DATA MINING

Subject Code: 06IS74

PART - A

Unit – 1 **6 Hours**

Introduction, Data – 1: What is Data Mining? Motivating Challenges; The origins of data mining; Data Mining Tasks. Types of Data; Data Quality.

Unit – 2 **6 Hours**

Data – 2: Data Pre-processing; Measures of Similarity and Dissimilarity

Unit – 3 **8 Hours**

Classification: Preliminaries; General approach to solving a classification problem; Decision tree induction; Rule-based classifier; Nearest-neighbour classifier.

Unit – 4 **6 Hours**

Association Analysis – 1: Problem Definition; Frequent Item set generation; Rule Generation;

Compact representation of frequent item sets; Alternative methods for generating frequent item sets.

PART - B

Unit – 5 **6 Hours**

Association Analysis – 2: FP-Growth algorithm, Evaluation of association patterns; Effect of Skewed support distribution; Sequential patterns.

Unit – 6 **7 Hours**

Cluster Analysis: Overview, K-means, Agglomerative hierarchical clustering, DBSCAN, Overview of Cluster Evaluation.

Unit – 7 **7 Hours**

Further Topics in Data Mining: Multidimensional analysis and descriptive mining of complex data objects; spatial data mining; Multimedia data mining; Text mining; Mining the WWW.Outlier analysis.

Unit – 8 **6 Hours**

Applications: Data mining applications; Data mining system products and research prototypes; Additional themes on Data mining; Social impact of Data mining; Trends in Data mining.

Text Notess:

1. **Introduction to Data Mining -** Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Pearson Education, 2007

2. **Data Mining – Concepts and Techniques -** Jiawei Han and Micheline Kamber, 2nd Edition, Morgan Kaufmann, 2006.

Reference Notess:

1. **Insight into Data Mining – Theory and Practice** - K.P.Soman, Shyam Diwakar, V.Ajay, PHI, 2006.

TABLE OF CONTENTS

Part A

5.4 Sequential patterns

Unit – 6 Cluster Analysis:

6.1 Overview

6.2 K-means

6.3 Agglomerative hierarchical clustering

6.4 DBSCAN

6.5 Overview of Cluster Evaluation

Unit – 7 Further Topics in Data Mining:

7.1 Multidimensional analysis and descriptive mining of complex data objects

7.2 spatial data mining

7.3Multimedia data mining

7.4 Text mining

7.5 Mining the WWW.Outlier analysis

Unit – 8 Applications:

8.1 Data mining applications

8.2 Data mining system products and research prototypes

8.3 Additional themes on Data mining

8.4 Social impact of Data mining

8.5 Trends in Data mining

PART-A

Unit – 1

Introduction, Data – 1: What is Data Mining? Motivating Challenges; The origins of data mining; Data Mining Tasks. Types of Data; Data Quality.

Text Notess:

Data Mining – Concepts and Techniques - Jiawei Han and Micheline Kamber, 2nd Edition, Morgan Kaufmann, 2006.

UNIT – 1

INTRODUCTION: DATA – 1

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS). The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed. The proliferation of database management systems has also contributed to recent massive gathering of all sorts of information. Today, we have far more information than we can handle: from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Information retrieval is simply not enough anymore for decision-making. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the "essence" of information stored, and the discovery of patterns in raw data.

What kind of information are we collecting?

We have been collecting a myriad of data, from simple numerical measurements and text documents, to more complex information such as spatial data, multimedia channels, and hypertext documents. Here is a non-exclusive list of a variety of information collected in digital form in databases and in flat files.

- **Business transactions**: Every transaction in the business industry is (often) "memorized" for perpetuity. Such transactions are usually time related and can be inter-business deals such as purchases, exchanges, banking, stock, etc., or intra-business operations such as management of in-house wares and assets. Large department stores, for example, thanks to the widespread use of bar codes, store millions of transactions daily representing often terabytes of data. Storage space is not the major problem, as the price of hard disks is continuously dropping, but the effective use of the data in a reasonable time frame for

competitive decision making is definitely the most important problem to solve for businesses that struggle to survive in a highly competitive world.

- **Scientific data**: Whether in a Swiss nuclear accelerator laboratory counting particles, in the Canadian forest studying readings from a grizzly bear radio collar, on a South Pole iceberg gathering data about oceanic activity, or in an American university investigating human psychology, our society is amassing colossal amounts of scientific data that need to be analyzed. Unfortunately, we can capture and store more new data faster than we can analyze the old data already accumulated.
- **Medical and personal data**: From government census to personnel and customer files, very large collections of information are continuously gathered about individuals and groups. Governments, companies and organizations such as hospitals, are stockpiling very important quantities of personal data to help them manage human resources, better understand a market, or simply assist clientele. Regardless of the privacy issues this type of data often reveals, this information is collected, used and even shared. When correlated with other data this information

can shed light on customer behaviour and the like.

- **Surveillance video and pictures**: With the amazing collapse of video camera prices, video cameras are becoming ubiquitous. Video tapes from surveillance cameras are usually recycled and thus the content is lost. However, there is a tendency today to store the tapes and even digitize them for future use and analysis.
- **Satellite sensing**: There is a countless number of satellites around the globe: some are geo-stationary above a region, and some are orbiting around the Earth, but all are sending a non-stop stream of data to the surface. NASA, which controls a large number of satellites, receives more data every second than what all NASA researchers and engineers can cope with. Many satellite pictures and data are made public as soon as they are received in the hopes that other researchers can analyze them.
- **Games**: Our society is collecting a tremendous amount of data and statistics about games, players and athletes. From hockey scores, basketball passes and car-racing lapses, to swimming times, boxer's pushes and chess positions, all the data are stored. Commentators and journalists are using this information for reporting, but trainers and athletes would want to exploit this data to improve performance and better understand opponents.
- **Digital media**: The proliferation of cheap scanners, desktop video cameras and digital cameras is one of the causes of the explosion in digital media repositories. In addition, many

radio stations, television channels and film studios are digitizing their audio and video collections to improve the management of their multimedia assets. Associations such as the NHL and the NBA have already started converting their huge game collection into digital forms.

- **CAD and Software engineering data**: There are a multitude of Computer Assisted Design (CAD) systems for architects to design buildings or engineers to conceive system components or circuits. These systems are generating a tremendous amount of data. Moreover, software engineering is a source of considerable similar data with code, function libraries, objects, etc., which need powerful tools for management and maintenance.
- **Virtual Worlds**: There are many applications making use of three-dimensional virtual spaces. These spaces and the objects they contain are described with special languages such as VRML. Ideally, these virtual spaces are described in such a way that they can share objects and places. There is a remarkable amount of virtual reality object and space repositories available. Management of these repositories as well as content-based search and retrieval from these repositories are still research issues, while the size of the collections continues to grow.
- **Text reports and memos (e-mail messages)**: Most of the communications within and between companies or research organizations or even private people, are based on reports and memos in textual forms often exchanged by e-mail. These messages are regularly stored in digital form for future use and reference creating formidable digital libraries.
- **The World Wide Web repositories**: Since the inception of the World Wide Web in 1993, documents of all sorts of formats, content and description have been collected and inter-connected with hyperlinks making it the largest repository of data ever built. Despite its dynamic and unstructured nature, its heterogeneous characteristic, and its very often redundancy and inconsistency, the World Wide Web is the most important data collection regularly used for reference because of the broad variety of topics covered and the infinite contributions of resources and publishers. Many believe that the World Wide Web will become the compilation of human knowledge.

1.1What is Data Mining?:

Data Mining is a technology that uses data analysis tools with sophisticated algorithms to search useful information from large volumes of data.

Data mining is also defined as a process of automatically discovering useful information from massive amount of data repositories.

Data mining is the practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis. Data mining uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events. Data mining is also known as Knowledge Discovery in Data (KDD).

The key properties of data mining are:

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Focus on large data sets and databases

Data mining can answer questions that cannot be addressed through simple query and reporting techniques.

Automatic Discovery: Data mining is accomplished by building models. A model uses an algorithm to act on a set of data. The notion of automatic discovery refers to the execution of data mining models.

Data mining models can be used to mine the data on which they are built, but most types of models are generalizable to new data. The process of applying a model to new data is known as scoring.

Prediction: Many forms of data mining are predictive. For example, a model might predict income based on education and other demographic factors. Predictions have an associated probability (How likely is this prediction to be true?). Prediction probabilities are also known as confidence (How confident can I be of this prediction?).

Some forms of predictive data mining generate rules, which are conditions that imply a given outcome. For example, a rule might specify that a person who has a bachelor's degree and lives in a certain neighborhood is likely to have an income greater than the regional average. Rules have an associated support (What percentage of the population satisfies the rule?).

Grouping: Other forms of data mining identify natural groupings in the data. For example, a model might identify the segment of the population that has an income within a specified range, that has a good driving record, and that leases a new car on a yearly basis.

Actionable Information: Data mining can derive actionable information from large volumes of data. For example, a town planner might use a model that predicts income based on demographics to develop a plan for low-income housing. A car leasing agency might a use model that identifies customer segments to design a promotion targeting high-value customers.

Data Mining and Knowledge Discovery

Data mining is an integral part of Knowledge Discovery in databases (KDD), which is a overall process of converting raw data into useful information, as shown in figure below. This process consists of a series of transformation steps, from preprocessing to post-processing of data mining results.

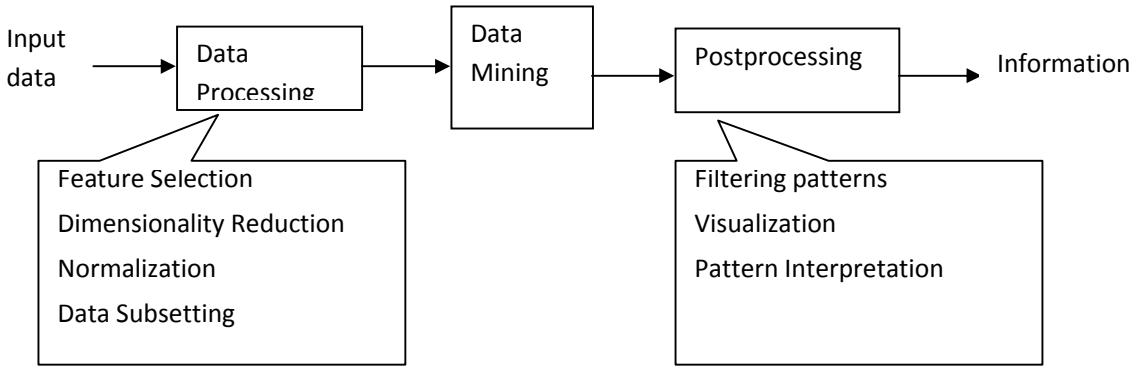


Figure 1.1 the process of Knowledge Discovery in databases (KDD).

Data Mining, also popularly known as *Knowledge Discovery in Databases* (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data mining is actually part of the knowledge discovery process.

The Knowledge Discovery in Databases process comprises of a few steps leading from raw data collections to some form of new knowledge. The iterative process consists of the following steps:

- **Data cleaning**: also known as data cleansing, it is a phase in which noise data and irrelevant data are removed from the collection.
- **Data integration**: at this stage, multiple data sources, often heterogeneous, may be combined in a common source.
- **Data selection**: at this step, the data relevant to the analysis is decided on and retrieved from the data collection.
- **Data transformation**: also known as data consolidation, it is a phase in which the selected data is transformed into forms appropriate for the mining procedure.
- **Data mining**: it is the crucial step in which clever techniques are applied to extract patterns potentially useful.
- **Pattern evaluation**: in this step, strictly interesting patterns representing knowledge are identified based on given measures.

• **Knowledge representation**: is the final phase in which the discovered knowledge is visually represented to the user. This essential step uses visualization techniques to help users understand and interpret the data mining results.

It is common to combine some of these steps together. For instance, *data cleaning* and *data integration* can be performed together as a pre-processing phase to generate a data warehouse. *Data selection* and *data transformation* can also be combined where the consolidation of the data is the result of the selection, or, as for the case of data warehouses, the selection is done on transformed data.

The KDD is an iterative process. Once the discovered knowledge is presented to the user, the evaluation measures can be enhanced, the mining can be further refined, new data can be selected or further transformed, or new data sources can be integrated, in order to get different, more appropriate results.

Data mining derives its name from the similarities between searching for valuable information in a large database and mining rocks for a vein of valuable ore. Both imply either sifting through a large amount of material or ingeniously probing the material to exactly pinpoint where the values reside. It is, however, a misnomer, since mining for gold in rocks is usually called "gold mining" and not "rock mining", thus by analogy, data mining should have been called "knowledge mining" instead. Nevertheless, data mining became the accepted customary term, and very rapidly a trend that even overshadowed more general terms such as knowledge discovery in databases (KDD) that describe a more complete process. Other similar terms referring to data mining are: data dredging, knowledge extraction and pattern discovery.

