seen, most of the landmarks in Calcutta have been properly classified. The optimum number of hyperplanes was found automatically to be equal to 13. The performance of the *VGACD-classifier* was studied in comparison with those of the

Fig. 8.4 *SPOT* image of Calcutta in the near-infrared band

VGA-classifier, K-NN rule, and Bayes maximum likelihood classifier. It showed that the *VGACD-classifier* provided the best performance, and its rate of convergence was also enhanced over the *VGA-classifier* [9].

8.4.2 Clustering of Satellite Images

The purpose of any clustering technique is to evolve a $K \times n$ partition matrix $U(X)$ of a data set X ($X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$) in IR^N, representing its partitioning into a number, say K, of clusters $(C_1, C_2, \ldots, C_K)$. The partition matrix $U(X)$ may be represented as $U = [u_{kj}]$, $k = 1, \ldots, K$ and $j = 1, \ldots, n$, where u_{kj} is the membership of pattern $\mathbf{x}_j$ to cluster C_k. Clustering methods are also often categorized as crisp and fuzzy. In crisp clustering, a point belongs to exactly one cluster, while in fuzzy clustering, a point can belong to more than one cluster with varying degrees of membership.

The different clustering methods, in general, try to optimize some measure of goodness of a clustering solution either explicitly or implicitly. The clustering problem can therefore be mapped to one of searching for an appropriate number of suitable partitions such that some goodness measure is optimized. It may be

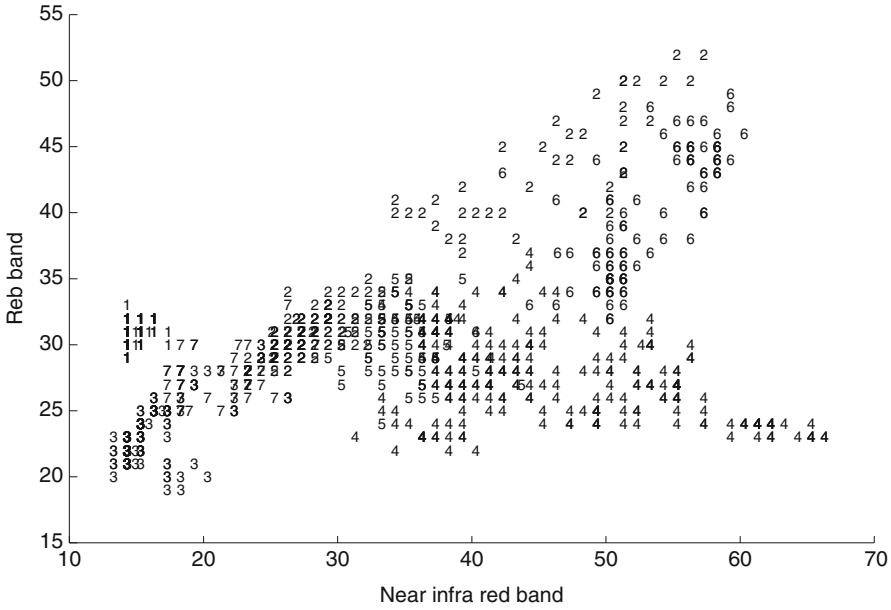

Fig. 8.5 Scatter plot for a training set of *SPOT image* of Calcutta containing seven classes (1,. . .,7)

noted that searching the exhaustive set of all possible partitions of the data is prohibitively exhaustive since the search space is huge and complex, with numerous local optima. Consequently, heuristic search methods are often employed in this domain, and these often get stuck at local optima. Several attempts have also been made to investigate the effectiveness of GAs and related methods for clustering [5–7, 14, 41, 43, 47].

