Chapter 17
Data Mining

Abstract Data mining, or knowledge discovery in databases, provides the tools to sift through the vast data stores to find the trends, patterns, and correlations that can guide strategic decision-making. The chapter highlights the major applications of data mining, their perspectives, goals of data mining, evolution of data mining algorithms—for transaction data, data streams, graph, and text-based data—and classes of data mining algorithms—prediction methods, clustering, and association rules. This is followed with cluster analysis, components of clustering task, pattern representation and feature extraction, similarity measures, and partitional algorithms. Data classification methods like decision trees and association rule mining are presented with worked examples. Sequential pattern mining algorithms are presented with typical pattern mining and worked examples. The chapter concludes with scientific applications of data mining, chapter summary, and list of practice exercises.

Keywords Data mining · Knowledge discovery · Goals of data mining · Data mining algorithms · Transaction data · Data streams · Graph data · Prediction methods · Clustering · Association rules · Rule mining · Cluster analysis · Pattern representation · Feature extraction · Similarity measures · Partitioned algorithms · Scientific applications

17.1 Introduction

As Information Technology (IT) has progressed, there has been an abundant increase in volumes of collected data in the recent past from all sorts of varieties. It is, therefore, beyond the capabilities of humans to extract meaningful information from this vast amount of data, and it has become necessary to develop algorithms which can extract meaningful information from these vast stores of data. Searching for useful chunks of information in the huge amounts of data is known as the field of *data mining*. Data mining can be applied to all varieties of formats of data, including relational, transaction, spatial databases, as well as large stores of unstructured data such as the World Wide Web.

K. R. Chowdhary, *Fundamentals of Artificial Intelligence*,
https://doi.org/10.1007/978-81-322-3972-7_17

The amount of data stored in digital form worldwide has on the average doubled every 9 months, over many years, which is twice the rate for increase of computing power, predicted by Moore's law. This doubling of stored information, called *storage law*, is one of the reasons of motivation for data mining. Irrespective of whether this increasing volume of data will support exploration in commercial or scientific activity, the data is potentially a valuable information [1].

It took many established organizations to accumulate large volumes of data about their employees, suppliers, customers, products, and services. To perform the data mining, also known as *knowledge discovery in databases*, organizations make use of tools to sift through this vast amount of data to find the trends, patterns, and correlations among the data sets, which can guide for strategic decision-making.

The traditional algorithms for data analysis assumed the input data sets of limited number of records, hence the available memory was sufficient. Current databases, however, are much larger to fit in the main memory of most computers. In addition, to be efficient, the techniques for data mining applied to very large databases must be highly scalable. An algorithm is considered as scalable if, given a fixed size of main memory, its runtime increases linearly with the number of records in the input database.

Data mining is concerned with the identification of patterns of important structures in data; these structures may represent patterns, statistical or predictive models of data, or some relationship among the parts of the data. In the context of data mining, the terms like patterns, models, and relationships have definite meanings as follows: a *pattern* is a compact summary of a subset of data (such as people who own a racer car are likely to participate in car races); a data *model* can be model of an entire data set, and it can be predictive. This model can be used to predict the future customers' behaviors (such as whether the customer may buy a so and so product), based on the historical data of interaction with some employees of the company. It can also be a joint probability distribution with reference to some variables.

An algorithm can enumerate a large number of data patterns using a finite database, but it is important to find out what data structure is suitable for a particular data set. Identifying interesting structures and useful data patterns among the large number of possibilities is the job of a good data mining algorithm, and it must do it fast even for large data sets. Consider sales transactions in a database of an online store, where some items' transactions are more frequent than others, e.g., smartphones. The variable values frequently occurring together in such databases could be used to answer, say, which items were bought together from the same store. Such an algorithm could also discover a pattern in the database in terms of demography, with very good confidence.

Data Mining is a Science

The concept of unsupervised learning—from basic facts or axioms—has remained a curiosity since long. In the present scenario, knowledge discovery engines are commonly used to extract general inferences from facts or using training data. Using more structured approach, the statistical methods attempt to quantify the vast amounts of

data by known and intuitively understood models. The approach of problem solution through assembling knowledge from existing data sources is a radical shift from traditional approaches of problem solution.

The nature of typical data sets, in respect of their size, diversity, high dimensionality, their distributed nature, and noisy contents, makes it challenging to decide the formal specifications to any problem. This lack of control gives scope for solutions that are over-fitting, have limited coverage with missing/incorrect data coupled with high dimensionality. Once such problem is specified, the solution techniques deal with the presentation and scalability of the problem, and complexity of the solution. This complete process through which data mining makes its transitions is a *science*, called *data science*.

Due to the emergence of WWW in the form of a large distributed data repository, and the provision of large online databases that can be tapped for significant commercial gain, interest of researchers and commercial organizations toward data mining techniques has gone up exponentially. Many core mining techniques have been explored by the researchers in the major domains, like classification, clustering, rule associations, and time-series analysis. The work on data mining has focused on scaling data mining algorithms to very large data sets. In the following sections of this chapter, we shall discuss the algorithms concerning three classical data mining problems: *market-basket analysis*, *clustering*, and *classification*.

Learning Outcomes of this Chapter:

1. Compare and contrast different uses of data mining as evidenced in both research and application. [Assessment]
2. Explain the value of finding associations in market-basket data. [Familiarity]
3. Characterize the kinds of patterns that can be discovered by association rule mining. [Assessment]
4. Describe how to extend a relational system to find patterns using association rules. [Familiarity]
5. Evaluate different methodologies for effective application of data mining. [Assessment]
6. Identify and characterize sources of noise and redundancy in presented data. [Assessment]

17.2 Perspectives of Data Mining

The data mining algorithms employ a variety of models to characterize or evaluate patterns. These models are from the areas of statistics, machine learning, databases, and experimental algorithms. In addition, there are mathematical approaches that used such approximation, and technical approaches such as dynamical systems. Following are the important perspectives used in the mining of data, which cover the most domain areas of research in data mining [2, 3].

Induction

It is the most common perspective, which is based on the principle of proceeding from specific to general, and answers the questions, like "Given ten specific examples of good tourist destinations, find out the characteristics of a favorite tourist attraction?" Thus, induction is typically implemented as a search through a space of possible hypotheses. Such searches usually employ some special characteristics or aspect to arrive at a good generalization, like "sand-dunes are favorite."

Compression

Many a time, one set of data may correspond to a number of general concepts. In such cases, mining techniques typically look for the most easily described pattern. This principle is called *Occam's Razor*, which effectively equates mining to compression, such that it becomes possible to describe the original data in a more compact form using learned patterns, rather than exhaustively enumerating original data itself requiring much larger size of space. The solid base to this issue are (1) feasibility of models such as Minimum Description Length (MDL) principle and (2) computational learning theory. Most data mining systems make use of one of these views of compression to establish the effectiveness of mining patterns. For example, if set of a data is of size ten, and the number of patterns mined turns out to be 20 features long, then this mining is not good at compression.

Definition 17.1 (*Pattern*) A pattern is a single data item of d dimensions, used by a clustering algorithm. It is also called observation, or datum, or feature vector. It is represented by $\mathbf{x} = (x_1, \ldots, x_d)$. □

Definition 17.2 (*Feature/attribute*) Each of the scalar components x_i of a pattern $\mathbf{x}$ are called features (or attributes) of the pattern. □

Querying

The perspective of query comes from the databases, because most business data reside in industrial and commercial databases and warehouses. Commercial database experts take data mining as a form of query. For mining the data (which often is in the form of text), it required to often enhance the expressiveness of the query, like "find all the customers with similar transactions." The other perspective of querying is to find out suitable model for database, in place of relational, for data mining.

Approximation

This view of mining has an objective to find an accurate model of data, and introduce some deliberate approximation in it to find out some hidden structures in the data. One technique that has been found useful for this is *latent semantic indexing*, which is useful in *document retrieval*. This technique makes use of transformations based on linear algebra and approximations of matrices to locate hidden structures of word usages, which means doing the searches beyond the simple keyword search.

Search

The search is related to *induction*, but its focus is on efficiency, and uses forward-pruning patterns, e.g., frequent itemsets, to restrict the overall pattern space.

Beyond what has been discussed above, there are many other approaches to classify the task of data mining, which fall in various categories:

- based on the *data* they operate on, e.g., discrete data, labeled data, continuous data, and time-series data;
- based on *application* domains, e.g., finance models, economic models, web-log mining, and semi-structured models;
- based on their *induced representations*, e.g., association rules, decision trees, and correlations.

17.3 Goals of Data Mining

The field of data mining aims to explore very large data sets efficiently, using methods that are convenient, easy, and practical. However, this should be without extensive training as well without a large work force. All the data mining applications have some common goals, of identifying the patterns in the data, interpretation of these patterns, and then perform the prediction or description either qualitatively or quantitatively in general for all the data including those which may be generated in the near future. Following are the major goals as well as the nature of applications of data mining.

Scaling analysis to large databases

The meaning of scalability is capability to handle large volumes of data which cannot fit in memory of any computer. The objective is to abstract away the patterns from the large databases, which provide information to mining algorithms to search for the patterns.

Scaling to higher dimensional data and models

The normal statistical data analysis is a two-step process—at the first step we formulate some model, then use the data to fit to this model. However, for human beings, formulation of a hypothesis that results in a model is not possible when the data set has a very large number of variables, of the order of thousands, involving various demographics, text document analysis, retail transactions, and web browsing. Through automated discovery from such a large volume of data, a model can be derived, and can be used in lower dimensional spaces, as the problem can be understood much easier at that level.

Automating search

The search requires enumeration of large data sets, creating hypotheses—the jobs beyond human capacity. However, these are done by algorithms meant for this purpose.

Finding pattern and models understandable

The most classical methods score on the models based on accuracy, i.e., how well the model is useful to predicts the data, and on utility, i.e., on the magnitude of benefits due to derived pattern. In addition to these, there are new measures in data mining, like how well the model can be understood, on the novelty of pattern discovered by it, and on how to further simplify the model.

17.4 Evolution of Data Mining Algorithms

Data mining is a data-driven field; here mining is concerned with real-world data sets. In traditional data analysis, popular data sets are d-dimensional vectors of $\mathbf{x}$ measurements on N number of objects, or N such objects having d number of measurements (or attributes). These data are called multivariate data, and are represented as an $N \times d$ matrix of data [4].

Some of the classical data mining and analysis problems associated with multivariate data are the following:

- *Clustering*: It is the process of learning a function, which can map $\mathbf{x}$ into a set of categories, such that the categories are not known in advance.
- *Classification*: It is the process of learning a function (i.e., learning a mapping) from $\mathbf{x}$ to y, where y is a categorical, or already defined scalar target variable of interest.
- *Density estimation*: It is estimating the probability density function (PDF), for vector $\mathbf{x}$, i.e., $p(\mathbf{x})$.
- *Regression*: The regression is the same as classification, except that table y takes real values.

The dimension d of vectors $\mathbf{x}$ is significant in multivariate modeling. For example in applications of text classification, and in clustering of gene expression data, d can be as large as 10^4. As per the density estimation theory, the amount of data needed to reliably estimate a density function grows exponentially in d. However, many predictive problems, like regression and classification, do not require a full d-dimensional estimate of the PDF $p(\mathbf{x})$, and instead rely on a simpler problem of determining a conditional probability function $p(y|x)$, where y is variable whose value is required to be predicted.

The first tools used to model multivariate data are old modeling methods from the domain of statistics and machine learning, e.g., logistic regression, linear regression, discriminant analysis, and N*ä*ive Bayes. The new predictive models used in this field are additive regression, decision trees, neural networks, as tools for more complex data models. They are more flexible, however they sacrifice the interpreting ability.

17.4.1 Transactions Data

A common form of data to be mined in most business contexts is records of individual "transactions" conducted by some individuals. Some of the common examples of these transactions are as follows: while doing purchase of groceries in a store each record is called a "basket", and in case of an individual's surfing a website each record is a description of a page requested in the session. Applying a multivariate concept we can view each of these record sets as sparse $N \times d$ matrix, where each of the N rows in this matrix corresponds to an individual basket or session, and each of the d columns corresponds to a particular item, and an entry $(i, j) \in N \times d$ is *true* if item j was purchased by a customer, or it was requested as part of the session i by the surfer, and it is *false* otherwise.

In actual cases, the parameters N and d can be very large. For example, a large retail chain or online store like Amazon or Flipkart might record on the order of a million baskets (i.e., N) per week and may have 10^5 different items in its store available for purchasing or downloading. That means 10^{11} entries of (i, j) cells. These numbers are a challenge for the system to be computationally tractable and suitable for statistical modeling. Considering these numbers for a store, we need to compute a pairwise correlation matrix taking time of the order of $O(Nd^2)$ and memory of $O(d^2)$, which in numeric vales are 10^{16} and 10^{10}, respectively.

However, we note that the $N \times d$ matrix is sparse. Considering each of the N customers buy only $d = 10$ items—the size of grocery basket—thus, only $10/10^5$ entries or 0.01% of this matrix are nonzero. Hence, our objective is to use a small subset of data, of the basket, called *itemset* I as information nugget (IN). An example of itemset is a combination of items as "bread, butter, and porridge" in a basket of grocery store.