1.2 Motivating Challenges:

The motivation challenges that motivated Data mining are -:

- Scalability
- High Dimensionality
- Heterogeneous and complex data
- Data ownership and distribution
- Non-traditional Analysis

- **Scalability:** Scaling and performance are often considered together in Data Mining. The problem of scalability in DM is not only how to process such large sets of data, but how to do it within a useful timeframe. Many of the issues of scalability in DM and DBMS are similar to scaling performance issues for Data Management in general.
- **High Dimensionality:** The variable in one-dimensional data is usually time. An example is the log of interrupts in a processor. Two-dimensional data can often be found in statistics like the number of financial transactions in a certain period of time. Three-dimensional data can be positions in three-dimensional space or points on a surface whereas time (the third dimension) varies. High-dimensional data contains all those sets of data that have more than three considered variables. Examples are locations in space that vary with time (here: time is the fourth dimension) or any other combination of more than three variables, e. g. product - channel - territory - period - customer's income.

- **Heterogeneous and complex data:** heterogeneous data means data set contains attributes of different types. Traditional data analysis methods contain data sets with same types of attributes. Complex data is a data with different attribute and information. For example webpage with hyper links, DNA and 3D structure, climate data(temperature, pressure, mist, humidity, time, location,,).
- **Data ownership and distribution:** sometimes the data needed for an analysis is not stored in one location or owned by one organization. Instead the data is distributed in geographically among multiple entities. This requires the development of distributed data minnig techniques.

Challenges posed by distributed data mining

The shift towards intrinsically distributed complex problem soving environments is prompting a range of new data mining research and development problems. These can be classified into the following broad challenges:

- **Distributed data: (communication overhead)** The data to be mined is stored in distributed computing environments on heterogeneous platforms. Both for technical and for organizational reasons it is impossible to bring all the data to a centralized place. Consequently, development of algorithms, tools, and services is required that facilitate the mining of distributed data.
- **Distributed operations: (consolidation of outcomes)** In future more and more data mining operations and algorithms will be available on the grid. To facilitate seamless integration of these resources into distributed data mining systems for complex problem solving, novel algorithms, tools, grid services and other IT infrastructure need to be developed.
- **Massive data:** Development of algorithms for mining large, massive and high-dimensional data sets (out-of-memory, parallel, and distributed algorithms) is needed.
- **Complex data types:** Increasingly complex data sources, structures, and types (like natural language text, images, time series, multi-relational and object data types etc.) are emerging. Grid-enabled mining of such data will require the development of new methodologies, algorithms, tools, and grid services.
- **Data privacy, security, and governance:** Automated data mining in distributed environments raises serious issues in terms of data privacy, security, and governance. Grid-based data mining technology will need to adress these issues.
- **User-friendliness:** Ultimately a system must hide technological complexity from the user. To facilitate this, new software, tools, and infrastructure development is needed in the areas of grid-supported workflow management, resource identification, allocation, and scheduling, and user interfaces.

- **Non-traditional analysis:** it is based on hypothesis and test paradigm. Hypothesis is proposed one, its an experiment designed to gather data. Currently huge data is present in data repositories so it requires thousands of hypotheses.

1.3 The origins of data mining:

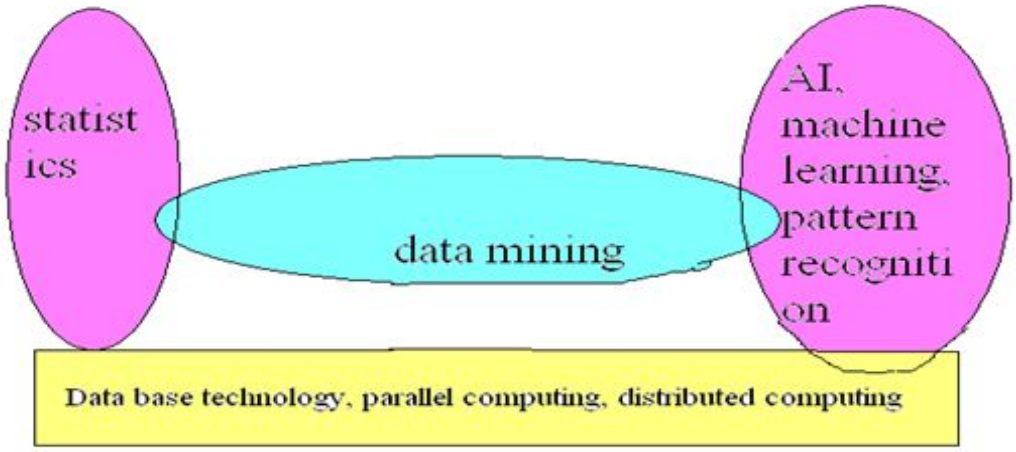


Figure 1.2: data mining as a confluence of many disciplines.

Researchers from different disciplines began to focus on developing more efficient and scalable tools that could handle diverse types of data. Data mining draws upon ideas from (1) sampling, estimation, and hypothesis test from statistics. (2) Search algorithms, modeling techniques machine learning, learning theories from AI, pattern recognition, statistics database systems.

Traditional Techniques may be unsuitable due to

- Enormity of data
- High dimensionality of data
- Heterogeneous, distributed nature of data

Data mining also had been quickly to adopt ideas from other areas including optimization, evolutionary computing, signal processing, information theory, visualizations. Database systems are needed to provide supports for efficient storage, indexing, query processing. The parallel computing and distribute technology are two major data addressing issues in data mining to increase the performance.

1.4 Data Mining Tasks:

Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories:

- Descriptive
- predictive

- Predictive tasks. The objective of these tasks is to predict the value of a particular attribute based on the values of other attribute.

  – Use some variables (independent/explanatory variable) to predict unknown or future values of other variables (dependent/target variable).

- Description Methods: Here the objective is to derive patterns that summarize the underlying relationships in data.

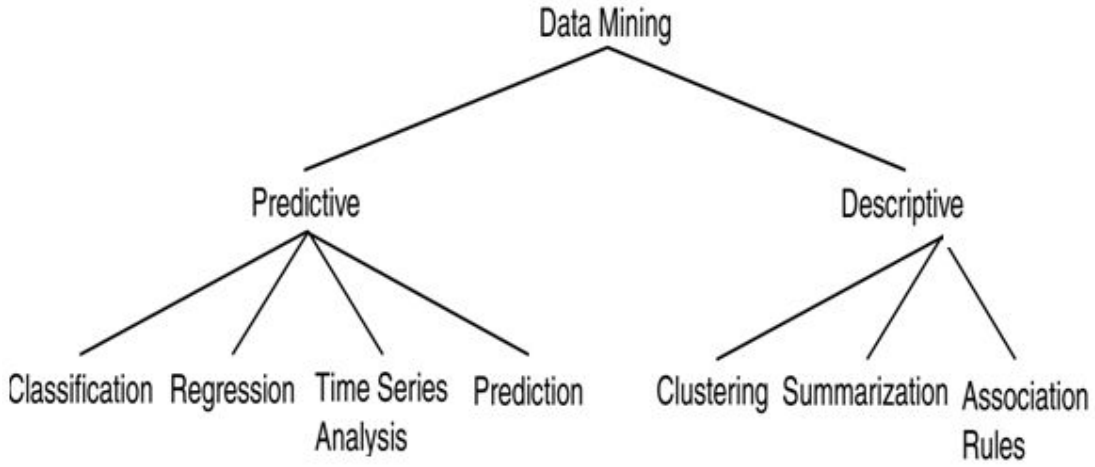
  – Find human-interpretable patterns that describe the data.

There are four core tasks in Data Mining:

i. Predictive modeling ii. Association analysis

iii. Clustering analysis, iv. Anomaly detection

Descriptive mining tasks characterize the general properties of the data in the database. Predictive mining tasks perform inference on the current data in order to make predictions.



Describe data mining functionalities, and the kinds of patterns they can discover (or) define each of the following data mining functionalities: characterization, discrimination, association and correlation analysis, classification, prediction, clustering, and evolution analysis. Give examples of each data mining functionality, using a real-life database that you are familiar with.

Figure 1.3 illustrates four of the core data mining tasks that are described in the remainder of this book.

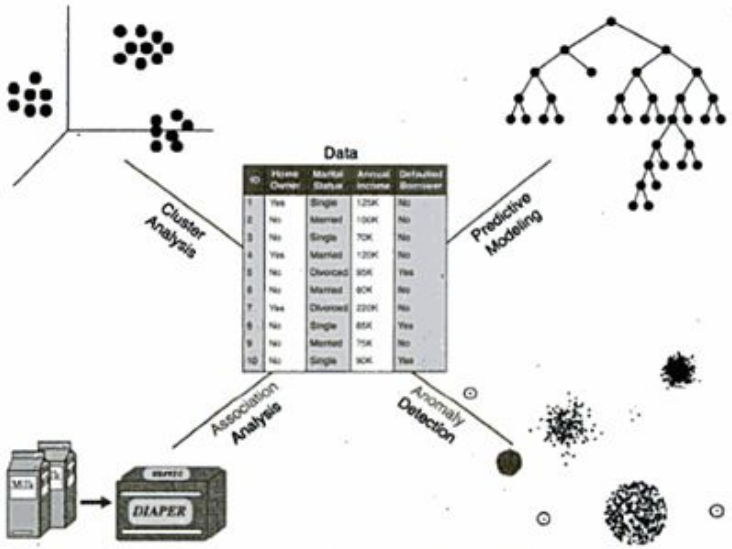


Figure 1.3. Four of the core data mining tasks.

1). predictive method

Find some missing or unavailable data values rather than class labels referred to as prediction. Although prediction may refer to both data value prediction and class label prediction, it is usually confined to data value prediction and thus is distinct from classification. Prediction also encompasses the identification of distribution trends based on the available data.

Example:

Predicting flooding is difficult problem. One approach is uses monitors placed at various points in the river. These monitors collect data relevant to flood prediction: water level, rain amount, time, humidity etc. These water levels at a potential flooding point in the river can be predicted based on the data collected by the sensors upriver from this point. The prediction must be made with respect to the time the data were collected

Classification:

- It predicts categorical class labels

- It classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Typical Applications
    - credit approval
    - target marketing
    - medical diagnosis
    - treatment effectiveness analysis

Classification can be defined as the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

Example:

An airport security screening station is used to deter mine if passengers are potential terrorist or criminals. To do this, the face of each passenger is scanned and its basic pattern(distance between eyes, size, and shape of mouth, head etc) is identified. This pattern is compared to entries in a database to see if it matches any patterns that are associated with known offenders

A classification model can be represented in various forms, such as

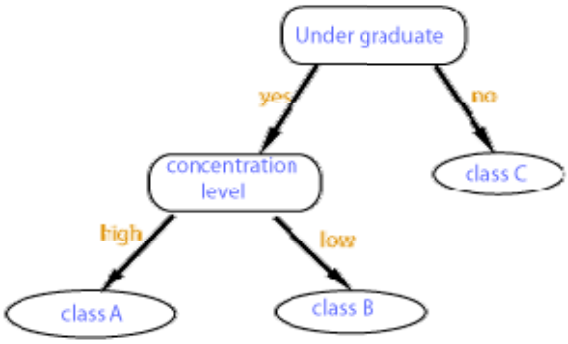
1) IF-THEN rules,

student (class , "undergraduate") AND concentration (level, "high") ==> class A

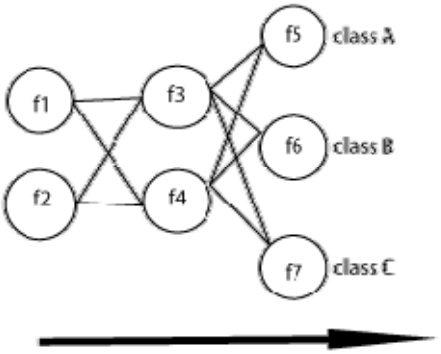
student (class ,"undergraduate") AND concentrtion (level,"low") ==> class B

student (class , "post graduate") ==> class C

2) Decision tree



3) Neural network.



.

Classification vs. Prediction

Classification differs from prediction in that the former is to construct a set of models (or functions) that describe and distinguish data class or concepts, whereas the latter is to predict some missing or unavailable, and often numerical, data values. Their similarity is that they are both tools for prediction: Classification is used for predicting the class label of data objects and prediction is typically used for predicting missing numerical data values.

2). Association Analysis

It is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. For example, a data mining system may find association rules like

major(X, “computing science””) owns(X, “personal computer”)

[support = 12%, confidence = 98%]

where X is a variable representing a student. The rule indicates that of the students under study, 12% (support) major in computing science and own a personal computer. There is a 98% probability (confidence, or certainty) that a student in this group owns a personal computer.