As earlier, variable length GAs (VGAs) are used for automatically evolving the near-optimal nondegenerate fuzzy partition matrix U^* [43]. The centers of a variable number of clusters are encoded in the chromosomes. Given a set of centers encoded in the chromosome, the fuzzy partition matrix is computed using Eq. (8.15). Thereafter, the centers are updated using Eq. (8.16). For the purpose of optimization, the Xie–Beni [58] cluster validity index is used. The Xie–Beni (XB) index is defined as a function of the ratio of the total variation σ to the minimum separation sep of the clusters. Here, σ and sep can be written as

$$\sigma(U, Z; X) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^2 D^2(\mathbf{z}_i, \mathbf{x}_k), \tag{8.30}$$

and

$$sep(Z) = \min_{i \neq j} \{||\mathbf{z}_i - \mathbf{z}_j||^2\}, \tag{8.31}$$

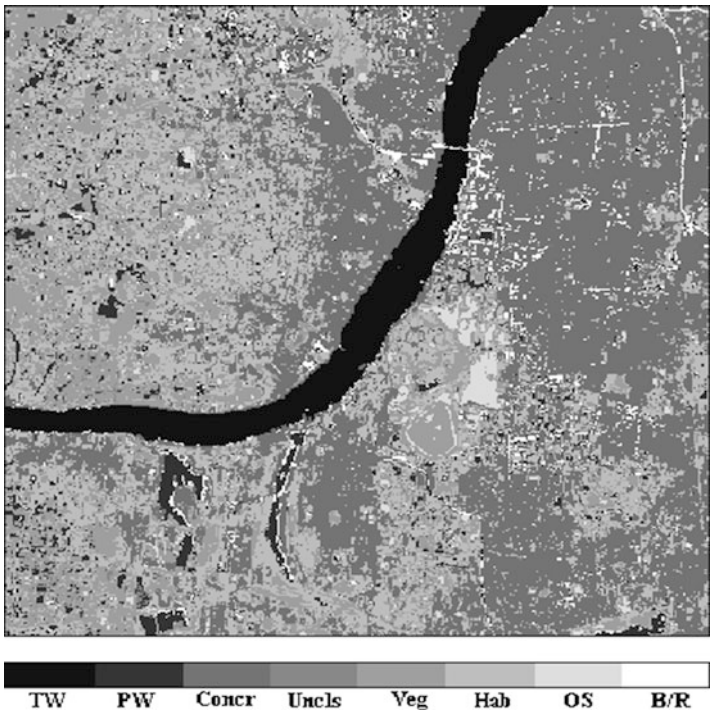

Fig. 8.6 Classified *SPOT* image of Calcutta using the *VGACD-classifier* ($H_{\max}$ =15, final value of H=13)

where $||.||$ is the Euclidean norm, and $D(\mathbf{z}_i, \mathbf{x}_k)$ is the distance between the pattern $\mathbf{x}_k$ and the cluster center $\mathbf{z}_i$. The XB index is then written as

$$XB(U, Z; X) = \frac{\sigma(U, Z; X)}{n\ sep(Z)} = \frac{\sum_{i=1}^{c}(\sum_{k=1}^{n} u_{ik}^2 D^2(\mathbf{z}_i, \mathbf{x}_k))}{n(\min_{i \neq j}\{||\mathbf{z}_i - \mathbf{z}_j||^2\})}. \tag{8.32}$$

Note that when the partitioning is compact and good, value of σ should be low while sep should be high, thereby yielding lower values of the XB index. The objective is therefore to minimize the XB index for achieving proper clustering. In other words, the best partition matrix U^* is the one such that

$$U^* \in \mathcal{U} \text{ and } XB(U^*, Z^*, X) = \min_{U_i \in \mathcal{U}} XB(U_i, Z_i, X), \tag{8.33}$$

where Z^* represents the set of cluster centers corresponding to U^*. Here, both the number of clusters as well as the appropriate fuzzy clustering of the data are evolved simultaneously using the search capability of genetic algorithms. The chromosome representation and other genetic operators used are described in detail in Maulik and Bandyopadhyay [43].