There are many algorithms that can bring out all the *frequent itemsets* from a sparse set of transaction data. A specific itemset I ($I \in d$) is called *frequent* if for I, the relation $f_i > T$ holds, where f_i is *frequency* (rows count in which all the items in I were purchased), where T is some preselected threshold of rows ($T \in N$). Consider that $f_i = 10$, and some preselected threshold is $T \times N = 0.00005 \times 10^5 = 5$. Obviously, the itemset I is a frequent itemset.

Other approach to data mining algorithms takes a statistical view of basket data, as a density estimation problem instead of a search problem. This has a requirement of an approach to find statistically significant itemset, i.e., an itemset I whose *empirical frequency* deviates significantly from some baseline frequency. For example, a Bayesian conditional probability-based approach can discover complex multi-items' association, which has been ignored by others approaches.

We discussed that the transaction data is in the form of a sparse matrix $N \times d$. This form is in fact not a true picture of data—the real transactional data has significant additional finer structures. This structure, in the case of retail items is that they are arranged in some order, e.g., in the order of product hierarchies, and in the case of the web pages, the order is, they are usually related to each other through hyperlinks. Thus, columns of the sparse matrix of products as well of the web pages can have

themselves further more attributes, like "price" for itemsets and "contents" for web pages, as well as inter-item relationships, like laptop and its cover, web page which is the home page, etc. In addition, the rows of the transactional data (i.e., basket) can have further attributes like purchased as general, weekends, monthly, or seasonally.

17.4.2 Data Streams

The transaction data, instead of all available at one time, are in fact arriving continuously as a stream. As they flow by, they are available for mining only once. Similarly, the web logs continue to grow as browsing continues, over time, resulting in a stream of data. In such situations, the data miner's interest is also to study the evolution of the activity. The data streams give challenge, as to how to compute the aggregate of transaction data. One possible approach is to use an incremental learning model, such as classification tree. The further sections in this text discuss in more detail about data streams.

17.4.3 Representation of Text-Based Data

The text-based data can be represented in the form of a graph. The N objects can be represented as nodes in a graph, and edges can represent relationships among the objects. Such "data graphs" can be used for representation of web pages, where web pages are nodes and hyperlinks are edges of the graph. These graphs can be represented by adjacency matrix, where nodes are labels for row and columns and edges are entries in the cells of adjacency matrices. These matrices are however, large and sparse. Like in graphs, some nodes have extremely high degree—outgoing or incoming edges—while others may have degree of one only. If nodes of these are sorted accordingly to their degrees, the result is a rule of the form,

$$degree \ \alpha \ \frac{1}{rank^a} \tag{17.1}$$

where a is called as "degree" of the exponent.

Use of matrices for the representation of graph has benefits that many classical methods in linear algebra can make use of graphs for analyzing the properties of the corresponding problems. For example, to discover from the connectivity information, Google *PageRank* algorithm makes use of a recursive system of equations, that define the importance of any page in terms of the importance of the pages pointing to it, as well as how many such pages are pointing to it. The page rank of each page then can be computed by solving these linear equations.

17.5 Classes of Data Mining Algorithms

The process of data mining is increasingly being recognized as a key solution to analyzing, digesting, and understanding the huge volume of digital data generated through businesses, government activities, and through scientific research. Achieving this solution requires the *scaling* of mining algorithms to very large data sets. Majority of the traditional database algorithms access the databases multiple times or sometimes access these randomly. These are not possible when databases are very large. At the same time we need to speed up the computation. The approaches used for designing algorithms for handling such large databases are *prediction methods*, *clustering*, and *association rules* [5].

17.5.1 Prediction Methods

The predictive modeling algorithms use data sets of training records as their input. The goal of predictive modeling is to build a model that predicts some specified attribute(s) value from the values of the other attributes. The predictive methods are based on one or more of the following: Bayesian probability-based methods, decision trees, neural networks, and support vector machines. We elaborate the two methods here. The Bayesian approach and neural nets have been discussed in the previous chapters.

Linear Classifiers

A model after the training procedure can be a rule set or the whole training set like the nearest neighbor. In the following we will see that a straight line can be a model as well. In an example, it has been found that the height and weight of persons are available and medical experts have found that some of them are over-weight or under-weight (see table 17.1). Accordingly, the data have been divided into two classes, and are labeled as shown in Fig. 17.1. Consultation with experts may be expensive, so we would like to construct a model from the available data. Then for any new person, given the weight and height, this model could easily predict whether he/she is over- or under-weight.

A model can be a rule like *If weight* ≥ 60, *then over-weight.*

Table 17.1 A training set

ID	1	2	3	4	5	6
Weight (kg)	50	60	70	70	80	90
Height (m)	1.6	1.7	1.9	1.5	1.7	1.6
Over-weighted	No	No	No	Yes	Yes	Yes

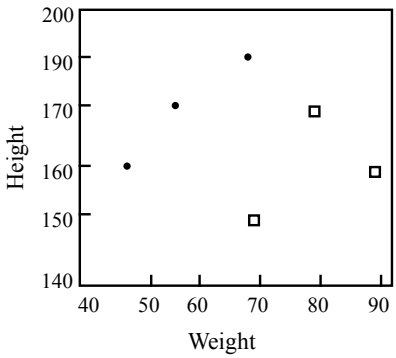

Fig. 17.1 A training set

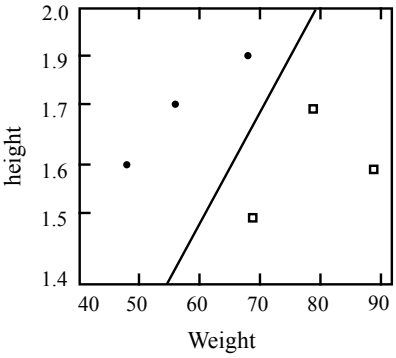

Fig. 17.2 A linear classifier

We note that this rule does not make much sense, because some tall people may be thin even though their weights are more than 60. A better model may be the following rule:

If weight/(height)$^2 \geq 23$, *then over-weight.*

The objective of classification is to identify a good model so that future predictions are accurate. Here "weight" and "height" are called *features* or *attributes*. In statistics, they are called variables. Each person is considered as a data instance (or a data observation). Mathematically, we have $\mathbf{x} = [weight, height]$ as a data instance and $y = 1$ or -1 as the label of each instance. Here, we have six training instances $x_1, \ldots, x_6$ with corresponding class labels as $y = [-1, -1, -1, 1, 1, 1]^T$ (here -1 and 1 mean under-weight and over-weight, respectively).

As discussed above, we now concentrate on the linear classifier, and show that a straight line can be a model also. Figure 17.2 shows that a line is separating the training data of over-weight and under-weight persons, and this line can be expressed by

$$0.2 \times weight - 10 \times height + 3 = 0. \tag{17.2}$$

In general, such a line can be expressed by

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{17.3}$$

where $\mathbf{x} = [weight, height]^T$, $\mathbf{w} = [0.2, -10]^T$, and $b = 3$. Then, for any new data $\mathbf{x}$, we check whether it is on the left or the right side of the line. That is,

$$\begin{aligned} if\ \mathbf{w}^T\mathbf{x} + b &> 0 \text{ predict } \mathbf{x} \text{ as "over-weight"}, \\ &< 0 \text{ predict } \mathbf{x} \text{ as "under-weight"}. \end{aligned}$$

Support Vector Machines

The support vector machines (SVMs) (see page 418, Chap. 14, for details) are based on simple principle of classification; they are powerful and popular approaches for predictive modeling. They are successful in a number of applications, such as face detection, handwriting recognition, text classification, and charmed quark detection.

SVMs are suitable for solving classification problems where attributes having two possible values 0 and 1 are to be predicted. The classification using SVMs is performed in a 2D space, where *predictor attributes* (i.e., 0, labeled with circles) separate from those with *dependent attribute* (i.e., 1, labeled with boxes) as shown in Fig. 17.3.

We can compute the optimal separating surface in SVMs by maximizing the *margin* of separation between predictor and dependent attributes. This margin is the distance between boundaries of the points of these two types of attributes (Fig. 17.3), and is important as it is a measure of safety (robustness) in separating the two sets of points, hence larger the margin, the better it is. As per the standard SVM formulation, computation of optimal separating surface requires solving a *quadratic optimization problem* [5].

Decision-Tree construction

Decision trees are specially attractive in data mining applications because any human analysis can easily comprehend the models of decision trees. Further, the construction of decision trees does not require any input parameters or prior knowledge about the data. We start at the root node of the tree, and repeatedly choose a child node based

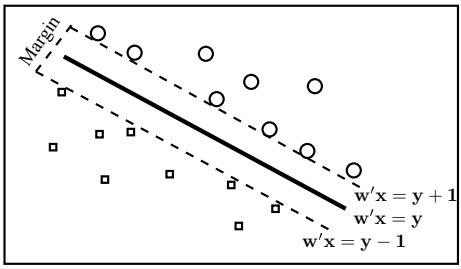

Fig. 17.3 Supported Vector Machine-based classification

on the *splitting criteria* that evaluates a condition on the input record at the current node. At the final node (leaf node), the process associates the record with the unique leaf node of the tree.

17.5.2 Clustering

The process of clustering partitions a set of data, according to some similarity measure, into several groups such that "similar" records are in the same group, so that each group represents a similar subpopulation in the data. As an example, each cluster could be a group of customers, which has similar purchase histories or interactions or some other factors or combinations (Sect. 17.6 discusses clustering in more details).

One technique to achieve scalability in clustering is to incrementally summarize the regions of the data such that denser regions are summarized first. Because a cluster corresponds to a denser region, the records within this region can be summarized collectively through a summarized representation called "cluster feature" (CF). An example of CF is a triple comprising of cluster centroid, cluster radius, and the number of points in the cluster.

The CF-based approach is considered efficient due to two reasons: (1) they consume lesser memory space as all the objects in a cluster are not required to be maintained, and (2) if properly designed, they constitute sufficient information for computing all intra-cluster and inter-cluster measurements required for making clustering decisions. In addition, the distances between clusters, CFs, and radii of clusters, and hence other properties of merged clusters, all can be computed quickly from the CFs of individual clusters.

Some points in clusters can be discarded, while the others can be compressed as defined below.

Definition 17.3 (*Discardable point*) A point is considered discardable if its membership can be ascertained with high confidence. □

Definition 17.4 (*Compressible point*) A point that is not discardable, but belongs to a tight subcluster consisting of a set of points that always move between clusters simultaneously, is called a compressible point. □

The CFs are also useful for scaling iterative clustering algorithms, like k-means algorithm, and while doing so it identifies three types of points (records): sets of discardable points, sets of compressible points, and a set of main-memory points. In a cluster, only the CF of all the discardable points are retained, and the actual points are discarded. Out of the remaining points, the compressible points are compressed, and those still remaining, since they are neither discardable nor compressible, are designated as main-memory records. The iterative cluster forming algorithm then moves only the main-memory points and the CFs of compressible points, between clusters until a criterion function is optimized, which concludes the formation of a cluster.

17.5.3 Association Rules

Under the association rules, we use a concept called *market basket*—a well-defined business activity—which is a collection of items purchased by a customer in an individual transaction. Such a transaction is possible due to a customer's purchase from, say, a grocery store, or an online purchase from a store such as Flipkart and Amazon.

By performing business activities over time, the retailers usually accumulate huge collection of transactions of performing a business activity. A common type of analysis performed on the collections of such transactions' database is to find sets of items, or *itemsets*, that appear together in many transactions. A pattern usually required to be extracted through this analysis consists of an itemset (market basket) and the corresponding number of transactions that contain it, with the objective that the business can use knowledge of these patterns to improve the placement of items in a store together, or can arrange the mail-order catalog pages, or web pages on a website, or use this criteria to motivate potential customers with attractive offers who can buy these itemsets.

The task of *association rule mining* is finding correlation between items in a data set. The initial research in rule mining was largely motivated by the analysis of market-basket data, the result of which allowed companies to better understand purchasing behaviors and, as a consequence, better market audiences [5].

17.6 Data Clustering and Cluster Analysis

Data mining and clustering are both exploratory activities, hence clustering methods are well suited for data mining. Clustering is often used as an important initial step in several data mining processes. The data mining approaches that use clustering methods are *predictive modeling, database segmentation*, and *visualization* of large databases [6].

Data mining is often carried out on relational databases, in cases of transactions that have well-defined structure and its columns can be used as features. However, it is also carried out on very large unstructured databases like WWW, where the contents are natural language text, mostly in HTML or XML format.

Clustering is in fact not a new field; it was being used in machine learning, statistics, and biology. However, scalability was not a design goal in these fields, and it was always assumed that complete a data set would fit in the main memory, and focus remained on improving clustering quality and not scalability. Due to this historical reason, majority of the clustering algorithms available today do not scale to large data sets. Clustering methods are used in data mining for segmenting the databases into homogeneous groups, that serve the purposes of data compression, because now we are working with the clusters and not with individual items. It helps to identify characteristics of subpopulations that are targeted for specific purposes

(e.g., marketing certain items aimed at specific section of the population). Clusters in large databases can be visualized, to help the analysts in identifying groups and subgroups that have similar characteristics.

By classification and grouping of objects based on common properties in some meaningful way, we are able to analyze and describe the world. We human beings are skilled enough at dividing the objects into groups (i.e., clusters) and can assign particular objects to these groups. For example, even young children can label the objects, as flowers, animals, trees, books, etc. Hence, clustering is an important criteria for understanding the objects as different classes. Every new object is identified based on whether it belongs to a so and so class or not. For example, if an object $x \in A$, where A is a class, say apple, then we say "x is an apple." Naturally, whether $x \in A$ or not depends on certain attributes of x.