Example:

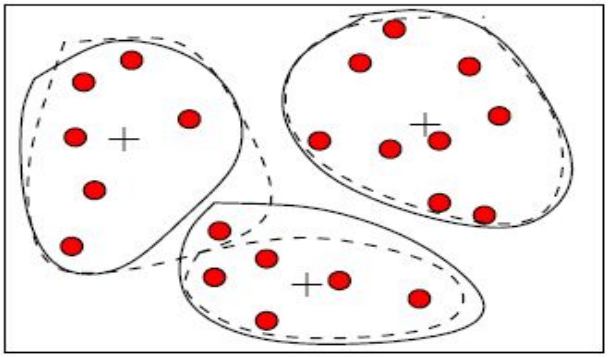
A grocery store retailer to decide whether to but bread on sale. To help determine the impact of this decision, the retailer generates association rules that show what other products are frequently purchased with bread. He finds 60% of the times that bread is sold so are pretzels and that 70% of the time jelly is also sold. Based on these facts, he tries to capitalize on the association between bread, pretzels, and jelly by placing some pretzels and jelly at the end of the aisle where the bread is placed. In addition, he decides not to place either of these items on sale at the same time.

3). Clustering analysis

Clustering analyzes data objects without consulting a known class label. The objects are clustered or grouped based on the principle of maximizing the intra-class similarity and minimizing the interclass similarity. Each cluster that is formed can be viewed as a class of objects.

Example:A certain national department store chain creates special catalogs targeted to various demographic groups based on attributes such as income, location and physical characteristics of potential customers (age, height, weight, etc). To determine the target mailings of the various catalogs and to assist in the creation of new, more specific catalogs, the company performs a clustering of potential customers based on the determined attribute values. The results of the clustering exercise are the used by management to create special catalogs and distribute them to the correct target population based on the cluster for that catalog.

Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together as shown below:



customer data with respect to customer locations in a city, showing three data clusters. Each cluster 'center' is marked with a '+'.

Classification vs. Clustering

- In general, in classification you have a set of predefined classes and want to know which class a new object belongs to.
- Clustering tries to group a set of objects and find whether there is *some* relationship between the objects.
- In the context of machine learning, classification is *supervised learning* and clustering is *unsupervised learning*.

4). Anomaly Detection

It is the task of identifying observations whose characteristics are significantly different from the rest of the data. Such observations are called anomalies or outliers. This is useful in fraud detection and network intrusions.

1.5 Types of Data:

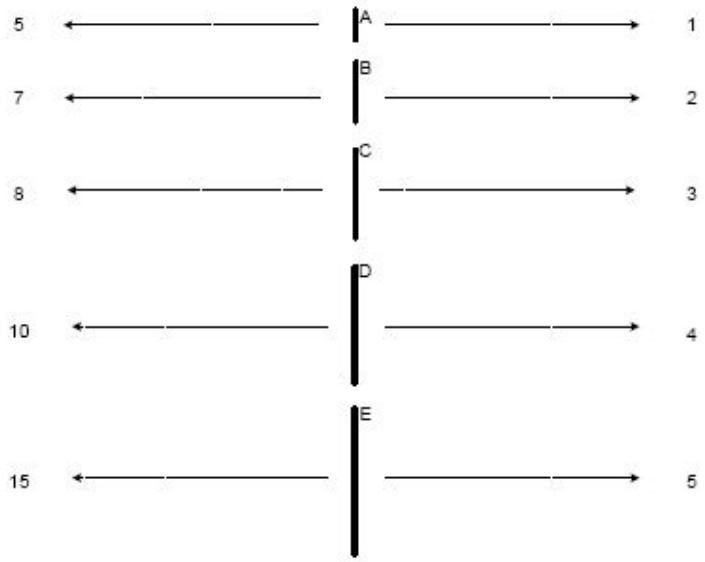
A Data set is a Collection of data objects and their attributes. An data object is also known as record, point, case, sample, entity, or instance. An attribute is a property or characteristic of an object. Attribute is also known as variable, field, characteristic, or feature.

1.5.1 Attributes and Measurements

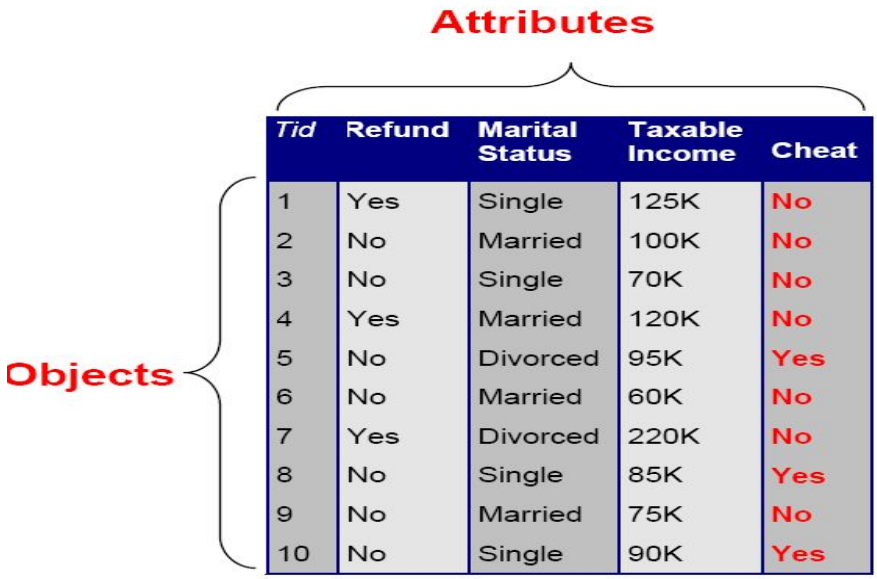
An attribute is a property or characteristic of an object. Attribute is also known as variable, field, characteristic, or feature. Examples: eye color of a person, temperature, etc. A collection of attributes describe an object.

Attribute Values: Attribute values are numbers or symbols assigned to an attribute. Distinction between attributes and attribute values– Same attribute can be mapped to different attribute values. Example: height can be measured in feet or meters.

The way you measure an attribute is somewhat may not match the attributes properties.



– Different attributes can be mapped to the same set of values. Example: Attribute values for ID and age are integers. But properties of attribute values can be different, ID has no limit but age has a maximum and minimum value.



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

The types of an attribute

A simple way to specify the type of an attribute is to identify the properties of numbers that correspond to underlying properties of the attribute.

- Properties of Attribute Values
  The type of an attribute depends on which of the following properties it possesses:
  – Distinctness: = ≠
  – Order: < >
  – Addition: + -
  – Multiplication: * /

There are different types of attributes
– Nominal
Examples: ID numbers, eye color, zip codes
– Ordinal
Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
– Interval
Examples: calendar dates, temperatures in Celsius or Fahrenheit.
– Ratio
Examples: temperature in Kelvin, length, time, counts

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. (=, ≠)	zip codes, employee ID numbers, eye color, sex: {*male, female*}	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects. (<, >)	hardness of minerals, {*good, better, best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Attribute Level	Transformation	Comments
Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
Ordinal	An order preserving change of values, i.e., *new_value = f(old_value)* where *f* is a monotonic function.	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Interval	*new_value =a * old_value + b* where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
Ratio	*new_value = a * old_value*	Length can be measured in meters or feet.

1.5.2 Describing attributes by the number of values

 Discrete Attribute– Has only a finite or countably infinite set of values, examples: zip codes, counts, or the set of words in a collection of documents, often represented as integer variables. Binary attributes are a special case of discrete attributes

 Continuous Attribute– Has real numbers as attribute values, examples: temperature, height, or weight. Practically, real values can only be measured and represented using a finite number of digits. Continuous attributes are typically represented as floating-point variables.

 Asymmetric Attribute-only a non-zero attributes value which is different from other values.

1.6 Data Quality:

Preliminary investigation of the data to better understand its specific characteristics, it can help to answer some of the data mining questions

– To help in selecting pre-processing tools

– To help in selecting appropriate data mining algorithms

 Things to look at: Class balance, Dispersion of data attribute values, Skewness, outliers, missing values, attributes that vary together, Visualization tools are important, Histograms, box plots, scatter plots Many datasets have a discrete (binary) attribute class

 Data mining algorithms may give poor results due to class imbalance problem, Identify the problem in an initial phase.

General characteristics of data sets:

- Dimensionality: of a data set is the number of attributes that the objects in the data set possess. Curse of dimensionality refers to analyzing high dimensional data.
- Sparsity: data sets with asymmetric features like most attributes of an object with value 0; in some cases it may be with value non-zero.
- Resolution: it is possible to obtain different levels of resolution of the data.

Now there are varieties of data sets are there, let us discuss some of the following.

1. **Record**
   - **Data Matrix**
   - **Document Data**
   - **Transaction Data**

2. **Graph**
   - **World Wide Web**
   - **Molecular Structures**

3. **Ordered**
   - **Spatial Data**
   - **Temporal Data**
   - **Sequential Data**
   - **Genetic Sequence Data**

Record Data

Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Transaction or market basket Data

A special type of record data, where each transaction (record) involves a set of items. For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Transaction data is a collection of sets of items, but it can be viewed as a set of records whose fields are asymmetric attributes.

Transaction data can be represented as sparse data matrix: **market basket representation**

– Each record (line) represents a transaction

– Attributes are binary and asymmetric

Tid	*Bread*	*Coke*	*Milk*	*Beer*	*Diaper*
1	1	1	1	0	0
2	1	0	0	1	0
3	0	1	1	1	1
4	1	0	1	1	1
5	0	1	1	0	1

Data Matrix

An M*N matrix, where there are M rows, one for each object, and N columns, one for each attribute. This matrix is called a data matrix, which holds only numeric values to its cells.

 If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute

 Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

The Sparse Data Matrix

It is a special case of a data matrix in which the attributes are of the same type and are asymmetric; i.e. , only non-zero values are important.

Document Data

Each document becomes a `term' vector, each term is a component (attribute) of the vector, and the value of each component is the number of times the corresponding term occurs in the document.

Graph-based data

In general, the data can take many forms from a single, time-varying real number to a complex interconnection of entities and relationships. While graphs can represent this entire spectrum of data, they are typically used when relationships are crucial to the domain. Graph-based data mining is the extraction of novel and useful knowledge from a graph representation of data. Graph mining uses the natural structure of the application domain and mines directly over that structure. The most natural form of knowledge that can be extracted from graphs is also a graph. Therefore, the knowledge, sometimes referred to as patterns, mined from the data are typically expressed as graphs, which may be sub-graphs of the graphical data, or more abstract expressions of the trends reflected in the data. The need of mining structural data to uncover objects or concepts that relates objects (i.e., sub-graphs that represent associations of features) has increased in the past ten years, involves the automatic extraction of novel and useful knowledge from a graph representation of data. a graph-based knowledge discovery system that finds structural, relational patterns in data representing entities and relationships. This algorithm was the first proposal in the topic and has been largely extended through the years. It is able to develop graph shrinking as well as frequent substructure extraction and hierarchical conceptual clustering.

A graph is a pair G = (V, E) where V is a set of vertices and E is a set of edges. Edges connect one vertices to another and can be represented as a pair of vertices. Typically each edge in a graph is given a label. Edges can also be associated with a weight.

We denote the vertex set of a graph g by V (g) and the edge set by E(g). A label function, L, maps a vertex or an edge to a label. A graph g is a sub-graph of another graph g' if there exists a sub-graph isomorphism from g to g'. (Frequent Graph) Given a labeled graph dataset, D = {G1, G2, . . . , Gn}, support (g) [or frequency(g)] is the percentage (or number) of graphs in D where g is a sub-graph. A frequent (sub) graph is a graph whose support is no less than a minimum support threshold, min support.

Spatial data

Also known as *geospatial data* or *geographic information* it is the data or information that identifies the geographic location of features and boundaries on Earth, such as natural or constructed features, oceans, and more. Spatial data is usually stored as coordinates and topology, and is data that can be mapped. Spatial data is often accessed, manipulated or analyzed through Geographic Information Systems (GIS).

Measurements in spatial data types: In the planar, or flat-earth, system, measurements of distances and areas are given in the same unit of measurement as coordinates. Using the geometry data type, the distance between (2, 2) and (5, 6) is 5 units, regardless of the units used.

In the ellipsoidal or round-earth system, coordinates are given in degrees of latitude and longitude. However, lengths and areas are usually measured in meters and square meters, though the measurement may depend on the spatial reference identifier (SRID) of the geography instance. The most common unit of measurement for the geography data type is meters.