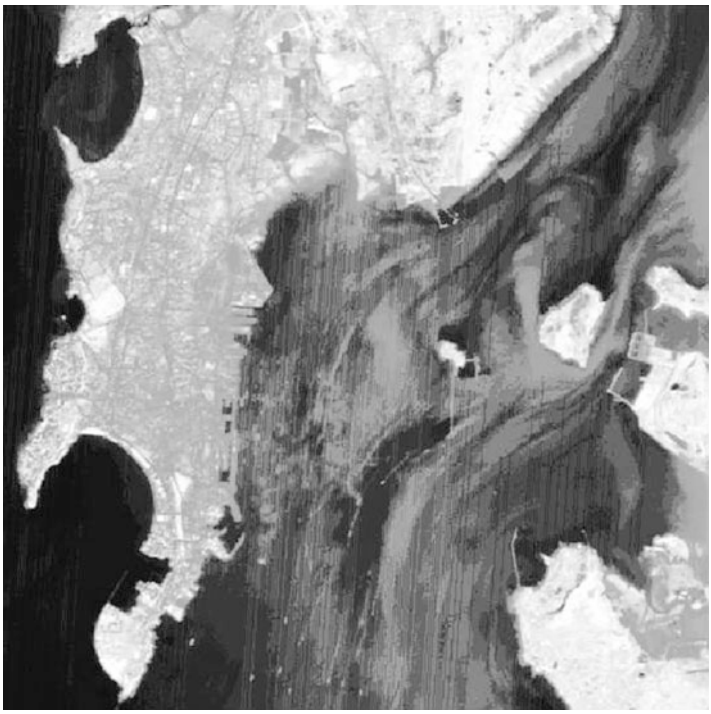

Fig. 8.7 IRS image of Mumbai in the near-infrared band with histogram equalization

The effectiveness of the genetic fuzzy clustering technique in partitioning the pixels into different landcover types is demonstrated on an Indian remote sensing (IRS) satellite image of a part of the city of Mumbai [43] (Fig. 8.7 shows the image in the near-infrared band). Detailed description is available in Maulik and Bandyopadhyay [43]. The segmented Mumbai image is shown in Fig. 8.8. The method automatically yielded seven clusters, which are labelled concrete (Concr.), open spaces (OS1 and OS2), vegetation (Veg), habitation (Hab), and turbid water (TW1 and TW2), based on the ground information available from earlier studies. The classes Hab, referring to the regions which have concrete structures and buildings, but with relatively lower density than the class concrete, and Concr. share common properties. Figure 8.8 shows that the large water body of the Arabian Sea has been distinguished into two classes which are named TW1 and TW2. The islands, dockyard, and several road structures have mostly been correctly identified in the image. Within the islands, as expected, there is a predominance of open space and vegetation. The southern part of the city, which is heavily industrialized, has been classified as primarily belonging to habitation and concrete. Some confusion within these two classes, namely, Hab and Concr, is observed (as reflected in the corresponding label); this is expected since these two classes are somewhat similar.

Figure 8.9 demonstrates the Mumbai image partitioned into seven clusters using the FCM technique. As can be seen, the water of the Arabian Sea has been