The clustering process groups various data objects based only on information in the data items, which describes the objects and their relationships to objects in the same group. The goal of clustering is that the objects within a group be similar/related to one another, and different/unrelated to objects in other groups. The clustering will be better or more distinct, if there is greater similarity/ homogeneity within a group and greater difference between the groups.

The following example demonstrates a sample database as clusters.

Example 17.1 Clusters of customers' database based on three purchase behaviors: quantity, unit price, and their combination.

From the given data set, for each customer we compute the total number of items purchased and average price of all the items purchased. There are total 9 transactions as shown in Table 17.2, which are distinguished across three clusters. The clustering is based on common properties of items in each group/cluster, as follows: customers in cluster 1 purchased a few high-priced items, customers in cluster 2 purchased many high-priced items, and customers in cluster 3 purchased few low-priced items. □

In clustering, we organize a collection of patterns into groups based on their similarity. These patterns are usually represented as vectors of measurements, or

Table 17.2 Data groupings of similar objects

Cluster no.	<Qty, unit price>
Cluster 1	<2, 1800>
	<3, 2050>
	<5, 2270>
Cluster 2	<15, 1800>
	<18, 2200>
	<12, 2380>
Cluster 3	<3, 250>
	<4, 180>
	<4, 200>

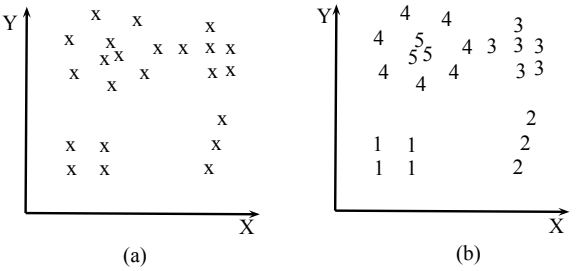

Fig. 17.4 Data clustering: **a** input patterns, **b** clusters formed

points in a multidimensional space. Intuitively, the patterns in the same cluster are more similar to each other than those in different clusters. Figure 17.4a, b depicts an example of clustering based on patterns, where input patterns are shown in (a), and the desired clusters formed are shown in part (b), where clusters belonging to the same cluster are shown by identical labels. Note that the measurement of the patterns in this case is (x, y) coordinate values.

One technique for clustering is called *supervised learning*, while other is called *unsupervised learning*. In supervised technique, a collection of labels, i.e., pre-classified patterns, are already provided, and the task is to label newly encountered unlabeled patterns. The already provided labels, called training patterns, are used to learn the descriptions of classes that in turn are used to label new unlabeled patterns. In the case of unsupervised learning, the task is to group a given collection of unlabeled patterns into some meaningful clusters. In fact, some kind of labels are there, associated with the clusters this time also, but this category of labels are data driven—obtained solely from the data, and not predefined.

17.6.1 Applications of Clustering

Clustering has applications in several fields, which require exploratory pattern-analysis, grouping, decision-making, and machine learning. Some applications, as examples, are given below.

Information Retrieval

The WWW comprises millions of pages of text, and a query to a web search engine, like “colleges” would return hundreds of pages having relevance to the query. However, these results can be grouped (as clusters) based on the categories like graduate programs, fees structures, intake, specializations, etc. Then each individual can be explored by the user.

Biology

Biologists are applying clustering techniques to analyze large volume of genetic information, for example, to find out the genes groups which have similar functions.

Business

In business and commerce, a large amount of information is collected on current and potential customers, then clustering is performed on this information to segment the customers into smaller number of groups so that additional analysis can be performed on each group, for example, to predict marketing potential.

Similarly, there are applications for image segmentation, pattern classification, and data mining, discussed in the following. In many such problems in such areas, there is little prior information (or statistical models) available about the data, and it is requirement that a decision-maker should make as few assumptions about the data as possible. It is under these conditions the clustering approach is useful for exploring the interrelationships among the data points to determine their structure.

17.6.2 General Utilities of Clustering

Clustering provides an abstraction from individual data objects to clusters comprising these data objects. Thus, each cluster is representative of a data item and can be called as a prototype for the data item. Having this, given a data item, it is possible to determine the closest representative cluster of that data item. Based on this deductive process, the following general *utilities* can be constructed using clustering techniques.

Summarization

Many data analysis methods, such as regression, have time complexity of $O(n^2)$, which makes it computationally difficult for large size of n, the number of objects. However, instead of applying the required algorithm to the entire data set, if it is applied on the representative clusters, the complexity will be far less, as it will be decided by the number of clusters, and not the data items.

Nearest neighbors

To find the nearest data item neighbor of a ith element, it needs to be compared with $n - i$ elements, and for all n items it is $O(n^2)$ complex. We know that if two data items are in two different clusters, then they cannot be nearer than the corresponding clusters. Thus, if n data items are in a set C clusters, then complexity to find the nearest neighbor is only $O(|C|)$, which may be far less than $O(n^2)$.

Compression

Consider using prototype for representing data items, such that each data item is represented by an index to its cluster. When similar data items are represented by a single cluster holding only one data item, it is effective compression. When this process is applied to sound, image, and video data, there is some loss of information, which is acceptable, however, there is substantial reduction in size.

Example 17.2 Consider a group of 12 sales records each indicating sales price, and have been sorted in ascending order as 5, 8, 11, 13, 15, 35, 45, 55, 72, 92, 201, and 215. It is required to partition these into three clusters.

Table 17.3 Simple clustering of data

Cluster 1	Cluster 2	Cluster 3
5, 8, 11, 13, 15	35, 45, 55, 72, 92	201, 215

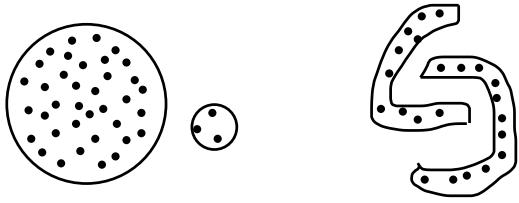

Fig. 17.5 Data sets on which centroids appear failed

(a) (b)

The partitions finally formed are shown in Table 17.3.

This has been obtained using a simple clustering technique that partitions the data along two largest gaps in the data sets. □

17.6.3 Traditional Clustering Methods

The partition-based technique, such as k-means partitioning, is based on optimizing a given criterion that attempts to break a data set into k-clusters. This approach assumes the cluster shapes as *hyper-ellipsoidal*, and the sizes of all clusters are assumed to be same. However, it cannot find clusters that vary in size, as shown in Fig. 17.5.

However, the Density-Based Spatial Clustering of Applications with Noise (DBscan) clustering technique can be used to construct clusters of arbitrary shapes and sizes. This method defines a cluster to be a *maximum-set* of density connected points—every core point in a cluster has got at least a minimum number of points within a given radius. We can reach to all these points in a cluster from any point in that cluster by traversing through a path of densely connected points, but the points across different cluster cannot. Finally, this technique can be applied only when density can be found out in advance, and it is uniform throughout the data set [6].

17.6.4 Clustering Process

A general pattern clustering process has the following steps:

1. pattern representation, which may also include feature selection and extraction,
2. defining proximity measures patterns specific to data domains,

3. grouping of patterns (clustering),
4. optionally, abstraction of data, and
5. optionally, assessment of output.

Figure 17.6 shows the first three steps from above, of a typical case of clustering. The feedback path indicates that the grouping process output could affect feature extraction and similarity computations in the next iteration.

The *pattern representation* depends on a number of criteria, these are available patterns, classes and their number, feature types, and their scale for clustering algorithm. All this information is not in the control of a programmer, hence *feature selection* process helps in identifying the most effective subset of the original features to use in clustering. Through *feature extraction*, one or more transformations of input features is carried out to produce new salient features. Either the selection or selection along with extraction can be used to obtain an appropriate set of features to use in clustering [7].

Definition 17.5 (*Pattern*) A pattern set is denoted by $\mathscr{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The ith pattern in $\mathscr{X}$ is denoted by $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$. In many cases a pattern set to be clustered is viewed as an $n \times d$ pattern matrix. □

Pattern proximity or closeness of one pattern to other is usually measured by a distance function defined on a pair of patterns. A simple distance measuring function is *Euclidean distance*, which is often used to reflect dissimilarity between two patterns—the more is the Euclidean distance between two patterns, the more is the dissimilarity between them. And, if the Euclidean distance is zero, the patterns are identical. Other similarity measures can be used to characterize the *conceptual similarity* between patterns.

The *grouping* step can be carried out in many ways. The output of clustering can be *crisp* or *fuzzy*. When clustering required at output is crisp (*hard*), the data is partitioned into groups, whereas when it is fuzzy partition, each pattern has a variable degree of membership of [0, 1], in each of the output clusters. In the other approach, the algorithms based on hierarchical clustering produce a nested series of partitions by merging or splitting of clusters based on similarity measures. In yet another way, partitioning is performed such that, algorithms identify a partition that optimizes a clustering criterion.

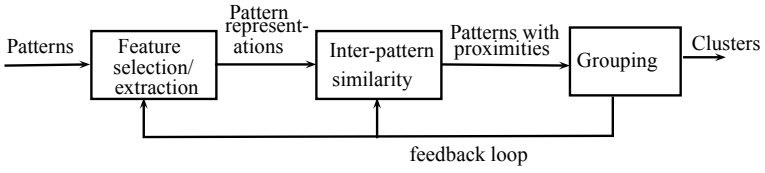

Fig. 17.6 Stages in clustering

Definition 17.6 (*Hard clustering*) It is a technique that assigns a class label l_i to each pattern $\mathbf{x}_i$, which identifies its class. The set of all labels for pattern set $\mathscr{X}$ is $\mathscr{L} = \{l_1, \ldots, l_n\}$, with $l_i = \{1, \ldots, k\}$, where k is the number of clusters. □

Definition 17.7 (*Fuzzy clustering*) It is a clustering procedure that assigns to each input pattern $\mathbf{x}_i$ a fractional degree of membership $f_{i,j}$ in each output cluster j, for all the k clusters. □

Data abstraction is aimed at extracting simple and compact representation of a data set. Here simplicity is from the point of view of automatic analysis, so that a machine can do the processing efficiently. Alternatively, the simplicity is due to being human oriented, such that the representation is easy to understand and intuitively convincing for humans. Usually, a data abstraction is a compact description of each cluster in the form of cluster prototypes or as representative patterns such as the centroid.

All clustering algorithms produce clusters when presented with data, irrespective of whether the data really contain clusters or not. It is not necessary that every set of data contains some clusters. For example, the continuous sequence 1, 2, . . . , 100, in no way represents a cluster. Only what we can have is all these numbers as clusters, each of size one. So, it is important, how to evaluate the output of a clustering algorithm? Apart from this, to evaluate the cluster quality, we want to know what characterizes a "good" clustering result and a "poor" one? If the data does contain clusters, some clustering algorithms may obtain "better" clusters than others.

Some of the criterias used to compare clustering algorithms are based on (1) the manner in which clusters are formed, (2) data structure of the cluster, and 3. how sensitive the clustering technique is to changes, which do not affect the data structure of the cluster.

17.6.5 Pattern Representation and Feature Extraction

A better quality of pattern representation will result in a clustering that is simple and easily understood; on the contrary, when it is a poor representation, it may result in a complex clustering, even some times its true structure may be difficult to recognize. For example, if a standard technique like Cartesian coordinates is used to represent the patterns, many clustering algorithms are likely to fragment the data into two or more clusters. But, if polar coordinate are used for representation of the clusters, the radius coordinate causes tight clustering and a one-cluster solution can be easily obtained.

A pattern can represent a physical object or an abstract notion. A physical object can be a chair, table, book, house, etc., while an abstract notion can be, e.g., a style of writing, attitude, belief, etc. Both of these can be represented in the form of multidimensional vectors, one dimension for a one feature. The features of a pattern can be either quantitative or qualitative. For example, if *weight* and *color* are the two features used, then $(black, 5)$ is the representation of a black object with 5 units of weight, for degree of blackness.

The features can be classified as quantitative and qualitative.

1. Quantitative features:
   a. discrete values;
   b. continuous values;
   c. interval values.
2. Qualitative features:
   a. nominal or unordered (e.g., color);
   b. ordinal (for temperature, e.g., cool or hot) or (for sound intensity, e.g., quiet or loud).

Other representations making use of structured features are represented as tree structures. In structured representations, a parent node represents a generalization of its child nodes. For example, a parent node "4-wheeler" could be a *generalization* of child nodes labeled as "cars", "jeep", and"tractor". Further, the node "cars" could be a generalization of car make, "Hundai", "Tata", "Maruti", etc. The generalized form of pattern representation, also called *symbolic objects*, are defined by logical conjunctions of events. These events link values and features, where features can take one or more values and all the objects are not required to be defined on the same set of features.

It is often beneficial to isolate only the most discriminative and descriptive features of the input set, and use these features exclusively in subsequent analysis.The isolation of features can be through selection or extraction. A feature *selection* technique identifies a subset of the existing features for subsequent use, while feature *extraction* technique computes new features from the original set to be used later. In both these cases, the objective is to obtain better classification or computation efficiency or at least one of these.