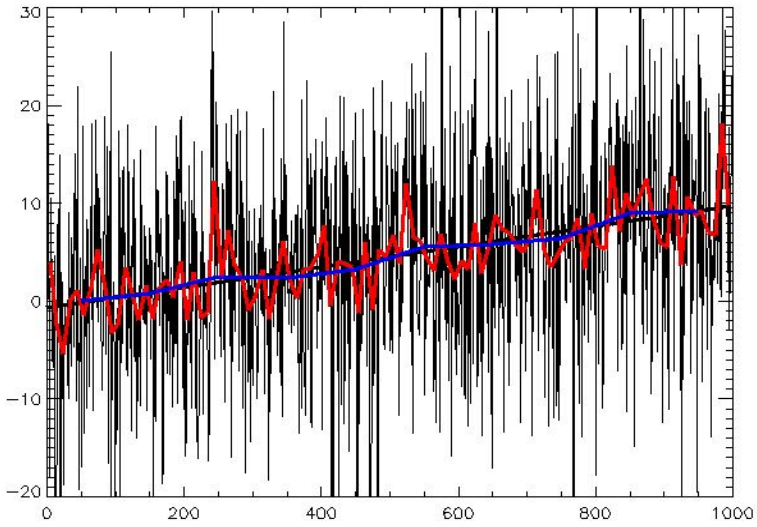
Orientation of spatial data: In the planar system, the ring orientation of a polygon is not an important factor. For example, a polygon described by ((0, 0), (10, 0), (0, 20), (0, 0)) is the same as a polygon described by ((0, 0), (0, 20), (10, 0), (0, 0)). The OGC Simple Features for SQL Specification does not dictate a ring ordering, and SQL Server does not enforce ring ordering.

Time Series Data

A time series is a sequence of observations which are ordered in time (or space). If observations are made on some phenomenon throughout time, it is most sensible to display the data in the order in which they arose, particularly since successive observations will probably be dependent. Time series are best displayed in a scatter plot. The series value X is plotted on the vertical axis and time t on the horizontal axis. Time is called the independent variable (in this case however, something over which you have little control). There are two kinds of time series data:

1. Continuous, where we have an observation at every instant of time, e.g. lie detectors, electrocardiograms. We denote this using observation X at time t, X(t).

2. Discrete, where we have an observation at (usually regularly) spaced intervals. We denote this as Xt.



Examples

Economics - weekly share prices, monthly profits

Meteorology - daily rainfall, wind speed, temperature

Sociology - crime figures (number of arrests, etc), employment figures

Sequence Data

Sequences are fundamental to modeling the three primary medium of human communication: speech, handwriting and language. They are the primary data types in several sensor and monitoring applications. Mining models for network intrusion detection view data as sequences of TCP/IP packets. Text information extraction systems model the input text as a sequence of words and delimiters. Customer data mining applications profile buying habits of customers as a sequence of items purchased. In computational biology, DNA, RNA and protein data are all best modeled as sequences.

A sequence is an ordered set of pairs (t1 x1) . . . (tn xn) where ti denotes an ordered attribute like time (ti−1 _ ti) and xi is an element value. The length n of sequences in a database is typically variable. Often the first attribute is not explicitly specified and the order of the elements is implicit in the position of the element. Thus, a sequence x can be written as x1 . . . xn. The elements of a sequence are allowed to be of many different types. When xi is a real number, we get a time series. Examples of such sequences abound — stock prices along time, temperature measurements obtained from a monitoring instrument in a plant or day to day

carbon monoxide levels in the atmosphere. When si is of discrete or symbolic type we have a categorical sequence.

1.7 Data quality

Data mining focuses on (1) the detection and correction of data quality problems (2) the use of algorithms that can tolerate poor data quality. Data are of high quality "if they are fit for their intended uses in operations, decision making and planning" (J. M. Juran). Alternatively, the data are deemed of high quality if they correctly represent the real-world construct to which they refer. Furthermore, apart from these definitions, as data volume increases, the question of *internal consistency* within data becomes paramount, regardless of fitness for use for any external purpose, e.g. a person's age and birth date may conflict within different parts of a database. The first views can often be in disagreement, even about the same set of data used for the same purpose.

Definitions are:

- Data quality: The processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria.
- Data exhibited by the data in relation to the portrayal of the actual scenario.
- The state of completeness, validity, consistency, timeliness and accuracy that makes data appropriate for a specific use.

Data quality aspects: Data size, complexity, sources, types and formats Data processing issues, techniques and measures *We are drowning in data, but starving of knowledge* (Jiawei Han).

Dirty data

What does *dirty data* mean?

Incomplete data(missing attributes, missing attribute values, only aggregated data, etc.) Inconsistent data (different coding schemes and formats, impossible values or out-of-range values), Noisy data (containing errors and typographical variations, outliers, not accurate values)

Data quality is a perception or an assessment of data's fitness to serve its purpose in a given context.

Aspects of data quality include:

- Accuracy
- Completeness
- Update status
- Relevance
- Consistency across data sources
- Reliability
- Appropriate presentation
- Accessibility

1.7.1 Measurement and data collection issues

Just think about the statement below" a person has a height of 2 meters, but weighs only 2kg`s ".

This data is inconsistence. So it is unrealistic to expect that data will be perfect.

Measurement error refers to any problem resulting from the measurement process. The numerical difference between measured value to the actual value is called as an **error**. Both of these errors can be random or systematic.

Noise and artifacts

Noise is the random component of a measurement error. It may involve the distortion of a value or the addition of spurious objects. Data Mining uses some robust algorithms to produce acceptable results even when noise is present.

Data errors may be the result of a more deterministic phenomenon called as artifacts.

Precision, Bias, and Accuracy

The quality of measurement process and the resulting data are measured by *Precision* and *Bias.* Accuracy refers to the degree of measurement error in data.

Outliers

Missing Values

It is not unusual for an object to be missed its attributes. In some cases information is not collected properly. Example application forms , web page forms.

Strategies for dealing with missing data are as follows:

- Eliminate data objects or attributes with missing values.
- Estimate missing values
- Ignore the missing values during analysis

Inconsistent values

Suppose consider a city like kengeri which is having zipcode 560060, if the user will give some other value for this locality then we can say that inconsistent value is present.

Duplicate data

Sometimes Data set contain same object more than once then it is called duplicate data. To detect and eliminate such a duplicate data two main issues are addressed here; first, if there are two objects that actually represent a single object, second the values of corresponding attributes may differ.

Issues related to applications are timelines of the data, knowledge about the data and relevance of the data.

Recommended Questions:

Q1. What is data mining? How it is different from database management system?

Q2. What are different challenges that motivated the development of DM?

Q3. Explain various DM tasks.

Q4. What are various DM functionalities? Explain

Q5. Differentiate between characterization and discrimination.

Q6. Discuss whether each of the following activities is a data mining task:

(a) Dividing the customers of a company according to their gender.

(b) Dividing the customers of a company according to their profitability.

(c) Extracting the frequencies of a sound wave.

(d) Monitoring the heart rate of a patient for abnormalities.

(e) Predicting the future stock price of a company using historical records.

(f) Computing the total sales of a company.

Q7. Suppose that you are employed as a data mining consultant for an Internet search engine company. Describe how data mining can help the company by giving a specific examples of how technique such as clustering, classification, association rule mining and anomaly detection can be applied.

Unit – 2

Data – 2: Data Pre-processing; Measures of Similarity and Dissimilarity

Text Notess:

Data Mining – Concepts and Techniques - Jiawei Han and Micheline Kamber, 2nd Edition, Morgan Kaufmann, 2006.

UNIT – 2

DATA – 2

2.1 Data Pre-processing

The data you wish to analyze by data mining techniques are **incomplete** (lacking attribute values

or certain attributes of interest, or containing only aggregate data), **noisy** (containing errors, or *outlier* values that deviate from the expected), and **inconsistent** (e.g., containing discrepancies in the department codes used to categorize items). Incomplete, noisy, and inconsistent data are commonplace properties of large real world databases and data warehouses. Incomplete data can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Other data may not be included simply because it was not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions. Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the history or modifications to the data may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.

Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies. If users believe the data are dirty, they are unlikely to trust the results of any data mining that has been applied to it.

Data Preprocessing

What preprocessing step can or should we apply to the data to make it more suitable for data mining?

– Aggregation

– Sampling

– Dimensionality Reduction

– Feature Subset Selection

– Feature Creation

– Discretization and Binarization

– Attribute Transformation

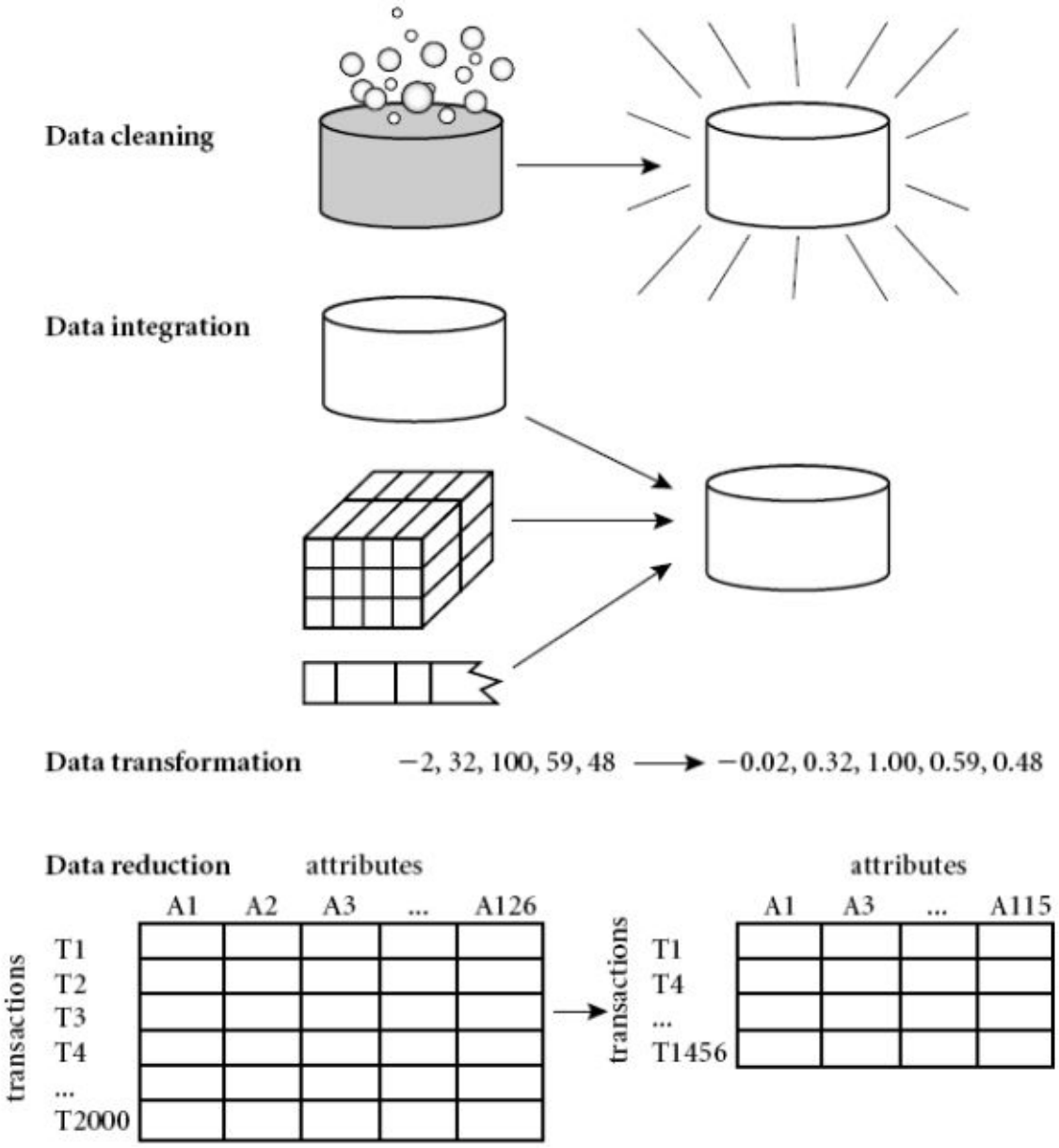


Figure 2.1 Forms of data preprocessing.