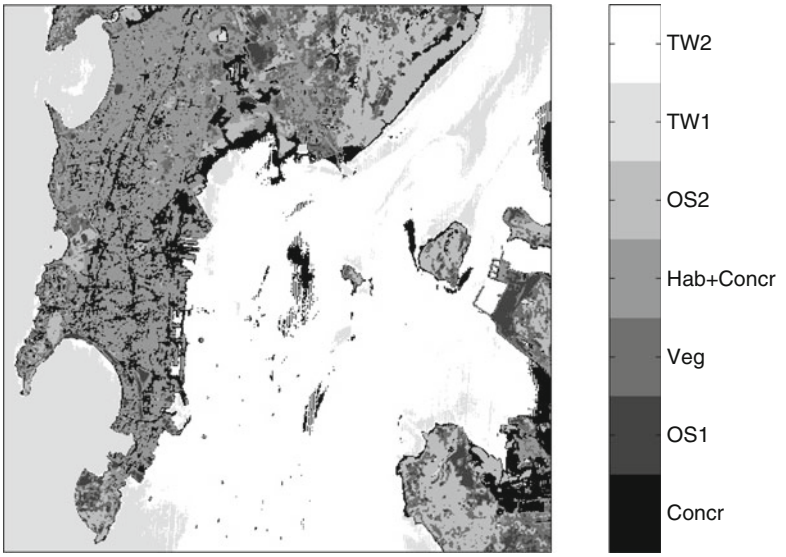

Fig. 8.8 Clustered image of Mumbai using genetic fuzzy clustering

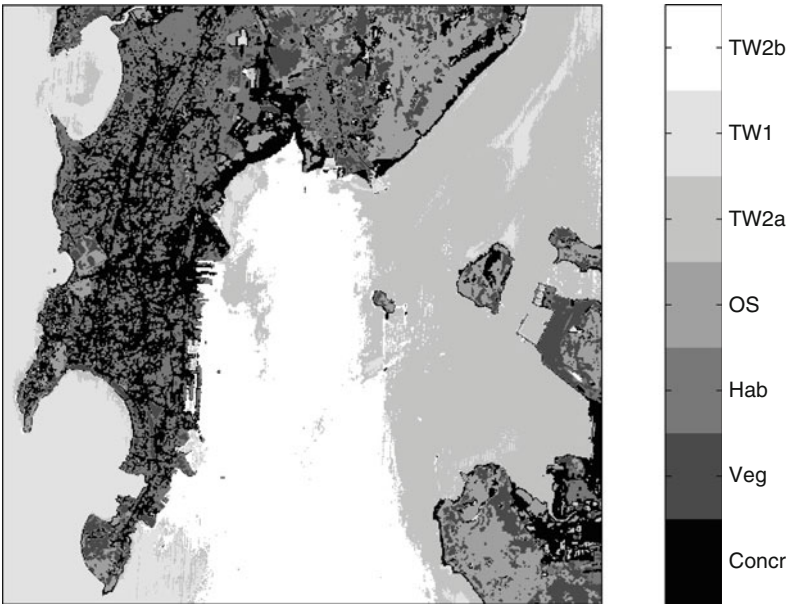

Fig. 8.9 Clustered image of Mumbai using FCM clustering

partitioned into three regions rather than two as obtained earlier. The other regions appear to be classified more or less correctly for this data. It was observed that the FCM algorithm gets trapped at local optima often enough, and the best value of the XB index was worse than that obtained using the genetic fuzzy clustering scheme.

8.5 Discussion and Conclusions

The basic concepts and issues in data mining and knowledge discovery have been discussed in this chapter. The challenges being faced by data miners, namely, very high dimensional and extremely large data sets, unstructured and semi-structured data, temporal and spatial patterns, and heterogeneous data, are mentioned. Some major data mining tasks are discussed with emphasis on algorithms developed for solving them. These include description of classifiers based on Bayes rule; nearest neighbor rule and support vector machines; clustering algorithms like K-means, fuzzy c-means, and single linkage; and association rule mining methods. The utility of some tools like neural networks, genetic algorithms, and fuzzy sets that are frequently used in data mining is discussed. Finally, two case examples are presented.

Traditional data mining generally involved well-organized database systems such as relational databases. With the advent of sophisticated technology, it is now possible to store and manipulate very large and complex data. The data complexity arises due to several reasons, e.g., high dimensionality, semi- and/or unstructured nature, and heterogeneity. Data related to the World Wide Web, the geoscientific domain, VLSI chip layout and routing, multimedia, financial markets, sensor networks, and genes and proteins constitute some typical examples of complex data. In order to extract knowledge from such complex data, it is necessary to develop advanced methods that can exploit the nature and representation of the data more efficiently. In Bandyopadhyay et al. [15], several such interesting attempts for knowledge discovery from complex data have been described.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Bocca JB, Jarke M, Zaniolo C (eds) Proceedings of the 20th international conference on very large data bases, VLDB, Morgan Kaufmann, Bases, VLDB. Morgan Kaufmann, pp 487–499. citeseer.ist.psu.edu/agrawal94fast.html
2. Anderberg MR (1973) Cluster analysis for application. Academic, New York
3. Anderson TW (1958) An introduction to multivariate statistical analysis. Wiley, New York
4. Baldwin JF (1996) Knowledge from data using fuzzy methods. Pattern Recognit Lett 17: 593–600