17.7 Clustering Algorithms

The selection of features in a pattern is an essential requirement in statistical pattern recognition. However, in the clustering context, where it lacks class labels, the feature selection is an ad hoc, but a necessity. As it lacks class labels, there can only be a trial-and-error process for the selection of features. The resultant patterns are clustered, and the output is evaluated using a *validity index*. There are some popular feature extraction processes, like principal components analysis, which do not depend on labeled data and can be used directly. The patterns having smaller number of features are beneficial, as the output can be visually inspected by a human.

For clustering the objects/patterns, the first requirement is to find out similarities between them, and more similar patterns are clubbed together to form clusters. In the following, we discuss how to measure the similarities between patterns and some standard algorithms for clustering.

17.7.1 Similarity Measures

Similarity is fundamental to the definition of a cluster. Hence, a measure of the similarity between any two patterns drawn from the same feature space is important for clustering procedures. Since there are many feature types and scales, the choice of distance or proximity measure must be done carefully. It is usually common to compute the dissimilarity (distance) between two patterns rather than the similarity [7].

Definition 17.8 (*Distance measure*) Given two objects O_1, O_2 from the possible universe of objects, the distance or dissimilarity between O_1 and O_2 is a real number denoted by $d(O_1, O_2)$. □

Consider that A, B, and C are three objects or patterns, the following properties hold for the distance measure for these objects:

$d(A, B) = d(B, A)$: by rule of *Symmetry*,
$d(A, B) = 0$, if and only if $A = B$: by *Constancy of self-similarity*,
$d(A, B) \geq 0$: by *Positivity*,
$d(A, B) \leq d(A, C) + d(C, B)$: by rule of *Triangular inequality*.

The dissimilarity between two patterns is defined on the feature space using the distance measure. Our focus shall be on distance metrics with continuous features. The popular metric for continuous features is the *Euclidean distance*, expressed by

$$\begin{aligned} d_2(\mathbf{x}_i, \mathbf{x}_j) &= (\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2)^{1/2} \\ &= \| \mathbf{x}_i - \mathbf{x}_j \|_2 . \end{aligned} \tag{17.4}$$

Equation (17.4) is a special case of the *Minkowski's metric*, where p was taken as 2, expressed by

$$\begin{aligned} d_p(\mathbf{x}_i, \mathbf{x}_j) &= (\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^p)^{1/p} \\ &= \| \mathbf{x}_i - \mathbf{x}_j \|_p . \end{aligned} \tag{17.5}$$

The approach based on Euclidean distance has an intuitive appeal, and the method is commonly used to evaluate proximity of objects in 2 and 3D spaces. The method works well when the data set comprises "isolated" or "compact" clusters. However, it has a drawback, as there is a tendency of large-scaled feature to dominate the others features. The solutions to this, is to incorporate normalization to the continuous features (with a common range or variance) or some other weighting schemes.

Example 17.3 Consider a set of 2D data points as shown in Table 17.4, and given a new data point, $x = (2.5, 2.9)$ as a query, rank these database points based on similarity with the query, using Euclidean distance.

Table 17.4 Original 2D data

	A_1	A_2
x_1	1.9	1.7
x_2	2.1	2.1
x_3	2.6	3.0
x_4	2.2	2.5
x_5	1.8	2.0

Table 17.5 Euclidean distances

Given data point	Euclidean distance with x
x_1	1.341
x_2	0.894
x_3	0.141
x_4	0.500
x_5	1.140

Using Eq. (17.4), we compute the Euclidean distance for the 2D data points x_1, …, x_5 with respect to the query $x = (2.5, 2.9)$. The result are shown in Table 17.5.

17.7.2 Nearest Neighbor Clustering

Since proximity between items plays an intuitive role in clustering, a method based on the *nearest neighbor* distances can be an obvious choice as a basis of clustering procedures.

An iterative algorithm assigns each unlabeled pattern to the cluster of its nearest labeled neighbor pattern, with the condition that the distance to that nearest pattern is below the given threshold.

This process continues until all the input patterns are labeled.

To grow the clusters from the nearest neighbor, a concept called *mutual neighbor distance*, call it MN_d, can be used, which is expressed as

$$MN_d(\mathbf{x}_i, \mathbf{x}_j) = C_n(\mathbf{x}_j, \mathbf{x}_i) + C_n(\mathbf{x}_i, \mathbf{x}_j). \tag{17.6}$$

In the above, $\mathbf{x}_i$, $\mathbf{x}_j$ are patterns, $C_n(\mathbf{x}_i, \mathbf{x}_j)$ represents the count of neighbor numbers of $\mathbf{x}_j$ with respect to $\mathbf{x}_i$. Figure 17.7 illustrates this concept. There are total six patterns, P, Q, R, S, T, $and U$. In part (a), nearest neighbor of pattern P is Q, and Q's nearest neighbor is P. Also, $C_n(P, Q) = C_n(Q, P) = 1$, hence $MN_d(P, Q) = 2$. If $C_n(Q, R) = 1$ but $C_n(R, Q) = 2$, then $MN_d(Q, R) = C_n(Q, R) + C_n(R, Q) = 3$.

Figure 17.7b is obtained from Fig. 17.7a by adding three new patterns S, T, $and U$. Now, $MN_d(Q, R) = 3$, but $MN_d(P, Q) = 5$. Note that MN_d between P and Q has increased from 2 to 5 due to additional three patterns S, T, $and U$, though the position of P and Q have not changed.

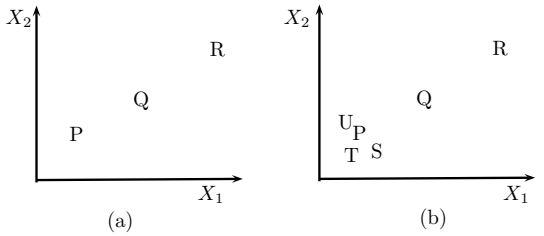

Fig. 17.7 Nearest neighbor clustering: **a** P and Q are more similar than P and R, **b** Q and R are more similar than Q and P

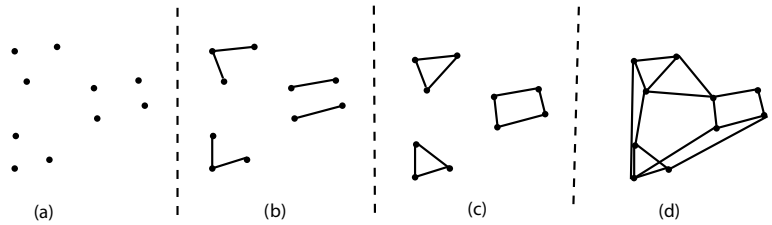

Fig. 17.8 Construction steps of k-nearest neighbor graph using original data: **a** original data, **b** 1- **c** 2- **d** 3-nearest neighbor graphs

A general case of the nearest neighbor algorithm is k-nearest neighbor algorithm. Figure 17.8 illustrates for $k = 1$, 2, and 3-nearest neighbor graphs for some simple data set.

There are many advantages of representing data items using a k-nearest neighbor groups (clusters). The far apart data items are completely disconnected in this approach, and since the data items are connected with the nearer items such that the weights on the edges in the graph are indicators of nearness, these weights are also indicators of population density in the data items' space. Since the items in sparser and denser regions are modeled uniformly, the sparsity of the representation results in algorithms that are computationally more efficient.

17.7.3 Partitional Algorithms

A partition-based clustering algorithm first obtains a single partition of the data, without any structure. As the next step, clusters are produced by optimization of a *criterion function* defined locally (over a subset of the patterns) or defined globally (on the entire set of the patterns). Searching for a set of possible labeling for an optimum value of a criterion is computationally expensive, and combinatorial in nature. To simplify this, the algorithm is typically run multiple times with different starting states, and the best configuration obtained from all of the runs is used as the output clustering.

The above discussed algorithm has a problem of providing a choice of the desired number of output clusters. However, it is computationally more efficient than hierarchical clustering, when used for a large data set.

Squared Error Algorithms

The squared error function approach is the most intuitive concept for partitional clustering, as it is ideally suited for compact and isolated clusters. For an input set of $\mathscr{X}$ patterns, the squared error for clustering $\mathscr{C}$, consisting K clusters $(C_1, \ldots, C_K)$, can be expressed as

$$e^2(\mathscr{X}, \mathscr{C}) = \sum_{j=1}^{K} \sum_{i=1}^{m_j} \| \mathbf{x}_i^{(j)} - \mathbf{c}_j \|^2 . \tag{17.7}$$

In Eq. (17.7), $\mathbf{c}_j$ is centroid of the jth cluster in total K clusters formed, m_j is the number of patterns in the jth cluster, and $\mathbf{x}_i^{(j)}$ is the ith pattern in the jth cluster.

Algorithm 17.1 Squared Error Clustering Algorithm

1: Select an initial partition **X** of patterns, with a fixed k number of clusters, and cluster centres
2: **repeat**
3: **for** each pattern $\mathbf{x}_i \in \mathbf{X}$ **do**
4: Find centroid $\mathbf{c}_j$ (of cluster C_j) having minimum distance with pattern $\mathbf{x}_i$
5: $C_j = C_j \cup \{\mathbf{x}_i\}$
6: Compute the new centroids (cluster centers) of all the clusters
7: **end for**
8: Merge and split clusters based on some heuristic criterion
9: **until** convergence is achieved
10: **end**

The steps of the squared error clustering algorithm are listed in Algorithm 17.1. The repetition in the *repeat …until* loop continues until the convergence is achieved, i.e., the cluster membership is stable.

k-means partitional clustering

The k-means is a partitional clustering technique that tries to find a k number of clusters, the count is specified by the user. These are represented by their centroids. It is the simplest and the most commonly used algorithm that uses the*squared error* criterion. The k-means algorithm starts with a random initial partition and keeps reassigning the patterns to clusters based on the similarity between the pattern and the cluster centers (centroid distances) until a convergence condition is reached. In the process of clustering, there is no reassignment of any pattern from one cluster to another, which gives it a property of linear time complexity. In other words, the squared error decreases to some minimum threshold after some number of iterations.

Following are the major advantages of k-means algorithm. (1) It is easy to implement, (2) its time complexity is $O(n)$, where n is the number of patterns. However, its

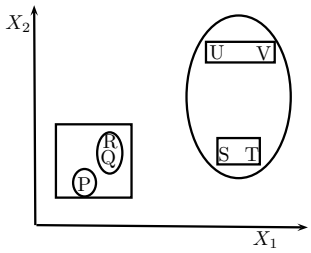

Fig. 17.9 The k-means clustering is sensitive to initial partition

disadvantage is that it is sensitive to selection of the initial partition—if not properly selected it may converge to a *local minima* of the criterion function value.

The following example demonstrates creation of clusters based on the k-means algorithm.

Example 17.4 Using the k-means approach to perform partitioning.

Figure 17.9 shows 2D patterns P, Q, R, S, T, U, $and V$. The process is started with initial patterns P, Q, and R. Around these, three ($k = 3$) clusters are to be constructed. We end up with the partition $\{\{P\}, \{Q, R\}, \{S, T, U, V\}\}$, where three clusters are shown by ellipses. The squared error criterion value turns out to be much larger for this partition. This will happen, for example, for the centroid versus the patterns in the largest ellipse. Hence, we construct a better partition $\{\{P, Q, R\}, \{S, T\}, \{U, V\}\}$, where clusters are shown by rectangles. This grouping results in the global minimum value of the squared error criterion function, for clustering comprising of $k = 3$ clusters. The correct three-cluster solution is obtained by choosing, for example, P, S, and U as the initial cluster means, which will form the partition as $\{\{P, Q, R\}, \{S, T\}, \{U, V\}\}$ [7]. □

17.8 Comparison of Clustering Techniques

A collection of all the clusters corresponding to a given input data set is generally called as clustering. In the following discussions, we shall distinguish between various types of clustering approaches such as partitional (i.e., un-nested) versus hierarchical (i.e., nested), monothetic versus polythetic, and hard versus fuzzy [7].

Partitional versus Hierarchical

At the top level, distinction is made between hierarchical and partitional approaches for clustering, where hierarchical-based methods produce a nested series of partitions. However, the partitional (un-nested) methods produce only one partition. In the hierarchical clustering, the features are used sequentially, whereas in the creation of partitional clustering they are used simultaneously.

The hierarchical clustering algorithms produce a sequence of clusters, which are nested in nature. They have a single all-inclusive cluster at the top, and single

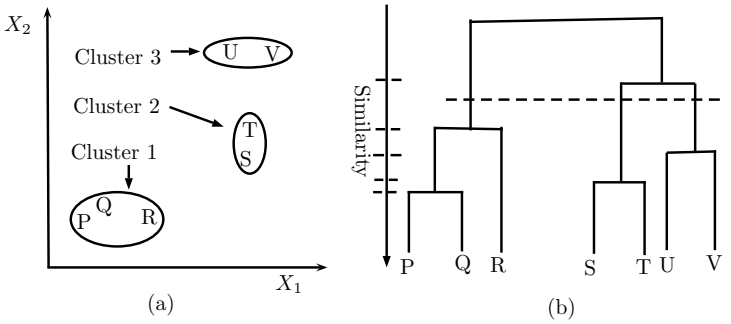

Fig. 17.10 **a** Data items in three clusters, **b** dendogram for clusters in (**a**)

point clusters at the bottom of the hierarchy. At the start, the algorithms, called *Agglomerative* hierarchical algorithms, treat each data point as a separate cluster. After this initial step, each further step merges two clusters that are most similar. For demonstration of this, see Fig. 17.10b, from bottom towards top. For total n number of data points at the beginning, each such step compares each node with each other node, hence the worst case analysis is $O(n^2)$. After each merger, the total number of clusters reduce by one. The user can repeat these steps until the desired number of clusters are obtained or the distance between two closet clusters goes above a certain threshold. Since in the worst case n number of data points reduce to one cluster in total $n - 1$ number of steps, the worst case time complexity of this algorithm is $O(n^3)$.