Data transformation operations, such as normalization and aggregation, are additional data preprocessing procedures that would contribute toward the success of the mining process. Data reduction obtains a reduced representation of the data set that is much smaller in volume, yet produces the same (or almost the same) analytical results. There are a number of strategies for data reduction. These include *data aggregation* (e.g., building a data cube), *attribute subset selection* (e.g removing irrelevant attributes through correlation analysis),

dimensionality reduction (e.g., using encoding schemes such as minimum length encoding or wavelets), and *numerosity reduction* (e.g., "replacing" the data by alternative, smaller representations such as clusters or parametric models).

a) Aggregation

• Aggregation refers to combing two or more attributes (or objects) into a single attribute (or object)

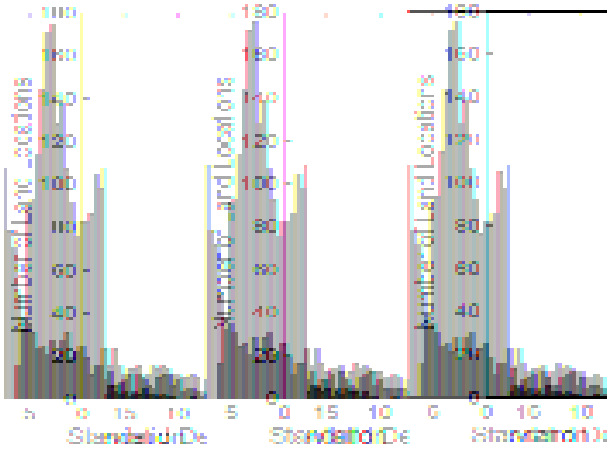
• For example, merging daily sales figures to obtain monthly sales figures

• Why aggregation?

– Data reduction

– If done properly, aggregation can act as scope or scale, providing a high level view of data instead of a low level view. Behavior of group of objects in more stable than that of individual objects

•The aggregate quantities have less variability than the individual objects being aggregated



Standard Deviation of Average Monthly Precipitation

b) Sampling

• Sampling is the process of understanding characteristics of data or models based on a subset of the original data. It is used extensively in all aspects of data exploration and mining

•Why sampling?

– Obtaining the entire set of "data of interest" is too expensive or time consuming

– Obtaining the entire set of data may not be necessary (and hence a waste of resources) Representative Sample

• A sample is representative for a particular operation if it results in approximately the same outcome as if the entire data set was used

• A sample that may be representative for one operation, may not be representative for another operation

– For example, a sample may be representative for histogram along one dimension but may not be good enough for correlation between two dimensions

Sampling Approaches

•Simple Random Sampling

There is an equal probability of selecting any particular item.

– Sampling without replacement: Once an item is selected, it is removed from the population for obtaining future samples

– Sampling with replacement: Selected item is not removed from the population for obtaining future samples

• Stratified Sampling

– When subpopulations vary considerably, it is advantageous to sample each subpopulation (stratum) independently

– Stratification is the process of grouping members of the population into relatively homogeneous subgroups before sampling

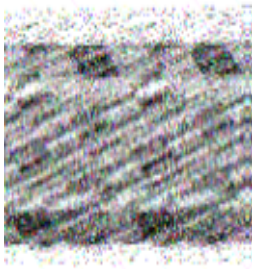
– The strata should be mutually exclusive : every element in the population must be assigned to only one stratum. The strata should also be collectively exhaustive : no population element can be excluded – Then random sampling is applied within each stratum. This often improves the representative-ness of the sample by reducing sampling error

Sample Size

• Even if proper sampling technique is known, it is important to choose proper sample size

• Larger sample sizes increase the probability that a sample will be representative, but also eliminate much of the advantage of sampling

•With smaller sample size, patterns may be missed or erroneous patters detected



8000 points

2000 points

500 points

Sample size required to obtain at least one sample from each group

C) Dimensionality Reduction

• Curse of dimensionality: Data analysis becomes significantly harder as the dimensionality of the data increases

Determining dimensions (or combinations of dimensions) that are important for modeling

• Why dimensionality reduction?

– Many data mining algorithms work better if the dimensionality of data (i.e. number of attributes) is lower

– Allows the data to be more easily visualized

– If dimensionality reduction eliminates irrelevant features or reduces noise, then quality of results may improve

– Can lead to a more understandable model Redundant features duplicate much or all of the information contained in one or more attributes

– The purchase price of product and the sales tax paid contain the same information

• Irrelevant features contain no information that is useful for data mining task at hand

– Student ID numbers would be irrelevant to the task of predicting their GPA

Dimensionality Reduction using Principal Component Analysis (PCA)

Find the principal directions in the data, and use them to reduce the number of dimensions of the set by representing the data in linear combinations of the principal components. Works best for multivariate data. Finds the m < d eigen-vectors of the covariance matrix with the

largest eigen-values. These eigen-vectors are the principal components. Decompose the data in these principal components and thus obtain as more concise data set.

Caution1: Depends on the normalization of the data!. Ideal for data with equal units.

Caution2: works only in linear relations between the parameters

Caution3: valuable information can be lost in pca

• Some of the most common approaches for dimensionality reduction, particularly for continuous data, use a linear or non-linear projection of data from a high dimensional space to a lower dimensional space

• PCA is a linear algebra technique for continuous attributes that finds new attributes (principal

components) that:

– Are linear combinations of original attributes

– Are orthogonal to each other

– Capture the maximum amount of variation in data

d) Feature Subset Selection

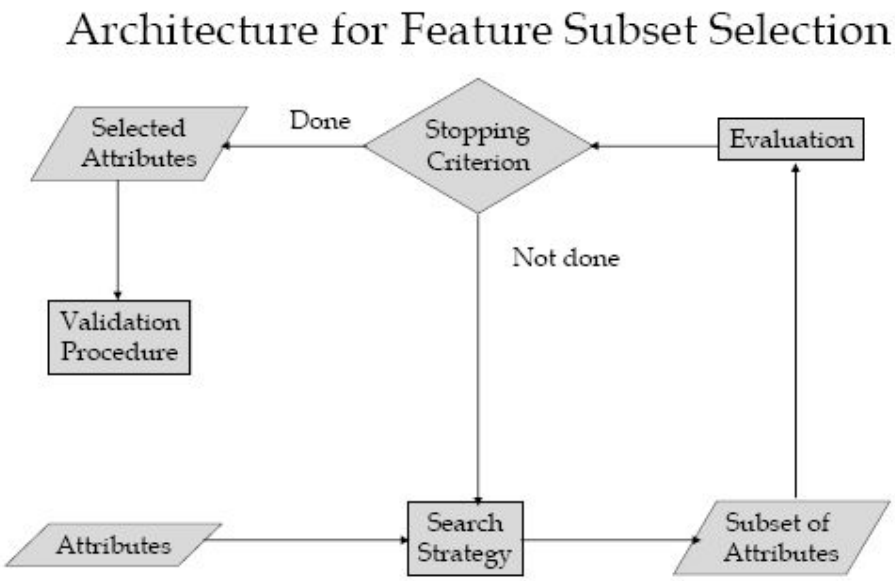
• There are three standard approaches to feature selection:

– Embedded approaches: Feature selection occurs naturally as part of the data mining algorithm

– Filter approaches: Features are selected before the data mining algorithm is run

– Wrapper approaches: Use the target data mining algorithm as a black box to find the best subset of attributes (typically without enumerating all subsets

Feature weighting is another process where more important attribute assigned with high value and less important attribute assigned with low value.



Flowchart of a feature subset selection process

e) Feature Creation

it is possible to create new set of attributes from old set. The new set of attributes are smaller than the number of original attributes, allowing us to reap all the previously described benefits.

This can be done by 3 methodologies: Feature extraction, mapping the data to a new space, and feature construction.

Feature Extraction: the creation of a new set of attributes from the original raw data is known as feature extraction.

Descriptive Data Summarization

For data preprocessing to be successful, it is essential to have an overall picture of your data. Descriptive data summarization techniques can be used to identify the typical properties of your data and highlight which data values should be treated as noise or outliers.

a) Measuring the Central Tendency

In this section, we look at various ways to measure the central tendency of data. The most common and most effective numerical measure of the "center" of a set of data is the *(arithmetic) mean*.

Let $x1,x2,\ldots xN$ be a set of N values or observations, such as for some attribute, like *salary*. The mean of this set of values is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}.$$

This corresponds to the built-in aggregate function, *average* (avg() in SQL), provided in relational database systems.

A distributive measure is a measure (i.e., function) that can be computed for a given data set by

partitioning the data into smaller subsets, computing the measure for each subset, and then merging the results in order to arrive at the measure's value for the original (entire) data set. Both sum() and count() are distributive measures because they can be computed in this manner. Other examples include max() and min(). An algebraic measure is a measure that can be computed by applying an algebraic function to one or more distributive measures. Hence, *average* (or mean()) is an algebraic measure because it can be computed by sum()/count().When computing data cubes, sum() and count() are typically saved in precomputation. Thus, the derivation of *average* for data cubes is straightforward. Sometimes, each value *xi* in a set may be associated with a weight *wi*, for *i* = 1…..*N*. The weights reflect the significance, importance, or occurrence frequency attached to their respective values. In this case, we can compute.

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_N x_N}{w_1 + w_2 + \cdots + w_N}.$$

This is called the weighted arithmetic mean or the weighted average. Note that the weighted average is another example of an algebraic measure.

A holistic measure is a measure that must be computed on the entire data set as a whole. It cannot be computed by partitioning the given data into subsets and merging the values obtained for the measure in each subset. The median is an example of a holistic measure.

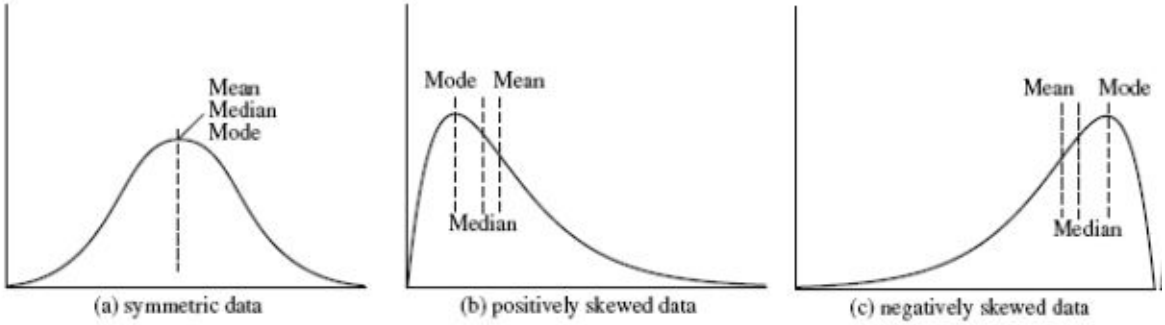


Figure 2.2 Mean, median, and mode of symmetric versus positively and negatively skewed data.

b) Measuring the Dispersion of Data

The degree to which numerical data tend to spread is called the dispersion, or variance of the data. The most common measures of data dispersion are *range*, the *five-number summary* (based on *quartiles*), the *interquartile range*, and the *standard deviation*. Boxplots can be plotted based on the five number summary and are a useful tool for identifying outliers.

Range, Quartiles, Outliers, and Boxplots

Let $x1;x2; : : : ;xN$ be a set of observations for some attribute. The range of the set is the difference between the largest (max()) and smallest (min()) values. For the remainder of this section, let's assume that the data are sorted in increasing numerical order.

The most commonly used percentiles other than the median are quartiles. The first quartile, denoted by $Q1$, is the 25th percentile; the third quartile, denoted by $Q3$, is the 75th percentile. The quartiles, including the median, give some indication of the center, spread, and shape of a distribution. The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the **interquartile range** (*IQR*) and is defined as $IQR= Q3-Q1$.

The five-number summary of a distribution consists of the median, the quartiles $Q1$ and $Q3$, and the smallest and largest individual observations, written in the order ***Minimum, Q1, Median, Q3, Maximum***: Boxplots are a popular way of visualizing a distribution. A boxplot incorporates the five-number summary as follows:

· Typically, the ends of the box are at the quartiles, so that the box length is the interquartile range, *IQR*.

· The median is marked by a line within the box.

· Two lines (called *whiskers*) outside the box extend to the smallest (*Minimum*) and largest (*Maximum*) observations.

Variance and Standard Deviation

The variance of N observations, $x1;x2; : : : ;xN$, is where x is the mean value of the observations, as defined in Equation (2.1).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N}\left[\sum x_i^2 - \frac{1}{N}(\sum x_i)^2\right],$$

The standard deviation, s, of the observations is the square root of the variance, s2.

where x is the mean value of the observations, as defined in Equation (2.1). The standard deviation, s, of the observations is the square root of the variance, s2.

The basic properties of the standard deviation, s, as a measure of spread are:

· σ measures spread about the mean and should be used only when the mean is chosen as the measure of center.

· σ=0 only when there is no spread, that is, when all observations have the same value. Otherwise

σ > 0.

The variance and standard deviation are algebraic measures because they can be computed from distributive measures.

f). Standardization or Mean Removal and Variance Scaling

Standardization of datasets is a **common requirement for many machine learning estimators** implemented in the scikit: they might behave badly if the individual feature do not more or less look like standard normally distributed data: Gaussian with **zero mean and unit variance**.

In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.