5. Bandyopadhyay S (2005) Satellite image classification using genetically guided fuzzy clustering with spatial information. Int J Remote Sens 26(3):579–593
6. Bandyopadhyay S (2005) Simulated annealing using reversible jump Markov chain Monte Carlo algorithm for fuzzy clustering. IEEE Trans Knowl Data Eng 17(4):479–490
7. Bandyopadhyay S, Maulik U (2001) Non-parametric genetic clustering: comparison of validity indices. IEEE Trans Syst Man Cybern C 31(1):120–125
8. Bandyopadhyay S, Maulik U (2002) An evolutionary technique based on k-means algorithm for optimal clustering in R^N. Inf Sci 146:221–237
9. Bandyopadhyay S, Pal SK (2001) Pixel classification using variable string genetic algorithms with chromosome differentiation. IEEE Trans Geosci Remote Sens 39(2):303–308
10. Bandyopadhyay S, Pal SK (2007) Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence. Springer, Hiedelberg
11. Bandyopadhyay S, Murthy CA, Pal SK (1995) Pattern classification using genetic algorithms. Pattern Recognit Lett 16:801–808
12. Bandyopadhyay S, Pal SK, Maulik U (1998) Incorporating chromosome differentiation in genetic algorithms. Inf Sci 104(3/4):293–319
13. Bandyopadhyay S, Murthy CA, Pal SK (2000) *VGA-classifier*: design and application. IEEE Trans Syst Man Cybern B 30(6):890–895
14. Bandyopadhyay S, Maulik U, Pakhira MK (2001) Clustering using simulated annealing with probabilistic redistribution. Int J Pattern Recognit Artif Intell 15(2):269–285
15. Bandyopadhyay S, Maulik U, Holder L, Cook DJ (eds) (2005) Advanced methods for knowledge discovery from complex data. Springer, Berlin/Heidelberg/New York
16. Bandyopadhyay S, Giannella C, Maulik U, Kargupta H, Liu K, Datta S (2006) Clustering distributed data streams in peer-to-peer environments. Inf Sci 176(14):1952–1985
17. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum, New York
18. Chiang DA, Chow LR, Wang YF (2000) Mining time series data by a fuzzy linguistic summary system. Fuzzy Sets Syst 112:419–432
19. Cios KJ, WPedrycz, Swiniarski RW, Kurgan LA (2007) Data mining: a knowledge discovery approach. Springer, New York
20. Cover TM (1965) Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Trans Electron Comput 14:8326–334
21. Dayhoff JE (1990) Neural network architectures an introduction. Van Nostrand Reinhold, New York
22. Devijver PA, Kittler J (1982) Pattern recognition: a statistical approach. Prentice-Hall, London
23. Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York
24. Ester M, Kriegel H-P, Xu X (1995) Knowledge discovery in large spatial databases: focusing techniques for efficient class identification. In: Advances in spatial databases. Springer, Berlin/Heidelberg
25. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd international conference on knowledge discovery and data mining, vol 96. AAAI Press, Portland, pp 226–231
26. Flockhart IW (1995) GA-MINER: parallel data mining with hierarchical genetic algorithms – final report. Technical report EPCC-AIKMS-GA-MINER-REPORT 1.0, University of Edinburgh, Edinburgh. citeseer.ist.psu.edu/flockhart95gaminer.html
27. Flockhart IW, Radcliffe NJ (1996) A genetic algorithm-based approach to data mining. In: Simoudis E, Han JW, Fayyad U (eds) Proceedings of the second international conference on knowledge discovery and data mining (KDD-96). AAAI Press, Portland, pp 299–302. citeseer.nj.nec.com/44319.html
28. Frigui H, Krishnapuram R (1999) A robust competitive clustering algorithm with application in computer vision. IEEE Trans Pattern Anal Mach Intell 21(1):450–465