In some of the hierarchical methods, each cluster is represented by its *centroid*. A centroid of a cluster is a data point, which is closest to the centre of that cluster. In centroid-based methods, the distance between two clusters can be measured by how similar the centroids of these clusters are. However, the centroid-based scheme of distance calculation fails when some data points in a cluster are closer to the centroid of other clusters than the centroid of their own cluster.

The working of a hierarchical clustering algorithm is illustrated in Fig. 17.10a. It makes use of a 2D data set, in the form of seven patterns labeled as P, Q, R, S, T, U, and V. After merging these data sets, three clusters (1, 2, and 3) are obtained. A hierarchical algorithm results in a nested grouping of patterns, which produces a dendrogram, as illustrated in Fig. 17.10b. The similarity levels at which groupings change are also marked in Fig. 17.10b; any two clusters can also be merged to form a larger cluster based on the minimum distance criteria. In this example, clusters 2 and 3 can be merged to make a single bigger cluster.

A partitional clustering is created simply by dividing the set of input data objects into non-overlapping subsets at the output such that each data object is in exactly one subset, i.e., a cluster. A simple partitional (monothetic) clustering considers the features sequentially to divide the given collection of patterns (see Fig. 17.11).

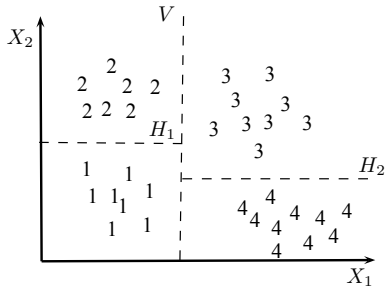

Fig. 17.11 Clustering based on monothetic partitioning

Monothetic versus Polythetic

This criteria is related to the sequential versus simultaneous use of features for performing the clustering. Most clustering algorithms are of the type *polythetic*, i.e., first, all the features enter into the algorithm before the computation of distances between patterns begin. Computation of distances, based on which clustering decisions are made, are based on all these features, and not any specific feature. However, a *monothetic* algorithm considers the features *sequentially* to divide the given collection of patterns into clusters.

Figure 17.11 illustrates the the construction of monothetic type of clusters. It shows that the collection is divided into two groups based on feature $x_{1,i} \in X_1$, which are separated by a broken vertical line V. These two clusters are independently further divided using the feature $x_{2,j} \in X_2$, indicated by broken lines H_1 and H_2. The major problem with this method is that, for a patterns' dimensionality of d, it generates 2^d clusters, which is exponential. For example, for even moderately large values of d, (say $d > 50$), which is typical in Information Retrieval (IR) applications, the number of clusters generated by this algorithm is 10^{15}—a division of data set into uninterestingly small and fragmented clusters.

A monothetic class is defined in terms of features, such that these features are both sufficient as well as necessary to identify the members of that class. For example, people are clustered in age groups 0–25, 26–50, and greater than 50 years old. And, of course there may be many other features existing for each individual in each cluster, but one essential feature is the range of age, which is strictly followed in each group.

A broad set of criteria that are neither the compulsory requirement, nor are sufficient, are used to define a polythetic class. A certain minimal number of defining characteristics must be possessed by each member of the category, but it is not necessary that some feature must compulsorily be found in each member of the category. The distance between the members defines the membership in a class.

Incremental versus non-incremental

Incremental algorithms are useful when the pattern set to be used is very large, and the constraints imposed on execution time, or memory space, or both of these affect the nature of the algorithm. The traditional clustering algorithms were not designed to work with large data sets, hence the feature of scalability was absent. However, this feature is very much in need in most present day applications of large data sets.

The field of data mining has led to the development of clustering techniques that minimize the number of scans through a pattern space, reduce the number of patterns examined during execution, or reduce the data structures' size in the algorithm. The main advantage of incremental algorithms is that for them it is not required to store the entire pattern matrix in the memory, to be used again and again. But, instead, only part of that is stored at a time, which helps in reducing the total requirement of space. These algorithms are generally not iterative, hence the execution time is also small.

Algorithm 17.2 illustrates a typical incremental clustering algorithm. Let D_0 be the first data item, and C_0 be the first cluster.

Algorithm 17.2 Incremental clustering Algorithm

1: $C_0 = \{D_0\}$
2: $d = 1$; data item counter
3: $k = 0$; cluster counter
4: **while** there is data-item available in input **do**
5: Pickup the next data item D_d
6: Compute distance of D_d from all existing clusters' centroids
7: Let m is minimum distance, from centroid of cluster C_m
8: **if** m greater than threshold **then**
9: $k = k + 1$; new cluster
10: Create new cluster C_k
11: $C_k = C_k \cup \{D_d\}$
12: **else**
13: $C_m = C_m \cup \{D_d\}$
14: **end if**
15: $d = d + 1$; next data-item
16: **end while**
17: **end**

Hard versus Fuzzy Clustering

A hard clustering algorithm assigns each input pattern to a single cluster in the output produced. However, a fuzzy clustering method may assign a degree of membership (belongingness) for each input pattern to several clusters in the output produced. A fuzzy clustering can be converted to a hard clustering by assigning each pattern to only one cluster, that which possesses the largest measure of membership of that pattern, and the partial membership of other clusters is ignored.

17.9 Classification

The basic idea of the data classification problem can be simply described as follows: given a training data with known labels or classes (e.g., as shown in Table 17.6), we would like to learn a model, so that it can be used to predict data with unknown labels. Let us consider that we have identified some customers through clustering

Table 17.6 Sample training database

Record ID	Employment	Age	Salary	Group
1	Self	30	30K	*C*
2	Industry	35	40K	*C*
3	Self	35	60K	*A*
4	Self	30	70K	*A*
5	Industry	35	40K	*C*
6	Academia	50	70K	*D*
7	Self	45	60K	*D*
8	Academia	30	70K	*B*
9	Industry	35	60K	*B*

of the aggregated purchase information about the currently existing customers for a certain company (see Table 17.2, page no. 520). Further, we have also acquired the mailing list of potential customers out of these, with their demographic information. As the next step, we would like to assign each person in the mailing list to one of the three groups: A , B, and C, as shown in Table 17.6. The latter is for the purpose of mailing them a catalog of items tailored to the individual's buying patterns. This task, data mining, makes use of historical information about current customers to help in the prediction of cluster membership of new customers.

Let us assume that the training database with historical information has records with attributes: ⟨salary, age, employment, and group⟩. The goal is to build a model that takes as input the *predictor attributes* and outputs a value for the *dependent attribute*. When the dependent attribute is a numerical value, the problem is called *regression*, otherwise it is a *classification* problem. In our present discussion the dependent attributes (also called *class labels*) are A, B, and C, hence it is a classification problem. The Table 17.6 is a sample training database with four predictor attributes: salary, age, and employment, and group as dependent attribute.

There are many classification models for data mining applications. These include the following: genetic algorithms, Bayesian networks, neural networks, log-linear methods, statistical methods, and decision tables. The classification trees (i.e., decision trees) are popular due to the following reasons:

- Their representation is intuitive, which makes classification model easy to understand,
- An analyst does not need to supply any input parameter for construction of decision trees,
- The accuracy of prediction of decision trees is better than other classification methods,
- It is possible to construct decision trees from very large training databases using algorithms that are scalable and fast.

Decision Trees

The decision trees are tree structures whose leaves are classifications and their branches are conjunctions of features that lead to classifications. Their significance is because many data mining methods generate decision trees. The problem solution methods learn these decision trees. One approach to learning a decision tree is to split the example set into subsets, based on the value test of some attribute. This process is repeated recursively on the subsets, with each split value becoming a subtree root. The splitting stops when the subset becomes so small that further splitting is not possible, i.e., the subset example contains only one classification. A split is considered best if it produces the minimum number of classification in the subset. That means, subsequent learning generates smaller subtrees, which will require less further splitting, in effect reducing the number of steps for solution of Problem [8].

A decision tree algorithm used for the classification comprises two steps: tree building and pruning. In the first step, most of the decision tree is grown top-down in a greedy way. Starting with the root node, the database is examined by a method, called "split selection", that selects the split condition at each node. Then, the database is partitioned and the procedure is applied recursively. In the pruning stage, the tree constructed in the previous phase is pruned to control its size. The pruning methods select the tree in a way that minimizes prediction errors. In some cases, decision-tree construction algorithms separate the process of tree building and pruning, but other algorithms interleave these processes to avoid unnecessary expansion of some nodes.

Algorithm 17.3 shows a sample tree building phase, node n where tree is to be split. At the output, the algorithm provides a decision tree for the data partition D, and new node value n which is the root of the decision tree.

Algorithm 17.3 Sample code for Tree building

```
% Input: Data-partition D, Node n, split selection criteria P
% Output: Decision tree for D, with it root as n
Build-Tree(n, D, P)
Apply P to D, and find splitting criteria for node n
if n splits then
    Create its child nodes n1, n2
    partition D into D1, D2 using split criteria
    Build-Tree(n1, D1, P)
    Build-tree(n2, D2, P)
end if
```

The choice of splitting criteria determines the equality of the decision tree. If the training database does not fit into memory, we need a scalable data access method. Many scalable algorithms are designed with a built-in feature, which ensures that only a small set of statistics, like aggregate measures and counts, be sufficient to implement the split selection method. Since aggregated data is far smaller in size than the actual data, it is possible to construct the statistics in memory for each node in a single scan over the corresponding database partition. These nodes satisfy the

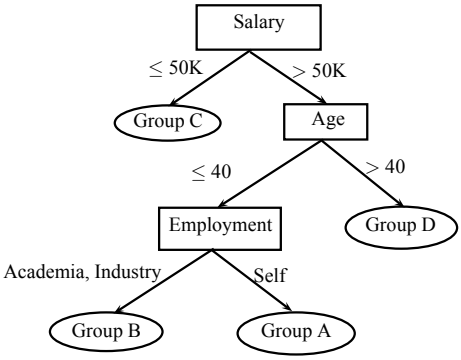

Fig. 17.12 Sample decision tree for catalog mailing

splitting criteria, such that in the process of recursively splitting we ultimately reach the class label node at leaf. In a decision tree, each internal node is labeled with a prediction attribute, called *splitting attribute*, and each leaf node is labeled with a *class label*.

Example 17.5 Figure 17.12 shows the decision tree for a training data set as shown in Table 17.6. The splitting attributes are salary, age, and employment, and the class labels are Groups A, B, C, *and* D.

Every edge that originates from an internal node is labeled with a *splitting predicate*, which involves only the node's splitting attribute. The splitting predicate in the decision tree (Fig. 17.12) are $\leq 50K$, $>50K$, ≤ 40, >40, *Self*, *Academia, and Industry*. The splitting predicate has a property that any record will take a unique path from the root to exactly one leaf node, that is the class label of that record. At a node, the combined information about splitting attributes and splitting predicates is called the *splitting criterion*. □

17.10 Association Rule Mining

Association rules are a set of significant correlations, frequent patterns, associations, or causal structures from data sets found in various types of databases. Such databases are transactional databases, relational databases, and other forms of data repositories. Mining of association rules is capturing those correlations, patterns, and rules and representing them in the form of some *if*…*then* rules. For example, given a set of transactions, each transaction comprising a set of items, an *association rule* can be an implication, $X \Rightarrow Y$, where X and Y are sets of items, indicating that presence of itemset X implies the itemset Y. Consider that an insurance company finds a strong correlation between two sets of policies X and Y of the form $X \Rightarrow Y$, which may be an indicator that customers holding policy set X were also likely to hold policy set Y, where X and Y may have one or more elements. Now the company

could more effectively target marketing the policy Y through those clients who hold policy X but not Y, to motivate them to buy policy Y. In effect, the association rule represents the knowledge about purchasing behavior of customers. The association rule mining has been effectively applied to many different domains that include *market-basket analysis* in commercial environments, astrophysics, crime prevention, fluid dynamics, and counter terrorism, i.e., all the areas in which a relationship between objects can be concluded as useful knowledge [5, 9].

Association rules analysis is a technique to uncover how items are associated with each other. There are three commonly used ways to measure the association, which indicate the strength of the association.

Definition 17.9 (*Support*) For a given set of data items, an association rule has *support s* for some set of items X if s percent of transactions include all the items of set X. □

In the sales transactions, for example, if we discover that sale of some items beyond a certain proportion tend to have a significant impact on the total profits, we might consider using that proportion as *support threshold*.