For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the l1 and l2 regularizers of linear models)

assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger that others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

The function scale provides a quick and easy way to perform this operation on a single array-like dataset:

```
>>> from sklearn import preprocessing
>>> X = [[ 1., -1.,  2.],
...      [ 2.,  0.,  0.],
...      [ 0.,  1., -1.]]
>>> X_scaled = preprocessing.scale(X)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

Normalization

Normalization is the process of **scaling individual samples to have unit norm**. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

This assumption is the base of the Vector Space Model often used in text classification and clustering contexts. The function normalize provides a quick and easy way to perform this operation on a single array-like dataset, either using the l1 or l2 norms.

Binarization

Feature binarization is the process of **thresholding numerical features to get boolean values**. This can be useful for downsteam probabilistic estimators that make assumption that the input data is distributed according to a multi-variate Bernoulli distribution. For instance, this is the case for the most common class of (Restricted) Boltzmann Machines.

It is also commmon among the text processing community to use binary feature values (probably to simplify the probabilistic reasoning) even if normalized counts (a.k.a. term frequencies) or TF-IDF valued features often perform slightly better in practice.

2.2 Measures of Similarity

Similarity Measurement

Similarity metric is the basic measurement and used by a number of data ming algorithms. It measures the similarity or dissimilarity between two data objects which have one or multiple attributes. Informally, the similarity is a numerical measure of the degree to which the two objects are alike. It is usually non-negative and are often between 0 and 1, where 0 means no similarity, and 1 means complete similarity. [1]

Considering different data type with a number of attributes, it is important to use the appropriate similarity metric to well measure the proximity between two objects. For example, euclidean distance and correlation are useful for dense data such as time series or two-dimensional points. Jaccard and cosine similarity measures are useful for sparse data like documents, or binary data. **Euclidean Distance**

Euclidean Distance between two points is given by Minkowski distance metric. It can be used in one-, tow-, or higher-dimensional space. The formula of Euclidean distance is as following.

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

where n is the number of dimensions. It measures the numerial difference for each corresponding attributes of point p and point q. Then it combines the square of differencies in each dimension into an overal distance. Here is an python example of calculating Euclidean distance of two data objects. This function calculates the distance for two person data object. It normalize the similarity score to a value between 0 and 1, where a value of 1 means that two people have identical preference, a value of 0 means that two people do not have common preference.

Jaccard Coefficient

Jaccard coefficient is often used to measure data objects consisting of asymmetric binary attributes. The asymmetric binary attributes have two values , 1 indicates present present and 0 indicates not present. Most of the attributes of the object will have the similiar value.

The Jaccard coefficient is given by the following equation:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}.$$

Where,

M11 represents the total number of attributes where object A and object B both have a value of 1.

M01 represents the total number of attributes where the attribute of A is 0 and the attribute of B is 1.

M10 represents the total number of attributes where the attribute of A is 1 and the attribute of B is 0.

The **Jaccard index**, also known as the **Jaccard similarity coefficient** (originally coined *coefficient de communauté* by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

The MinHash min-wise independent permutations locality sensitive hashing scheme may be used to efficiently compute an accurate estimate of the Jaccard similarity coefficient of pairs of sets, where each set is represented by a constant-sized signature derived from the minimum values of a hash function.

The **Jaccard distance**, which measures *dis*similarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union:

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

This distance is a proper metric.

Extended Jaccard Coefficient: It is used to compare documents. It measures the similarity of two sets by comparing the size of the overlap against the size of the two sets. Should the two sets have only binary attributes then it reduces to the Jaccard Coefficient. As with cosine, this is useful under the same data conditions and is well suited for market-basket data.

$$EJ(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \mathbf{x} \cdot \mathbf{y}}$$

Tanimoto Coefficient (Extended Jaccard Coefficient)

Various forms of functions described as Tanimoto Similarity and Tanimoto Distance occur in the literature and on the Internet. Most of these are synonyms for Jaccard Similarity and Jaccard Distance, but some are mathematically different. Many sources cite an unavailable IBM Technical Report as the seminal reference. The similarity ratio is equivalent to Jaccard similarity, but the distance function is *not* the same as Jaccard Distance.

Tanimoto coefficient is also known as extended Jaccard coefficient. It can used for handling the similarity of document data in text mining. In the case of binary attributes, it reduces to the Jaccard coefficient. Tanimoto coefficient is defined by the following equation:

$$T(A,B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B}.$$

Where A and B are two document vector object.

Tanimoto's Definitions of Similarity and Distance

In that paper, a "similarity ratio" is given over bitmaps, where each bit of a fixed-size array represents the presence or absence of a characteristic in the plant being modeled. The definition of the ratio is the number of common bits, divided by the number of bits set in either sample.

Presented in mathematical terms, if samples X and Y are bitmaps, X_i is the ith bit of X, and $\wedge$, $\vee$ are bitwise and, or operators respectively, then the similarity ratio T_s is

$$T_s(X,Y) = \frac{\sum_i (X_i \wedge Y_i)}{\sum_i (X_i \vee Y_i)}$$

If each sample is modelled instead as a set of attributes, this value is equal to the Jaccard Coefficient of the two sets. Jaccard is not cited in the paper, and it seems likely that the authors were not aware of it.

Tanimoto goes on to define a distance coefficient based on this ratio, defined over values with non-zero similarity:

$$T_d(X,Y) = -\log_2(T_s(X,Y))$$

This coefficient is, deliberately, not a distance metric. It is chosen to allow the possibility of two specimens, which are quite different to each other, to both be similar to a third. It is easy to construct an example which disproves the property of triangle inequality.

Cosine similarity

The cosine similarity is a measure of similarity of two non-binary vectors. The typical example is the document vector, where each attribute represents the frequency with which a particular word occurs in the document. Similar to sparse market transaction data, each document vector is sparse since it has relatively few non-zero attributes. Therefore, the cosine similarity ignores 0-0 matches like the Jaccard measure. The cosine similarity is defined by the following equation:

$$cos(A,B) = \frac{A \times B}{\|A\|\|B\|}$$

Pearson Correlation

The correlation coefficient is a measure of how well two sets of data fit on a straight line. Correlation is always in the range -1 to 1. A correlation of 1 (-1) means that x and y have a perfect positive (negative) linear relationship. If the correlation is 0, then there is no linear relationship between the attributes of the two data objects. However, the two data objects might have non-linear relationships.

Pearson correlation is defined by the following equation. x and y represents two data objects.

$$corr(x,y) = \frac{covariance(x,y)}{standard\_deviation(x) \times standard\_deviation(y)}$$

It measures the strength and the direction of the linear relationship between two variables. The value is always between [-1; 1] where 1 is strong positive relation, 0 is no relation and -1 is a strong negative correlation. It is the most widely used correlation coefficient and works very well when all is linear but not with curvilinear.

$$p = \frac{cov(x,y)}{\sigma_x \cdot \sigma_y}$$

Here is an python example of calculating Pearson Correlation of two data objects. The code for the Pearson correlation score first finds the items rated by both critics. It then calculates the sums and the sum of the squares of the ratings for the two critics, and calculates the sum of the products of their ratings. Finally, it uses these results to calculate the Pearson correlation coefficient, shown in bold in the code below. Unlike the distance metric, this formula is not very intuitive, but it does tell you how much the variables change together divided by the product of how much they vary individually.

Recommended Questions:

1. Why preprocessing of the data is required before applying DM techniques?
2. What are the different types of attributes?
3. What is the difference between symmetric and asymmetric attributes?
4. Explain general characteristics of data sets?
5. What are the different types of data sets that one come across while applying Dtechniques?
6. What is data cleaning?
7. Discuss various aspects of Data quality?
8. Differentiate between precision and accuracy?
9. How missing values are handled while cleaning the data?
10. Explain various data preprocessing strategies?
11. Write a short note on the following:
    (a) Dimensionality reduction

(b) aggregation

(c) sampling

(d) feature selection

(e) variable transformation

12. Explain various distances used for measuring dissimilarity between the objects.

13. Under what condition a distance becomes a metric.

14. Discuss the issues related to proximity calculation for the following vectors, x and y, calculate the indicated similarity or distance measures.

    (a) x=(1,1,1,1), y=(2,2,2,2) cosine, correlation, Euclidean

    (b) x=(0,1,0,1) y=(1,0,1,0) cosine, correlation, Euclidean and Jaccard.

15. Explain why computing proximity between two attributes is often simpler than computing the similarity between two objects.

Unit – 3

Classification: Preliminaries; General approach to solving a classification problem; Decision tree induction; Rule-based classifier; Nearest-neighbour classifier.

Text Notess:

Data Mining – Concepts and Techniques - Jiawei Han and Micheline Kamber, 2nd Edition, Morgan Kaufmann, 2006.

UNIT – 3
CLASSIFICATION

3.1 Preliminaries

The input data for a classification task is a collection of records. Each record, also known as an instance or example, is characterized by a tuple (x, y), where x is the attribute set and y is a special attribute, designated as the class label. sample data set used for classifying vertebrates into one of the following categories: mammal, bird, fish, reptile, or amphibian. The attribute set includes properties of a vertebrate such as its body temperature, skin cover, method of reproduction ability to fly, and ability to live in water. the attribute set can also contain continuous features. The class label, on the other hand, must be a discrete attribute. This is a key characteristic that distinguishes classification from regression, a predictive modelling task in which y is a continuous attribute. .

Definition 3.1 (Classification). Classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y. The target function is also known informally as a classification model.A classification model is useful for the following purposes.

Descriptive Modeling

A classification model can serve as an explanatory tool to distinguish between objects of different classes. For example, it would be useful for both biologists and others to have a descriptive model.

Predictive Modeling

A classification model can also be used to predict the class label of unknown records. As a classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record. Suppose we are given the following characteristics of a creature known as a gila monster: Classification techniques are most suited for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (e.g., to classify a person as a member of high-, medium-, or low-income group) because they do not consider the implicit order among the categories. Other forms of relationships, such as the subclass–

super class relationships among categories

3.2 General approach to solving a classification problem

A classification technique (or classifier) is a systematic approach to building classification models from an input data set. Examples include decision tree classifiers, rule-based classifiers, neural networks, support vector machines and na¨ıve Bayes classifiers. Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.