29. Fukunaga K (1972) Introduction to statistical pattern recognition. Academic, New York
30. Han J, Kamber M (2000) Data mining: concepts and techniques. Morgan Kaufmann Publishers, San Francisco
31. Hartigan JA (1975) Clustering algorithms. Wiley, New York
32. Haykin S (1994) Neural networks, a comprehensive foundation. Macmillan College Publishing Company, New York
33. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
34. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall, Englewood Cliffs
35. Kargupta H, Chan P (eds) (1999) Advances in distributed and parallel knowledge discovery. MIT/AAAI Press, Cambridge
36. Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York
37. Kohonen T (2001) Self-organization and associative memory. Springer, Heidelberg
38. Krishnapuram R, Joshi A, Nasraoui O, Yi L (2001) Low complexity fuzzy relational clustering algorithms for web mining. IEEE Trans Fuzzy Syst 9:595–607
39. Lippmann RP (1987) An introduction to computing with neural nets. IEEE ASSP Mag 4(2):4–22
40. Lopes C et al (1999) Rule-evolver: an evolutionary approach for data mining. In: New directions in rough sets, data mining, and granular-soft computing. Springer, Berlin/Heidelberg, pp 458–462
41. Maulik U, Bandyopadhyay S (2000) Genetic algorithm based clustering technique. Pattern Recognit 33:1455–1465
42. Maulik U, Bandyopadhyay S (2002) Performance evaluation of some clustering algorithms and validity indices. IEEE Trans Pattern Analy Mach Intell 24(12):1650–1654
43. Maulik U, Bandyopadhyay S (2003) Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. IEEE Trans Geosci Remote Sens 41(5):1075– 1081
44. Ng R, Han J (1994) Efficient and effective clustering method for spatial data mining. In: Proceedings of the 1994 international conference on very large data bases, Santiago, Chile, pp 144–155
45. Oliver A, Monmarché N, Venturini G (2002) Interactive design of web sites with a genetic algorithm. In: Proceedings of the IADIS international conference www/internet, Lisbon
46. Oliver A, Regragui O, Monmarché N, Venturini G (2002) Genetic and interactive optimization of web sites. In: Eleventh international world wide web conference, Honolulu, Hawaii
47. Pakhira MK, Bandyopadhyay S, Maulik U (2004) Validity index for crisp and fuzzy clusters. Pattern Recognit 37(3):487–501
48. Pal SK, Bandyopadhyay S, Murthy CA (1998) Genetic algorithms for generation of class boundaries. IEEE Trans Syst Man Cybern 28(6):816–828
49. Pao YH (1989) Adaptive pattern recognition and neural networks. Addison-Wesley, New York
50. Pedrycz W (1998) Fuzzy set technology in knowledge discovery. Fuzzy Sets Syst 98:279–290
51. Picarougne F, Fruchet C, Oliver A, Monmarché N, Venturini G (2002) Web searching considered as a genetic optimization problem. In: Local search two day workshop, London, UK
52. Picarougne F, Monmarché N, Oliver A, Venturini G (2002) Web mining with a genetic algorithm. In: Eleventh international world wide web conference, Honolulu, Hawaii
53. Raymer ML, Punch WF, Goodman ED, Kuhn LA (1996) Genetic programming for improved data mining: an application to the biochemistry of protein interactions. In: Proceedings of first annual conference on genetic programming, MIT Press, Cambridge, pp 375–380
54. Selim SZ, Ismail MA (1984) K-means type algorithms: a generalized convergence theorem and characterization of local optimality. IEEE Trans Pattern Anal Mach Intell 6:81–87
55. Spath H (1989) Cluster analysis algorithms. Ellis Horwood, Chichester
56. Tou JT, Gonzalez RC (1974) Pattern recognition principles. Addison-Wesley, Reading

57. Wei Q, Chen G (1999) Mining generalized association rules with fuzzy taxonomic structures. In: Proceedings of NAFIPS 99, IEEE Press, New York, pp 477–481
58. Xie XL, Beni G (1991) A validity measure for fuzzy clustering. IEEE Trans Pattern Analy Mach Intell 13:841–847
59. Yager RR (1996) Database discovery using fuzzy sets. Int J Intell Syst 11:691–712
60. Zadeh LA (1965) Fuzzy sets. Inf Control 8:338–353
61. Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD international conference on management of data. ACM Press, New York, pp 103–114. http://doi.acm.org/10.1145/233269.233324