Definition 17.10 (*Confidence*) "Confidence" says how likely the itemset Y is purchased when itemset X is purchased, i.e., to determine the value of $X \rightarrow Y$. For a given set of data items, an association rule has *confidence c* if c percent of transactions that contain itemset X also contain itemset Y. □

"Confidence" is measured by the proportion of transactions with item X, in which Y also appears. For the total number of transactions T, we compute c as

$$\begin{aligned} c &= \frac{(X, Y)}{X} \\ &= \frac{(X, Y)/T}{X/T} \\ &= \frac{support(X, Y)}{support(X)}. \end{aligned}$$

In association rule mining, usually the goal is to discover all association rules having the value of support and confidence greater than some user-specified minimum threshold value [8].

Definition 17.11 (*Lift*) "Lift" (l) says how much it is likely that the itemset Y will be purchased whenever itemset X is purchased, while controlling "lift" for how popular the item Y is.

The lift is measured as

$$l = \frac{support(X, Y)}{support(X) \times support(Y)}. \tag{17.8}$$

A lift value greater than 1 means item Y is likely to be bought if X is bought, while its value less than 1 means Y is unlikely to be bought if X is bought.

Table 17.7 Transactional database

Tran. ID	Cust. ID	Item name	Price (in $)	Date
101	201	*Laptop*	1500	8/20/2018
101	201	*Tablet*	300	8/20/2018
101	201	*Smartphone*	100	8/20/2018
102	201	*Music system*	500	8/25/2018
102	201	*Smartphone*	100	8/25/2018
103	202	*Laptop*	1500	8/30/2018
103	202	*Music system*	500	8/30/2018
103	202	*Smartphone*	100	8/30/2018

Example 17.6 Given the sales transactions in Table 17.7, find out the "support" for the following:

1. Laptop,
2. Smartphone, and
3. Laptop and Smartphone

and "confidence" of a music system with respect to

1. Laptop,
2. Smartphone, and
3. Laptop and Smartphone.

There are three transactions in this table: numbers 101, 102, and 103, stored in a relational database system. We note that out of the three transactions, two have Laptop, three have Smartphone, and two have Laptop and Smartphone combined. Accordingly, their "support" is 67, 100, and 67 percent, respectively.

In the second part, we are interested to compute confidence of "Laptop" → "Music system", i.e.,

$$\begin{aligned} c &= \frac{support(laptop,\ music\ system)}{support(laptop)} \\ &= \frac{1/3}{2/3} \\ &= 50\%. \end{aligned}$$

The "lift" for the above is computed as follows.

$$\begin{aligned} c &= \frac{support(laptop,\ music\ system)}{support(laptop) \times support(music\ system)} \\ &= \frac{1/3}{2/3 \times 2/3} \\ &= 0.75. \end{aligned}$$

The *lift* of 0.75 (< 1) indicates that item Y is unlikely to be bought by a customer who is buying item X. □

Most of the algorithms for association rules mining comprise two distinct steps:

1. *Find all sets of items with minimum support.* The data are of the order of millions of transactions, and a mining algorithm needs to count a large number of candidate itemsets to identify the frequent ones; hence this phase is computationally expensive, and often time consuming.
2. *Generation of inferences.* The inferences or rules can be generated directly from the frequent itemsets, and there is no need to scan the data sets again.

In the above two steps, most of the time is usually consumed by the first step, hence the scalability becomes more important. The techniques for scalability can be divided into two groups: 1) those techniques which reduce the total number of candidates to be counted; and 2) those which make candidates' counting more efficient. The first category of techniques make use of the property—subsets of a frequent itemset must also be frequent, and uses this concept as a pruning technique to reduce the number of itemsets to be counted.

Variations of actual problem

The other approaches used in the data mining algorithms are focused on variations of the actual problem. For example, for data sets and *support* values where the frequent itemsets are very long—with n items there are 2^n possible frequent subsets—this makes finding all frequent itemsets an intractable problem. However, by looking ahead, the set of maximal frequent itemsets can still be found efficiently. We can make use of the following criteria for this: if an itemset is identified as infrequent, its subsets are also infrequent, and hence none of its subsets need to be counted. A key approach to this solution is to maximize the probability that itemsets counted due to looking ahead are actually frequent, which requires estimation of some heuristics. For this, a good heuristic can be to bias candidates' generation such that most frequent items appear in most of the candidate groups. This heuristic will make it more likely for high-frequency items to be part of long and frequent itemsets.

Why at all the finding of frequent itemsets is a nontrivial problem? The first reason is the number of customer transactions in the database, which can be so large that it usually does not fit into the memory of a system. The second is the potential number of frequent itemsets is exponential on the number of different items, although the actual number of different itemsets can be much smaller. If a table has four different itemsets, there can be $2^4 - 1 = 15$ potential frequent itemsets. But, if minimum support is taken as 60%, only five itemsets are actually frequent. Thus, there is need of algorithms that are scalable with respect to the number of transactions, such that they need to examine as few infrequent itemsets as possible.

Nested Hash Tables

This is a new and efficient technique, which makes use of hash tables in a nested manner to check as to which itemsets are contained in a transaction. This approach is more effective when it is required to count shorter itemsets. A technique for longer itemsets is *database projection* technique, given as Algorithm 17.4.

Algorithm 17.4 Database Projection algorithm for counting itemsets

1: Partition the candidate itemset into groups such that candidates in each group share a set of common items.
2: Discards transactions T_d, that do not include all the common items, from T. (T = all transactions)
3: Discards the common items from remaining set ($T - T_d$) (since they are known to be present.)
4: Discard the items not present in any of the candidates.
5: Count each of the candidate group.

This reduction in the number and size of the remaining transactions can result in substantial improvements in the speed of counting.

17.11 Sequential Pattern Mining Algorithms

Sequential pattern mining is aimed at discovering patterns of frequent subsequences in a sequence database. This database usually comprises a large number of sequences of ordered events, with or without a notion of time; each such subsequence is considered as a database record. Such sequences commonly occur in any metric space, either as a *total ordering* or *partial ordering*. There can be many examples of such sequences: events in time, in codons, or nucleotide in amino acid, in computer networks, in website traversals, or characters in a text string are all examples, where existence of a sequence may be significant, and the detection of frequent subsequences might be useful. *Sequential pattern mining* has emerged as a technology to discover such subsequences [10].

The sequential pattern mining problem may be defined as follows: given a database of sequences, such that each sequence is a list of transactions ordered by transaction time, find out all sequential patterns with a user-specified *minimum support* value. Here, a transaction consists of a set of items. The support of a pattern is the number of data sequences that contain the pattern as a fraction of the total number of transactions [11]. This parameter indicates a *minimum* number of sequences in which a pattern must appear to be considered frequent. For example, if a user sets the minimum support threshold to 2 sequences, the task of sequential pattern mining consists of finding all subsequences appearing in at least 2 sequences of the input database.

17.11.1 Problem Statement

A sequential pattern mining problem can be stated using the following examples.

- Can we find out some strong correlations between users doing online purchasing from a web page based on their behavior patterns versus the sequences of pages visited and/or speed at which he/she browses a website pages?
- A *market basket* is a record with fields: customer ID, and list of all items, with most of the item fields as False, and True only where the customer has bought those items. Can we derive through analysis, more useful information about the buying pattern of the a customer, if information related to time and sequence is also included in the market basket?
- Can we recognize a user as a genuine user or hacker based on the suspicious behavior observed due to analyzing a sequence of commands entered by that user?
- Can we declare the elements of actions as "best practices", based on the outcomes and analysis of sequence of actions performed on any activity that results in the given outcomes?
- Given the sequences of alerts and status by any system before it has failed, can we determine those sequences set or subsequence set through some analysis which are confirmatory predictions about the failure? That is, whether it is possible to map those sequences to failure?

In simple terms, having many significant or important events occurring over the scale of time, space, or some other metric, it is required to learn from the data by considering the ordered sequences appearing in the data.

17.11.2 Notations for Sequential Pattern Mining

We want to have a formally defined notation to describe the problem of mining sequential patterns. We base our discussions on *market basket*, consisting of customer ID and list of items. Let the set of items in the form of *literals* be $\mathscr{I} = \{i_1, i_2, \ldots, i_m\}$. A non-empty unordered collection of items $\alpha = (i_1, i_2, \ldots, i_k)$ is an *event*. However, for ease of processing, and without any loss of generality, we assume them to be in lexicographic order. A *sequence* $(\alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_l)$ is an ordered list of events, and a sequence with k-elements, where $k = \sum_j |\alpha_j|$ is a k-sequence, i.e., the total sum of events in that sequence. For example, $(A \rightarrow BC)$ is a 3-sequence, as there are three events A, B, C in this sequence. A sequence $\langle \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_m \rangle$ is called as a *subsequence* of another sequence $\langle \beta_1 \rightarrow \beta_2 \ldots \rightarrow \beta_n \rangle$, if there exist integers $i_1 < i_2 < \cdots < i_m$ such that $\alpha_1 \subseteq \beta_{i_1}, \alpha_2 \subseteq \beta_{i_2}, \ldots, \alpha_m \subseteq \beta_{i_m}$, and $m \leq n$. For example, the sequence $\langle A \rightarrow BC \rangle$ is a subsequence of $\langle AE \rightarrow F \rightarrow BCD \rangle$, since $A \subseteq AE$ and $BC \subseteq BCD$. Note that the order of events is also preserved in this case.

Consider that input sequences are stored in a database $\mathscr{D}$, and input sequence is an event with fields: *sequence-id*, *event time*, and the *list of items*. It is assumed that event time is unique in every event, hence this field can be used as the event ID. We define the *support* (the frequency) of a sequence, as the number of input sequences in the database that contains the sequence α_s and express it as $\sigma(\alpha, \mathscr{D})$.

The support can also calculated by a different method. For user-specified minimum support value min_supp (a frequency), a sequence is frequent if the sequence occurs more than min_supp times. We denote the frequent k-sequences as $\mathscr{F}_k$. In addition, a frequent sequence is called as *maximal* if it is not a subsequence of any other frequent sequence. Further, a frequent sequence is called *maximal* if it is not a subsequence of any other frequent sequence.

The above discussion can be termed as the problem definition for sequence mining algorithms, whose data are transaction database or they are transaction data sets.

17.11.3 Typical Sequential Pattern Mining

We consider an example of customer's retail transactions or purchase sequences in a store, which shows for each customer the store items he/she purchased in every week for a month's duration. Out of these sequences of customer purchases, each sequence can be represented using a schema [$TransactionID/CustomerID$, ⟨ordered sequence of events⟩], where each sequence may be a set of store items like *bread*, *butter*, *milk*, *vegetable*, etc. Considering only two customers, a sequence in the database may be [$T1$, ⟨($bread$, $butter$), ($bread$, $vegetable$, $butter$), ($bread$), ($coffee$, $milk$)⟩]; [$T2$, ⟨($milk$), ($coffee$, $bread$), ($vegetbable$, $milk$)⟩]. We note that customer having transaction ID $T1$ made a purchase in each of the 4 weeks in the month, while the customer with transaction ID $T2$ made purchases during 3 weeks. There can be one or more items purchased in each of an event, i.e., during each market visit. Hence, different sequences can have different lengths ($T1$ has four and $T2$ has three), and each event in a sequence can have one or more items in its set.

Sequence databases are mined using a sequential pattern mining algorithm, which looks for repeated patterns, called *frequent sequences*. These patterns can be later used to find associations between different items or events in their data, with end use like marketing campaigns, reorganization of business, prediction, planning, etc. Among the most common applications of sequential pattern mining, are web-based applications, making use of WWW for e-commerce, business, trading, etc. Typical applications of web usage mining are the areas, like user modeling such as web content personalization, prefetching and catching, reorganization of website, e-commerce, and business intelligence.

Web usage mining is an application of sequential pattern mining, which confines to finding user's navigational patterns on WWW. These patterns are extracted from web logs. Against the market-basket concept, where an event comprises the number of items, the ordered events in web mining are each comprising only one item. This is because a web user can access only one page at a time, and not many items in

an event. If, instead of an event time, a *time window* is considered for web mining, then there can be more than one item (multiple pages), and in that case it becomes a general case like market basket, having many items in an event, where an event is in the form of a time window.

Example 17.7 Sequential pattern mining to raise promotional sale in a online store.

As discussed above, when a user is accessing web pages, such that only one page is accessed at a time, it gives rise to a sequence database, which is equal to only one item in each sequence's ordered event list.

Let $E = \{a, b, c, d, e, f\}$ be a set of events for six products' web pages accessed by different users in different order in some online sales application. The E also indicates a set of six products.

We assume that web accesses database is web log, and for four users the web access sequence is in the form of four records: $[T1, \langle adbce\rangle]$; $[T2, \langle abecdf\rangle]$; $[T3, \langle babfaec\rangle]$; $[T4, \langle abfac\rangle]$. From the web-log pattern mining of this web sequence database, we find a frequent sequence abc, indicating that all the users who visit product a's web page http://www.prompt-sale.com/a.htm also immediately visit product b's web page http://www.prompt-sale.com/b.htm and then visit product c's page http://www.prompt-sale.com/c.htm (These are only ficticious names of portals).

Having the patterns sequence of visiting the web pages in a certain order, e.g., the customers who visit the web page for product a will also visit web page b and c, it is possible to increase the sales of product b and c. By placing promotional discounts on product a's web page, which is visited a number of times in sequence, it is possible to increase the sale of other products b, and c. □

The web log can be maintained on the server where the web page is stored. However, it can also be stored on the client side, or on a proxy server. Each choice has its own advantages and disadvantages concerning finding the users' relevant patterns and navigational sessions.