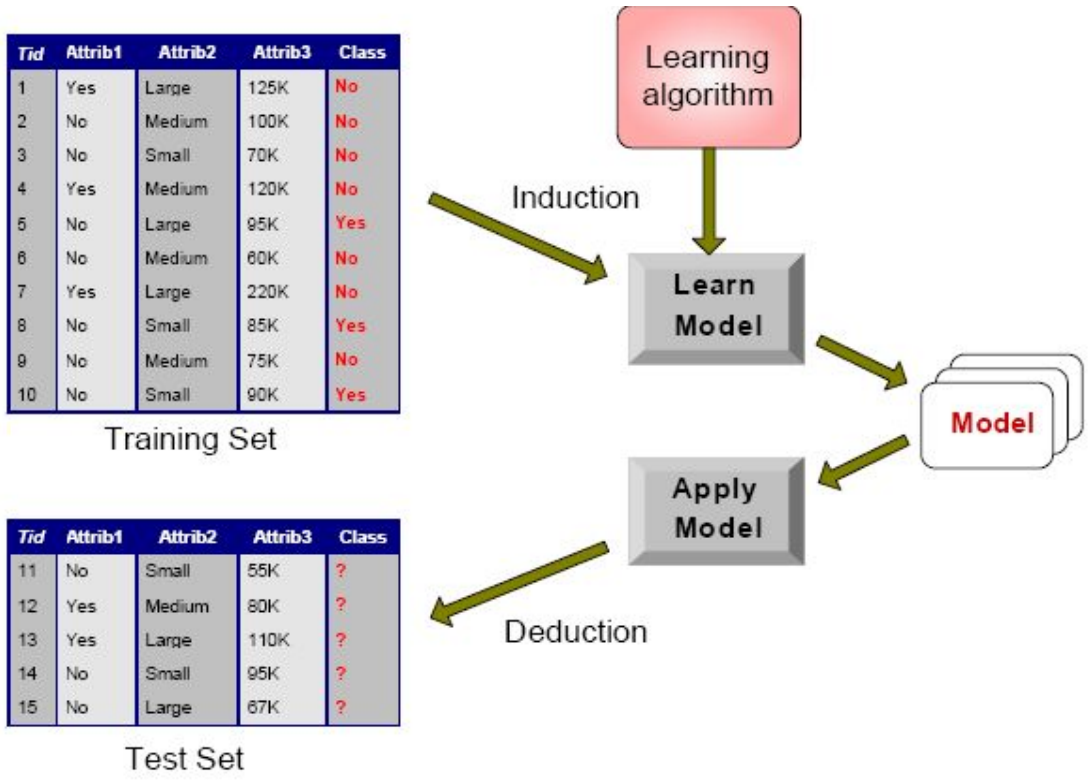


Figure 3.3. General approach for building a classification model.

3.3 Decision tree induction

3.3.1 How a Decision Tree Works

To illustrate how classification with a decision tree works, consider a simpler version of the vertebrate classification problem described in the previous section. Instead of classifying the vertebrates into five distinct groups of species, we assign them to two categories: mammals and non-mammals. Suppose a new species is discovered by scientists. How can we tell whether it is a mammal or a non-mammal? One approach is to pose a series of questions about the characteristics of the species. The first question we may ask is whether the species is cold- or warm-blooded. If it is cold-blooded, then it is definitely not a mammal. Otherwise, it is either a bird or a mammal. In the latter case, we need to ask a follow-up question: Do the females of the species give birth to their young? Those that do give birth are definitely mammals, while those that do not are likely to be non-mammals (with the exception of egg-laying mammals such as the platypus and spiny anteater) The previous example illustrates how we can solve a classification problem by asking a series of carefully crafted questions about the attributes of the test record. Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class label of the record. The series of questions and their possible answers can be organized in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges. Figure 4.4 shows the decision tree for the mammal classification problem. The tree has three types of nodes:

- **A root node** that has no incoming edges and zero or more outgoing edges.
- **Internal nodes,** each of which has exactly one incoming edge and two or more outgoing edges.
- **Leaf or terminal** nodes, each of which has exactly one incoming edge and no outgoing edges.

In a decision tree, each leaf node is assigned a class label. The non terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics. For example, the root node shown in Figure 4.4 uses the attribute Body.

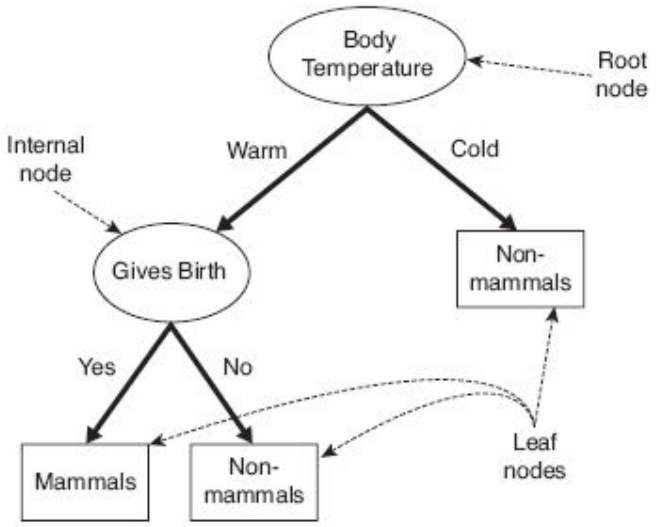


Figure 3.4. A decision tree for the mammal classification problem.

Temperature to separate warm-blooded from cold-blooded vertebrates. Since all cold-blooded vertebrates are non-mammals, a leaf node labeled Non-mammals is created as the right child of the root node. If the vertebrate is warm-blooded, a subsequent attribute, Gives Birth, is used to distinguish mammals from other warm-blooded creatures, which are mostly birds. Classifying a test record is straightforward once a decision tree has been constructed. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test.

This will lead us either to another internal node, for which a new test condition is applied, or to a leaf node. The class label associated with the leaf node is then assigned to the record. As an illustration, Figure 4.5 traces the path in the decision tree that is used to predict the class label of a flamingo. The path terminates at a leaf node labeled Non-mammals.

3.3.2 How to Build a Decision Tree

There are exponentially many decision trees that can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space. Nevertheless, efficient algorithms have been developed to induce a reasonably accurate, albeit suboptimal, decision tree in a reasonable amount of time.

These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data. One such algorithm is Hunt's algorithm, which is the basis of many existing decision tree induction algorithms, including ID3, C4.5, and CART.

This section presents a high-level discussion of Hunt's algorithm and illustrates some of its design issues.

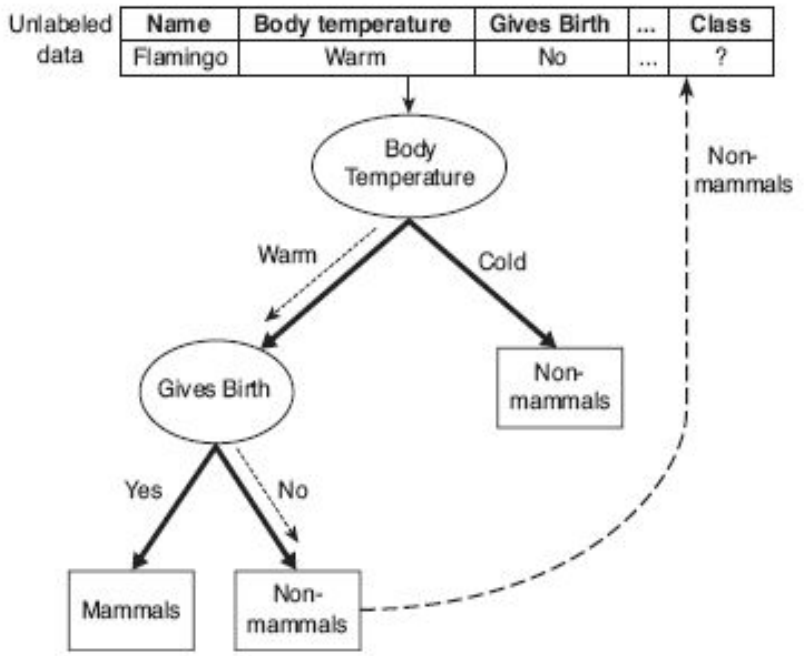


Figure 3.**5.** Classifying an unlabeled vertebrate.
The dashed lines represent the outcomes of applying various attribute test conditions on the unlabeled vertebrate. The vertebrate is eventually assigned to the Non-mammal class.

Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that are associated with node t and y = {y1, y2, . . . , yc} be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in Data belong to the same class yt, then t is a leaf node labeled as

y_t.

Step 2: If Data contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

3.4 Rule-based classifier

In this section, we look at rule-based classifiers, where the learned model is represented as a set of IF-THEN rules. We first examine how such rules are used for classification. We then study ways in which they can be generated, either froma decision tree or directly from the training data using a *sequential covering algorithm.*

3.4.1 Using IF-THEN Rules for Classification

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

IF *condition* THEN *conclusion*.

An example is rule *R*1,

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

The "IF"-part (or left-hand side) of a rule is known as the rule antecedent or precondition. The "THEN"-part (or right-hand side) is the rule consequent. In the rule antecedent, the condition consists of one or more *attribute tests* (such as *age = youth*, and *student = yes*) that are logically ANDed. The rule's consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). *R*1 can also be written as

R1: (*age = youth*) ^ (*student = yes*))(*buys computer = yes*).

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

A rule *R* can be assessed by its coverage and accuracy. Given a tuple, ***X***, from a class labeled data set,*D*, let *ncovers* be the number of tuples covered by *R*; *ncorrect* be the number of

tuples correctly classified by R; and jDj be the number of tuples in D. We can define the coverage and accuracy of R as

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

That is, a rule's coverage is the percentage of tuples that are covered by the rule (i.e., whose attribute values hold true for the rule's antecedent). For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.

3.4.2 Rule Extraction from a Decision Tree

Decision tree classifiers are a popular method of classification—it is easy to understand how decision trees work and they are known for their accuracy. Decision trees can become large and difficult to interpret. In this subsection, we look at how to build a rule based classifier by extracting IF-THEN rules from a decision tree. In comparison with a decision tree, the IF-THEN rules may be easier for humans to understand, particularly if the decision tree is very large.

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent ("IF" part). The leaf node holds the class prediction, forming the rule consequent ("THEN" part).

Example 3.4 Extracting classification rules from a decision tree. The decision tree of Figure 6.2 can

be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree.

$R1$: IF	$age = youth$	AND $student = no$	THEN $buys\_computer = no$
$R2$: IF	$age = youth$	AND $student = yes$	THEN $buys\_computer = yes$
$R3$: IF	$age = middle\_aged$		THEN $buys\_computer = yes$
$R4$: IF	$age = senior$	AND $credit\_rating = excellent$	THEN $buys\_computer = yes$
$R5$: IF	$age = senior$	AND $credit\_rating = fair$	THEN $buys\_computer = no$

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they are mutually exclusive and exhaustive. By *mutually exclusive*, this means that we cannot have rule conflicts here because no two rules will be

triggered for the same tuple. (We have one rule per leaf, and any tuple can map to only one leaf.) By *exhaustive*, there is one rule for each possible attribute-value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter—they are *unordered*.

Since we end up with one rule per leaf, the set of extracted rules is not much simpler than the corresponding decision tree! The extracted rules may be even more difficult to interpret than the original trees in some cases. As an example, Figure 6.7 showed decision trees that suffer from subtree repetition and replication. The resulting set of rules extracted can be large and difficult to follow, because some of the attribute tests may be irrelevant or redundant. So, the plot thickens. Although it is easy to extract rules from a decision tree, we may need to do some more work by pruning the resulting rule set.

3.4.3 Rule Induction Using a Sequential Covering Algorithm

IF-THEN rules can be extracted directly from the training data (i.e., without having to generate a decision tree first) using a sequential covering algorithm. The name comes from the notion that the rules are learned *sequentially* (one at a time), where each rule for a given class will ideally *cover* many of the tuples of that class (and hopefully none of the tuples of other classes). Sequential covering algorithms are the most widely used approach to mining disjunctive sets of classification rules, and form the topic of this subsection. Note that in a newer alternative approach, classification rules can be generated using *associative classification algorithms*, which search for attribute-value pairs that occur frequently in the data. These pairs may form association rules, which can be analyzed and used in classification. Since this latter approach is based on association rule mining (Chapter 5), we prefer to defer its treatment until later, in Section 6.8. There are many sequential covering algorithms. Popular variations include AQ, CN2, and the more recent, RIPPER. The general strategy is as follows. Rules are learned one at a time. Each time a rule is learned, the tuples covered by the rule are removed, and the process repeats on the remaining tuples. This sequential learning of rules is in contrast to decision tree induction. Because the path to each leaf in a decision tree corresponds to a rule, we can consider decision tree induction as learning a set of rules *simultaneously*.

A basic sequential covering algorithm is shown in Figure 6.12. Here, rules are learned for one class at a time. Ideally, when learning a rule for a class, *Ci*, we would like the rule to cover all (or many) of the training tuples of class *C* and none (or few) of the tuples from other classes.

In this way, the rules learned should be of high accuracy. The rules need not necessarily be of high coverage.

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.
Input: D, a data set class-labeled tuples;
Att vals, the set of all attributes and their possible values.
Output: A set of IF-THEN rules.
Method:
(1) Rule set = fg; // initial set of rules learned is empty
(2) for each class c do
(3) repeat
(4) Rule = Learn One Rule(D, Att vals, c);
(5) remove tuples covered by Rule from D;
(6) until terminating condition;
(7) Rule set = Rule set +Rule; // add new rule to rule set
(8) endfor
(9) return Rule Set;

This is because we can have more than one rule for a class, so that different rules may cover different tuples within the same class. The process continues until the terminating condition is met, such as when there are no more training tuples or the quality of a rule returned is below a user-specified threshold. The Learn One Rule procedure finds the "best" rule for the current class, given the current set of training tuples. "How are rules learned?" Typically, rules are grown in a general-to-specific manner .We can think of this as a beam search, where we start off with an empty rule and then gradually keep appending attribute tests to it. We append by adding the attribute test as a logical conjunct to the existing condition of the rule antecedent. Suppose our training set, D, consists of loan application data. Attributes regarding each applicant include their age, income, education level, residence, credit rating, and the term of the loan. The classifying attribute is loan decision, which indicates whether a
loan is accepted (considered safe) or rejected (considered risky). To learn a rule for the class "accept," we start off with the most general rule possible, that is, the condition of the rule antecedent is empty. The rule is:
IF THEN loan decision = accept.
We then consider each possible attribute test that may be added to the rule. These can be derived from the parameter Att vals, which contains a list of attributes with their associated values. For example, for an attribute-value pair (att, val), we can consider attribute tests such as att = val, att _ val, att > val, and so on. Typically, the training data will contain many

attributes, each of which may have several possible values. Finding an optimal rule set becomes computationally explosive. Instead, Learn One Rule adopts a greedy depth-first strategy. Each time it is faced with adding a new attribute test (conjunct) to the current rule, it picks the one that most improves the rule quality,

based on the training samples. We will say more about rule quality measures in a minute. For the moment, let's say we use rule accuracy as our quality measure. Getting back to our example with Figure 6.13, suppose Learn One Rule finds that the attribute test income = high best improves the accuracy of our current (empty) rule. We append it to the condition, so that the current rule becomes