17.11.4 Apriori-Based Algorithm

Apriori algorithm is used for mining of frequent itemsets, and for *association rule* learning over transactional databases. The algorithm works based on the identification of frequent individual items in the database and extending them to larger and larger itemsets, as long as those itemsets appear frequently in the database. Apriori has the property that "any subset of a frequent itemset must also be frequent." This algorithm makes several rounds of computation to compute frequent itemsets, such that in the ith round it computes all frequent i-itemsets. A round in the Apriori algorithm has the following three steps:

1. candidate generation,
2. candidate counting, and

3. discarding unimportant candidates.

In the ith round of the candidate generation step, it generates a set of candidates with i itemsets. Then, in the candidate counting step, it scans the transaction database and counts the support of candidate itemsets. In the third step, candidates with support lower than a user-specified minimum threshold are discarded, and frequent i itemsets are retained [10].

In the first round, the *Apriori* algorithm generates a set of candidate itemsets each containing all 1 itemsets, and counts their support value. Therefore, after the first round, all frequent 1 itemsets are known. Similarly, the candidate itemsets generated during the candidate generation step of round two are roughly all pairs of items. Apriori algorithm reduces the set of candidate itemsets through pruning—a priori, based on the knowledge about infrequent itemsets obtained from the previous rounds. Those are the itemsets which cannot be frequent. The concept of pruning used here is based on the general rule that "if an itemset is frequent", all its subsets must also be frequent. Hence, before we go to the candidates' counting step, it is better to discard every candidate itemset whose subset is infrequent [11].

The *Apriori*-based algorithm is presented as Algorithm 17.5. The task of the algorithm is to find the set F of frequent sequences (also known as frequent sequential patterns), for a given minimum support threshold min_supp.

Algorithm 17.5 Apriori-based algorithm

1: Start scanning of database in Apriori-based algorithm at $k = 1$
2: Let C_1 = initial sequence generated in 1st generation (set of *candidate* 1-*sequences*)
3: **repeat**
4: $k = k + 1$ (next step)
5: Scan the entire database (In kth-iteration it finds frequent itemsets of size k)
6: Join F_{k-1} frequent itemsets with itself to generate C_k (set of candidate sequences in the kth-iteration).
7: Prune sequences in C_k which have no subsequences in F_{k-1} (i.e., they are not large)
8: Create frequent itemsets F_k by adding all sequences from C_k with support $\geq min\_supp$
9: **until** no more candidate sequences left
10: **end**

The scanning of the database in an Apriori-based algorithm starts at $k = 1$. Let us call the initial sequence generated in the first generation as C_1, the set of candidate 1-sequences (in kth generation as C_k). Subsequently, the database is scanned by the algorithm several times. Let, in kth-iteration it find frequent itemsets F_k, of size k ($k \geq 2$). Next, the algorithm performs Apriori-generate and join ($\bowtie_{ap-gen} F_1$), to join $k-1$ frequent itemsets in F_{k-1} with itself to generate C_k, the set of candidate sequences in the kth-iteration. It then prunes sequences in C_k which have no subsequences in F_{k-1} (i.e., which are not large), and creates F_k by adding all sequences from C_k with support at least equal to min_supp. The above iteration process goes on for $k = 1, 2, \ldots$ until there are no more candidate sequences left.

Example 17.8 Illustrating iteration steps in the Apriori algorithm.

Let the total number of items after scanning in the fist generation be 5, i.e., $C_1 = \{a : 6, b : 6, c : 3, d : 3, e : 5\}$, here every item has been represented with its support in the form $s : n$, where s is the sequence, and n is the support count. Let us assume that a minimum absolute support is $min\_supp = 4$ (i.e., 67%). After pruning is over, the list of frequent 1-sequences is $F_1 = \{a : 6, b : 6, e : 5\}$. Using the join operation ($\bowtie$), the Apriori has generated candidate sequences at $k = 2$ as, $C_2 = F_1 \bowtie_{ap-gen} F_1 = \{aa : 3, ab : 5, ae : 4, ba : 3, bb : 4, be : 4, ea : 1, eb : 2, ee : 0\}$. Here, $\bowtie_{ap-gen}$ stands for Apriori "generate and set-join operation." Note that sequences like ac, ad are not considered as their subsequences: a, c and a, d are each pair not frequent, and similar rules applied to the others cases. The support count in the sequences of C_2 is as per their frequency in the database.

Next, it prunes all sequences in C_2 that do not have frequent subsequences (i.e., their subsequences do not appear in F_1), and so on. The final resulting frequent set is $f_s = U_k F_k = \{a, b, e, ab, ae, bb, be, aba\}$, where U_k is the union of k number of frequent sequences of F_k. The pruning procedure performed on itemsets in C_k removes sequences that have infrequent subsequences. □

Example 17.9 Perform data mining using Apriori-based algorithm to find out the frequent itemsets for the sales transaction data, for a minimum support of 60%.

Consider the transactional database shown in Table 17.7 (page number 539); for convenience of reference it is reproduced as Table 17.8.

For min_supp value of 60%, an itemset is frequent if it is present in at least two transactions. As per the Apriori algorithm, we perform many rounds through the given database. In the first round all single items, which are candidate itemsets, are counted during the candidate *counting step*. The frequent itemsets resulting in 1-sequence itemset are {$Laptop$(67%), Smartphone(100%), Music system(67%)}.

In the second round, the pairs are counted as frequent itemsets, and only those pairs are candidates in which the individual item is frequent in round one, and they are present in at least 60% of the transactions. As per these restrictions, the 2-sequence

Table 17.8 Transactional database

Tran. ID	Cust. ID	Item name	Price (in $)	Date
101	201	*Laptop*	1500	8/20/2018
101	201	*Tablet*	300	8/20/2018
101	201	*Smartphone*	100	8/20/2018
102	201	*Music system*	500	8/25/2018
102	201	*Smartphone*	100	8/25/2018
103	202	*Laptop*	1500	8/30/2018
103	202	*Music system*	500	8/30/2018
103	202	*Smartphone*	100	8/30/2018

itemsets are *{Laptop, Smartphone}*(67%), *{Music system, Smartphone}*(67%). Note that the pairs in which *{Tablet}* is an item are dropped, as it is not a frequent item in the first round. However, it disqualifies as a pair also due to minimum support.

In round three, no itemset qualifies for 3-sequence. Hence, with respect to a minimum support of 60%, the frequent itemsets in this sample database and their support values are

$\{Laptop\}$ 67%,
$\{Musicsystem\}$ 67%,
$\{smartphone\}$ 100%,
$\{Laptop, smartphone\}$ 67%, and
$\{Musicsystem, smartphone\}$ 67%.

The Apriori algorithm counts the support of all frequent itemsets, as well of those infrequent candidate itemsets that could not be eliminated during the pruning step. These latter items sets are called *negative border*. Note that an item is in negative border if it is infrequent, but its subsets are frequent. In the example above, the only negative border itemset is {*Laptop, Music system*}. It is necessary that all subsets of an itemset in the negative border are frequent, in the absence of that the item should have been eliminated by the subset pruning step. □

We know that the total number of subsequences possible for n items is 2^n, which is exponential. Accordingly, as the number of sequences in the database becomes larger, the Apriori techniques suffer exponential growth of candidate sequences during execution, and consequently, suffer increased delays in mining.

The Apriori is a family of algorithms; they are better suited for discovering intra-transaction associations and then to generate rules about the discovered associations. However, the task of mining the sequences is actually discovering inter-transaction associations, i.e., sequential patterns across the same or similar data. Hence, the algorithm uses transactional databases as its data source.

One form of the Apriori algorithm has *horizontal formatting*, i.e., the original data is sorted first by Customer_Id and then by Transaction-time, so that each customer's transactions appear together, but in the order of time stamp of the transaction. This transformation results to the database, where the time stamps determine the order of events. The mining is then performed on these database using the breadth-first search (BFS) approach. Following are the steps of Apriori-based algorithm using horizontal formatting.

1. *Sort phase.* This phase transforms the data set from the original database into customer sequence database by sorting on the primary key as Customer_Id, secondary key as Transaction_Time.
2. L_itemset phase. This phase finds all *large itemsets L* (which meet the minimum support).
3. *Transformation phase.* As there is a repeated requirement to find out which of the long sequences are present in the customer sequence, each customer's sequence is transformed by replacing the corresponding transaction with the set of L_itemsets contained in that transaction. The transactions without any L_itemsets

are dropped, as well as the customer sequences not containing any L_itemsets, but these customer sequences are still considered as part of database for the purpose of customers' count.

4. *Sequence phase.* The sequence phase mines the set of L_itemsets to find the frequent subsequences. The algorithm make multiple passes over the data, with each pass beginning with a *seed-set*, for producing potentially large sequences (i.e., candidates), and these candidates' support is also calculated during this pass. The sequences not meeting the minimum support criteria are pruned, and the remaining become the seed-set for the next pass. The process starts with the large 1-sequences, and terminates when either no candidates are generated or none meet the minimum support.
5. *Maximal Phase.* The phase finds all maximal sequences among the set of large sequences. The process is similar to finding all subsets of a given itemset, the algorithm is similar as well.

Example 17.10 (*Association rule mining*) Consider the database having four transactions as shown in Table 17.9. It is required to find the association rules if the minimum support is 60% and minimum confidence is 80%.

We perform the following steps for the solution.

1. We note that item a appears in four transaction, hence, it has support of 100. Accordingly, the items with frequency and support are $(a, 4, 100\%)$, $(b, 4, 100\%)$, $(c, 2, 50\%)$, $(d, 3, 75\%)$, $(e, 2, 50\%)$, and $(f, 1, 25\%)$.
2. Next, construct the itemsets having two items using the previous phase with support $\geq 60\%$: $(a, b, 4, 100\%)$, $(a, d, 3, 75\%)$, and $(b, d, 3, 75\%)$.
3. Next, construct the itemsets having three items using the previous phase with support $\geq 60\%$ or more : $(a, b, d, 3, 75\%)$.

Next, we formulate rules and compute their confidence. For this, we take the items from the previous phases with confidence at least 60% (see Table 17.10).

The rules with confidence less than 80% are pruned, and we are left with rules shown in Table 17.11.

Table 17.9 Transactions database

Transaction_Id	Itemset
T_1	$\{a, b, d, f\}$
T_2	$\{a, b, c, d, e\}$
T_3	$\{a, b, c, e\}$
T_4	$\{a, b, d\}$

Table 17.10 Rules with confidence ≥60%

Rule	Confidence c (%)
$a \rightarrow b = P(b\|a) = \frac{\|b \cap a\|}{\|a\|} = \frac{4}{4}$	100
$b \rightarrow a = P(a\|b) = \frac{\|a \cap b\|}{\|b\|} = \frac{4}{4}$	100
$a \rightarrow d = P(d\|a) = \frac{\|d \cap a\|}{\|a\|} = \frac{3}{4}$	75
$d \rightarrow a = P(a\|d) = \frac{\|a \cap d\|}{\|d\|} = \frac{3}{3}$	100
$b \rightarrow d = P(d\|b) = \frac{\|d \cap b\|}{\|b\|} = \frac{3}{4}$	75
$d \rightarrow b = P(b\|d) = \frac{\|b \cap d\|}{\|d\|} = \frac{3}{3}$	100
$ab \rightarrow d = P(d\|ab) = \frac{\|d \cap ab\|}{\|ab\|} = \frac{3}{4}$	75
$d \rightarrow ab = P(ab\|d) = \frac{\|ab \cap d\|}{\|d\|} = \frac{3}{3}$	100
$ad \rightarrow b = P(b\|ad) = \frac{\|ad \cap b\|}{\|ad\|} = \frac{3}{3}$	100
$b \rightarrow ad = P(ad\|b) = \frac{\|b \cap ad\|}{\|b\|} = \frac{3}{4}$	75
$bd \rightarrow a = P(a\|bd) = \frac{\|bd \cap a\|}{\|bd\|} = \frac{3}{2}$	100
$a \rightarrow bd = P(bd\|a) = \frac{\|a \cap bd\|}{\|a\|} = \frac{3}{4}$	75

Table 17.11 Rules with confidence ≥80%

Rule	Confidence c (%)
$a \rightarrow b = P(b\|a) = \frac{\|b \cap a\|}{\|a} = \frac{4}{4}$	100
$b \rightarrow a = P(a\|b) = \frac{\|a \cap b\|}{\|b\|} = \frac{4}{4}$	100
$d \rightarrow a = P(a\|d) = \frac{\|a \cap d\|}{\|d\|} = \frac{3}{3}$	100
$d \rightarrow b = P(b\|d) = \frac{\|b \cap d\|}{\|d\|} = \frac{3}{3}$	100
$d \rightarrow ab = P(ab\|d) = \frac{\|ab \cap d\|}{\|d\|} = \frac{3}{3}$	100
$ad \rightarrow b = P(b\|ad) = \frac{\|ad \cap b\|}{\|ad\|} = \frac{3}{3}$	100
$bd \rightarrow a = P(a\|bd) = \frac{\|bd \cap a\|}{\|bd\|} = \frac{3}{2}$	100

17.12 Scientific Applications in Data Mining

The scientific applications have resulted in the accumulation of high-dimensional data, data in the form of data streams, and temporal and spatial data. Following are potential scientific applications of data mining [12].

Biomedical Engineering

Examples of data in biological sciences include genome DNA sequence sequencing of organisms, as well as large macro molecules such as proteins, and RNA. Due to the large size availability of this data, there is need to create systems for organizing, storing, and dissemination. In addition, there is abundance of potential for automated learning from these data sets. A number of robust machine learning and data mining algorithms have emerged to take advantage of previous knowledge, and to create new knowledge. Consequently, biology is changing from the field of research which was dominated by—"formulation of hypothesis, conducting experiments, and evaluate results", to more of a computational science, with attitude of "collecting and

storing data, mining new hypotheses, and confirm these with data or supplement by experiment." These results/data can be combined with the clinical data to achieve better results and resolution for treatments.

Telecommunications

Due to the wide use of telecommunication in the world, vast quantities of high-quality data is available already, mainly including call details collected at network switches mainly for billing, which can be used for data mining tasks in detection of toll fraud and marketing to consumers. Based on the data mining algorithms on telecommunications data, the following benefits can be drawn:

- *New architectures for networks.* The new generation network infrastructures may adapt to changes in traffic demands dynamically, to be achieved through data mining techniques to understand and predict the network load.
- *Mobility and micro-billing.* Consumer-related activities may have separate billing patterns.
- *Mobile services.* Data mining may enable customers for adaptive solutions, so that they can get it through a few keystrokes.
- *Security.* Data may be collected and maintained through the records of billing, travels, migration, to obtain national security-related information, to ensure the national security.

Geospatial Data

The geographical data has scope and uses, which include digital data of all sorts, which can be created, stored, processed, and disseminated by government and private organizations, many of them through high-resolution remote sensing devices, data collected through geographical positioning systems, and other devices, which are position aware, like cellular phones.

The new pattern mining and clustering algorithms, which are highly scalable, are useful to discover new and unexpected patterns, trends, and relationships embedded in these data.

Climate Data and Earth's Ecosystems

The climate data acquired through the terrestrial observation, Earth-observation satellites, and ecosystem models provides lot of data, which can be mined for patterns to discover future ecological problems and their management, including ecology and even the health of the planet Earth. Such data consists of global snapshots of the Earth, at typical intervals, with variables like atmosphere, land, and ocean (sea surface temperature, precipitation), or accumulation of carbon.

There are two main components of Earth science data mining: (1) Modeling ecological data, and (2) Design of efficient algorithms for discovering potential patterns in these data. Through these patterns as well the existing patterns, it may be possible to predict the effects, such as El Nino and tornado, etc.

17.13 Summary

The digital storage capacity has doubled over every 9 months—this is the phenomena, which has necessitated the mining of data. Data mining provides tools, to sift through vast amount of data accumulated by organizations over the years, to find the trends, patterns, and correlations, which can be helpful in planning.

Data mining process is confronted with dimensionality, size, diversity, noise, and distributed nature of data sets, which makes the formal specification of the problem, a challenge. The solution techniques needs to deal with complexity, scalability, and presentation. The entire process through which data mining makes its transitions is called *data science*. The algorithms on data mining address three classical data mining problems, i.e., clustering, market bucket analysis, and density estimation.

One important difference between the traditional databases, and data mining is the size of data in later, such that data does not fit in the storage of computer, therefore, the mining algorithm should be scalable. Data mining is concerned with the identification of interesting data structure in the data, which may be patterns, statistical or predictive model, or relation information. A model can be used to predict future customer behaviors, e.g., the customer "A" may buy the so and products and goods.

The major goals of data mining are

- scaling analysis of large databases,
- scaling to higher dimensional and data models,
- automated search, and
- finding patterns and models understandable.

Five perspectives from the existing areas of machine learning (statistics, algorithms, and databases) are used in data mining. These comprise induction, compression, querying, approximation, and search.

The data mining techniques are also classified based on the following criteria:

- their *induced representations*, e.g., decision trees, association rules, and correlations;
- the *data* on which they operate, e.g., continuous, discrete, labeled, time series; or
- *application domains*, e.g., finance, economic models, web-log mining, and semi-structured models.

The common form of data for data mining is records of individuals, like transaction record or record in the form of requested web page—the set of such records can be viewed as N *rows* $\times$ d *column* matrix, and entry (i, j) is 1, for example, in a transaction, if item j was purchased in the session i. The itemsets (d) in a row is called *basket*. In one approach to data mining, the objective is to find statistically significant itemset, that is, whose frequency is higher than some baseline frequency. The transaction data is often in the form of a continuous stream, like continuous sales transactions, or web logs. This gives a challenge on how to compute aggregate of the data. The *incremental learning* algorithms are used in such cases.

Many of the traditional algorithms access the databases multiple times, or randomly for analysis. This is not possible when databases are very large. The algorithm used for such large databases are *prediction methods, clustering*, and *association rules*. The clustering partitions a set of records according to some similarity function, into groups such that "similar" records are in the same group, an identification of similar subpopulations in the data. Clustered records are represented by a summarized feature called *cluster feature* (CF), which are efficient as they occupy less space. This grouping is sufficient for inter-cluster and intra-cluster measurements. A pattern $\mathbf{x}$ (which may be a feature vector, a datum, or observation) is a single data item used by the clustering algorithms. It is typically a vector of d dimensions, represented as $\mathbf{x} = (x_1, \ldots, x_d)$. In the feature vector $\mathbf{x}$, the individual scalar components x_i are called features (or attributes).

Structured features can be represented using a tree data structure. The *feature selection* techniques identify a subset of the existing features to be used later, while *feature extraction* techniques compute new features from the original set. It is common to measure the dissimilarity between two patterns by measure of distance between these objects, say, O_1 and O_2, determined by $d(O_1, O_2)$.

The *association rules*, another method of classification, makes the use of market basket—collection of items purchased by a customer—finds correlation between itemsets, helping to understand the purchase behavior of customers. The clustering process groups the data objects using the information found in the data items as well as the relation between data items. Clustering is also called *unsupervised classification*, where it is required to group the given unlabeled patterns into meaningful clusters. The *supervised classification* classifies the unlabeled patterns into pre-classified patterns.

General applications of clustering are for pattern analysis, decision-making, and machine learning. Typical applications are Information Retrieval (IR), analysis of semantic information, analysis of business (sales) transactions, e.g., to find out potential customers, etc. The utilities like summarization, nearest neighbor, and compression can also be constructed based on techniques of data clustering. Typical pattern clustering activity involves—pattern representation, feature selection and extraction, pattern proximity measures, and data groupings (clustering).

Proximity or similarity measure is done by a distance function in Euclidean distance, where grouping can be based on many criterions, like soft (fuzzy) versus hard partitioning, hierarchical versus partitional algorithms, etc. Patterns for clustering algorithm may be represented using Cartesian coordinates or polar coordinates, and a pattern can measure either a physical object, like chairs, or an abstract notion, like style of writing. Different types of clustering are categories, which are distinguished as having contrast characteristics, e.g., monothetic *versus* polythetic, hierarchical (nested) *versus* partitional (un-nested), exclusive versus fuzzy, and complete versus partial. The hierarchical clustering algorithm yields nested grouping of patterns in the form of a *dendogram*, such that two clusters can be nested to form a larger cluster based on minimum distance criterion.

When the patterns set to be used is very large such that constraints on memory size and execution time affect the algorithm, we use the incremental algorithm. The latter has advantage that it does not require the entire matrix to be stored in the memory, hence space requirement is less. Since these algorithms are not iterative, the time required is also less.

Since proximity between items plays an intuitive role of clustering, the nearest neighbor approach is a useful basis of clustering procedures, which iteratively assigns each unlabeled pattern to the cluster of the nearest labeled pattern provided that the distance with that is below some prespecified threshold value. The partitional technique of clustering produce clusters by optimizing a criterion function defined on a subset of patterns or on all patterns. Its algorithm is run multiple number of times, and every time with different starting points, and the best outcome is considered from all the runs.

Many data mining approaches generate classification trees—a tree whose leaves are classifications and their branches are conjunctions of features that lead to those classifications. The decision-tree algorithm has two phases—tree building and tree pruning. In the tree building phase, the algorithm examines the database using the*split selection method* to compute the locally best criteria. This works recursively to build trees.

We can mine the rules in the data, called *association rules*. The rules mining algorithms capture the set of important correlations present in the database. For a given set of transactions where each transaction is a set of items, an associative (or association) rule can be an implication $X \Rightarrow Y$, e.g., in case of sales transactions, this may indicate that those who purchase itemset X may also purchase itemset Y. Therefore, for selling itemset Y, one may target the people whose bought X, and not again who bought itemset Y. The association rule mining consists of two steps: (1) find all itemsets with minimum support, and then, (2) generate the inferences from these minimum support itemsets.

Sequential pattern mining technique can be described as given a database of sequences where each sequence is a list of transactions ordered by transaction time, discover all sequential patterns having a minimum support. A *sequence* in a sequential pattern mining is an ordered list of events, where an event is denoted as $\langle i_1, i_2, \ldots, i_k \rangle$, and i_j is an item.

A sequential pattern mining algorithm mines the database of sequences, looking for repeating patterns (known as frequent sequences). These sequences can be analyzed to find associations between different items or events in their data for purposes, like marketing campaigns, business reorganization, prediction, and planning.

Web usage mining is an application of sequential pattern mining concerned with finding user navigational patterns on the World Wide Web, by extracting knowledge from web logs.

A commonly used family of algorithms, called Apriori algorithm, computes frequent itemsets through several rounds, such that in a round i it computes all i-itemsets' frequent transactions. A round has two steps: 1) candidate generate, 2) candidate counting, i.e., those having support greater than min_supp.

The scientific applications have resulted in accumulation of high-dimensional data, stream data, and spatial and temporal data. Hence, they are most appropriate fields for data mining algorithms. Some of the applications of data mining in scientific fields are Biomedical engineering, Telecommunication, Geospatial data, Climate data, and Earth ecosystems.

Exercises

1. How is data mining different from querying databases, like Oracle or MySql?
2. What were the trends in Information Technology (IT), which gave birth to the field of data mining? Why did the field of data mining emerge so late compared to databases?
3. Are the goals presented in this chapter identically applicable to all the data mining domains, for example, mining of data generated due to collision in particle accelerators versus online sales transactions versus Twitter data? Justify your answer.
4. Given that the *Apriori algorithm* makes use of prior knowledge of subset support properties,
   a. Show that all non-empty subsets of a set of frequent items must also be frequent.
   b. Show that the support of any non-empty subset R of itemset S must be at least as large as the support of S.
5. The algorithms for frequent patterns mining consider only distinct items in a transaction (market basket or shopping basket). However, the multiple occurrences of an item are common in a shopping basket, e.g., we often buy the things like 10 eggs, 3 breads, 2 kg dalia, 4 kg oil, 2 kg milk, etc., and this can be important in transaction data analysis. Suggest an approach on how you will modify the Apriori algorithm, or propose alternate method to efficiently consider multiple occurrences of items?
6. Assume nonnegative prices of items in a store, and find out the nature of constraint they represent in each of the following cases. Also suggest, how you will mine the association rules in these.
   a. At least one Sudoku game.
   b. Itemsets, whose sum of prices is less than $250.
   c. There is one free item, and other items, whose sum of prices is equal or more than $300.
   d. The average price of all the items is between $200 and $500.
7. Given a decision tree, you have the following options:
   a. Convert the decision tree into rules, and then prune the resulting rules,
   b. Prune the decision tree and then convert the pruned tree into rules.

   Critically analyze both the approaches, and discuss their merits and demerits.

8. Find out the worst case time complexity of decision-tree algorithm. Assume that data set is D, each item has n number of attributes, and the number of training tuples are $|D|$. (Hint. Answer is $|D|.n.\log|D|$.)
9. It is required to cluster a given set of data into three clusters, where (x, y) represent the location of the object, and the distance function is Euclidean distance. The points are $P_1(3, 12)$, $P_2(3, 8)$, $P_3(9, 5)$, $Q_1(4, 7)$, $Q_2(6, 4)$, $Q_3(7, 5)$, $R_1(2, 3)$, and $R_2(5, 8)$. Use the k-means algorithm to show the three cluster centers after the first round of execution, and after the final round of execution.

References

1. Fayyad U, Uthurysamy R (2002) Evolving data mining into solutions for insights. Commun ACM 45(8):28–31
2. Han J et al (1999) Constraint-based multidimensional data mining. Computer 4:46–50
3. Ramakrishnan N, Ananth YG (1999) Data mining: from serendipity to science. Computer 8:34–37
4. Smyth P et al (2002) Data-driven evolution of data mining algorithms. Commun ACM 45(8):33–37
5. Bradley P et al (2002) Scaling mining algorithms to large databases. Commun ACM 45(8):38–43
6. Karpis George et al (1999) Hierarchical clustering using dynamic modelling. Computer 4:68–75
7. Jain AK et al (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
8. Ganti V et al (1999) Mining very large databases. Computer 8:38–45
9. Ceglar A, Roddick JF (2006) Association mining. ACM Comput Surv 38(2):1–46
10. Nizar R et al (2010) A taxonomy of sequential pattern mining algorithms. ACM Comput Surv 43:1:3.1–3.41
11. Carl H et al (2013) Sequential pattern mining—approaches and algorithms. ACM Comput Surv 45(2):19:1–19:39
12. Han J et al (2002) Emerging scientific applications in data mining. Commun ACM 45(8):54–58