IF income = high THEN loan decision = accept. Each time we add an attribute test to a rule, the resulting rule should cover more of the "accept" tuples. During the next iteration, we again consider the possible attribute tests and end up selecting credit rating = excellent. Our current rule grows to become

IF income = high AND credit rating = excellent THEN loan decision = accept.

The process repeats, where at each step, we continue to greedily grow rules until the resulting rule meets an acceptable quality level.

3.4.4 Rule Pruning

Learn One Rule does not employ a test set when evaluating rules. Assessments of rule quality as described above are made with tuples from the original training data. Such assessment is optimistic because the rules will likely overfit the data. That is, the rules may perform well on the training data, but less well on subsequent data. To compensate for this, we can prune the rules. A rule is pruned by removing a conjunct (attribute test). We choose to prune a rule, R, if the pruned version of R has greater quality, as assessed on an independent set of tuples. As in decision tree pruning, we

refer to this set as a pruning set. Various pruning strategies can be used, such as the pessimistic pruning approach described in the previous section. FOIL uses a simple yet effective method. Given a rule, R,

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg},$$

where pos and neg are the number of positive and negative tuples covered by R, respectively. This value will increase with the accuracy of R on a pruning set. Therefore, if the FOIL Prune value is higher for the pruned version of R, then we prune R. By convention, RIPPER starts

with the most recently added conjunct when considering pruning. Conjuncts are pruned one at a time as long as this results in an improvement.

Rule based classifier

Classify records by using a collection of "if…then…" rules

Rule: (*Condition*) → *y*

– Where *Condition* is a conjunction of attributes *y* is the class label

– *LHS*: rule antecedent or condition

– *RHS*: rule consequent

– Examples of classification rules:

(Blood Type=Warm) □ (Lay Eggs=Yes) → Birds

(Taxable Income < 50K) □ (Refund=Yes) → Evade=No

R1: (Give Birth = no) □ (Can Fly = yes) → Birds

R2: (Give Birth = no) □ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) □ (Blood Type = warm) → Mammals

R4: (Give Birth = no) □ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

Application of Rule-Based Classifier

A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule.

R1: (Give Birth = no) □ (Can Fly = yes) → Birds

R2: (Give Birth = no) □ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) □ (Blood Type = warm) → Mammals

R4: (Give Birth = no) □ (Can Fly = no) → Reptiles

R The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

Coverage of a rule: Fraction of records that satisfy the antecedent of a rule

Accuracy of a rule: Fraction of records satisfy both the antecedent and consequent of a rule table

Characteristics of Rule-Based Classifier

Mutually exclusive rules: Classifier contains mutually exclusive rules if the rules are independent of each other every record is covered by at most one rule.

Exhaustive rules: Classifier has exhaustive coverage if accounts for every possible combination of attribute values each record is covered by at least one rule.

3.6 Nearest-neighbour classifier

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuplewith training tuples that are similar to it. The training tuples are described by *n* attributes. Each tuple represents a point in an *n*-dimensional space. In this way all of the training tuples are stored in an *n*-dimensional pattern space. When given an unknown tuple, a ***k***-nearest-neighbor classifier searches the pattern space for the *k* training tuples that are closest to the unknown tuple. These *k* training tuples are the *k* "nearest neighbors" of the unknown tuple. "Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples, say, $\boldsymbol{X}1 = (x11, x12, : : : , x1n)$ and $\boldsymbol{X}2 = (x21, x22, : : : , x2n)$, is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple X1 and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count. Typically, we normalize the values of each attribute before using Euclid's Equation. This helps prevent attributes with initially large ranges (such as income) from outweighing attributes with initially smaller ranges (such as binary attributes). Min-max normalization, for example, can be used to transform a value v of a numeric attribute A to v0 in the range [0, 1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

where minA and maxA are the minimum and maximum values of attribute A. Chapter 2 describes other methods for data normalization as a form of data transformation. For k-

nearest-neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors. When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space. Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple. "But how can distance be computed for attributes that not numeric, but categorical, such as color?" The above discussion assumes that the attributes used to describe the tuples are all numeric. For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple X1 with that in tuple X2. If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0. If the two are different (e.g., tuple X1 is blue but tuple X2 is red), then the difference is considered to be 1. Other methods may incorporate more sophisticated schemes for differential grading (e.g., where a larger difference score is assigned, say, for blue and
white than for blue and black).

"What about missing values?" In general, if the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference. Suppose that each of the attributes have been mapped to the range [0, 1]. For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing. If A is numeric and missing fromboth tuples X1 and X2, then the difference is also taken to be 1.

"How can I determine a good value for k, the number of neighbors?" This can be determined experimentally. Starting with k = 1, we use a test set to estimate the error rate of the classifier. This process can be repeated each time by incrementing k to allow for one more neighbor. The k value that gives the minimum error rate may be selected. In general, the larger the number of training tuples is, the larger the value of k will be (so that classification and prediction decisions can be based on a larger portion of the stored tuples). As the number of training tuples approaches infinity and k =1, the error rate can be no worse then twice the Bayes error rate (the latter being the theoretical minimum).

Recommended Questions:

1. What is classification? Differentiate between classification and prediction.
2. Explain the process of building a classifier using diagram.
3. What is Decision Tree? How it is built? How this tree works?
4. Explain the methods for expressing attribute test conditions.
5. Explain the terms Entropy and Gini index in context of decision tree.
6. What is gain ratio?
7. Write the characteristics of Decision tree induction.
8. What is rule based classifier? Give an example.
9. How a rule based classifier works? How it is built?
10. Write some important characteristics of rule based classifiers.
11. What are NN classifiers? Explain some characteristics of it.

Unit – 4

Association Analysis – 1: Problem Definition; Frequent Item set generation; Rule Generation; Compact representation of frequent item sets; Alternative methods for generating frequent item sets.

Text Notess:

Data Mining – Concepts and Techniques - Jiawei Han and Micheline Kamber, 2nd Edition, Morgan Kaufmann, 2006.

UNIT – 4
ASSOCIATION ANALYSIS – 1

This chapter presents a methodology known as association analysis, which is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of association rules or sets of frequent items. For example, the following rule can be extracted from the data set shown in Table 4.1:

{Diapers} → {Beer}.

Table 4.1. An example of market basket transactions.

TID	ITEMS
1	{Bread, Milk}
2	{ Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

The rule suggests that a strong relationship exists between the sale of diapers and beer because many customers who buy diapers also buy beer. Retailers can use this type of rules to help them identify new opportunities for cross- selling their products to the customers.

4.1Problem Definition

This section reviews the basic terminology used in association analysis and presents a formal description of the task.

Binary Representation Market basket data can be represented in a binary format as shown in Table 4.2, where each row corresponds to a transaction and each column

corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable.

Table 4.2 A binary 0/1 representation of market basket data.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

This representation is perhaps a very simplistic view of real market basket data because it ignores certain important aspects of the data such as the quantity of items sold or the price paid to purchase them. Itemset and Support Count Let I = {i1,i2,. . .,id} be the set of all items in a market basket data and T = {t1, t2, . . . , tN } be the set of all transactions. Each transaction ti contains a subset of items chosen from I. In association analysis, a collection of zero or more items is termed an itemset. If an itemset contains k items, it is called a k-itemset. For instance, {Beer, Diapers, Milk} is an example of a 3-itemset. The null (or empty) set is an itemset that does not contain any items.

The transaction width is defined as the number of items present in a transaction. A transaction tj is said to contain an itemset X if X is a subset of tj. For example, the second transaction shown in Table 6.2 contains the item-set {Bread, Diapers} but not {Bread, Milk}. An important property of an itemset is its support count, which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, σ(X), for an itemset X can be stated as follows:

$$\sigma(X) = \left|\{t_i | X \subseteq t_i,\ t_i \in T\}\right|,$$

Where the symbol | · | denote the number of elements in a set. In the data set shown in Table 4.2, the support count for {Beer, Diapers, Milk} is equal to two because there are only two transactions that contain all three items.

Association Rule An association rule is an implication expression of the form X → Y , where X and Y are disjoint itemsets, i.e., X ∩ Y = 0. The strength of an association rule can be measured in terms of its support and confidence. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in *Y* appear in transactions that contain *X*. The formal definitions of these metrics are

Support s(X------->Y) = $\frac{\partial(XUY)}{N}$ **4.1**

Confidence C(X------>Y) = ∂(XUY)/∂(X) 4.2

Formulation of Association Rule Mining Problem The association rule mining problem can be formally stated as follows:

Definition 4.1 (Association Rule Discovery). Given a set of transactions T , find all the rules having support ≥ minsup and confidence ≥ minconf, where minsup and minconf are the corresponding support and confidence thresholds.

From Equation 4.2, notice that the support of a rule X -→ Y depends only on the support of its corresponding itemset, X ∩ Y . For example, the following rules have identical support because they involve items from the same itemset,

{Beer, Diapers, Milk}:

{Beer, Diapers} -→{Milk}, {Beer, Milk} -→{Diapers},

{Diapers, Milk} -→{Beer}, {Beer} -→{Diapers, Milk},

{Milk} -→{Beer,Diapers}, {Diapers} -→{Beer,Milk}.

If the itemset is infrequent, then all six candidate rules can be pruned immediately without our having to compute their confidence values. Therefore, a common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

1. Frequent Itemset Generation, whose objective is to find all the item-sets that satisfy the minsup threshold. These itemsets are called frequent itemsets.

2. Rule Generation, whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules.

The computational requirements for frequent itemset generation are generally more expensive than those of rule generation.

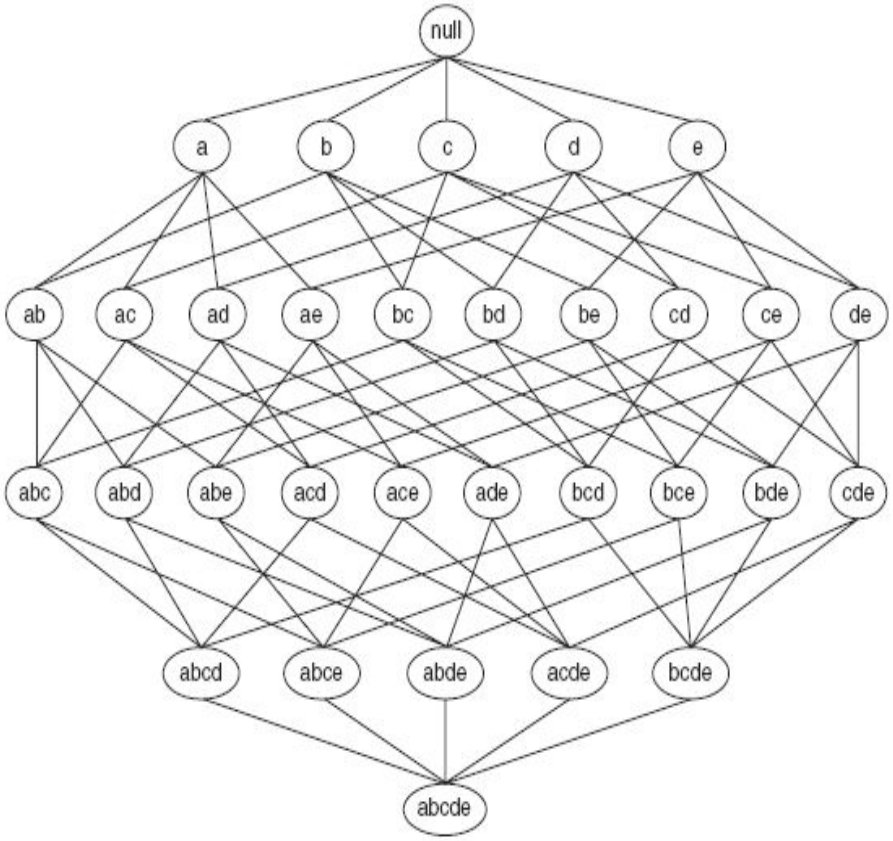


Figure 4.1. An itemset lattice.

4.2 Frequent Item set generation

A lattice structure can be used to enumerate the list of all possible itemsets. Figure 4.1 shows an itemset lattice for I = {a, b, c, d, e}. In general, a data set that contains k items can potentially generate up to 2k - 1 frequent itemsets, excluding the null set. Because k can be very large in many practical applications, the search space of itemsets that need to be explored is exponentially large.

A brute-force approach for finding frequent itemsets is to determine the support count for every candidate itemset in the lattice structure. To do this, we need to compare each

candidate against every transaction, an operation that is shown in Figure 4.2. If the candidate is contained in a transaction, its support count will be incremented. For example, the support for {Bread,Milk} is incremented three times because the itemset is contained in transactions 1, 4, and 5. Such an approach can be very expensive because it requires O(NMw) comparisons, where N is the number of transactions, M =2k - 1 is the number of candidate itemsets, and w is the maximum transaction width.

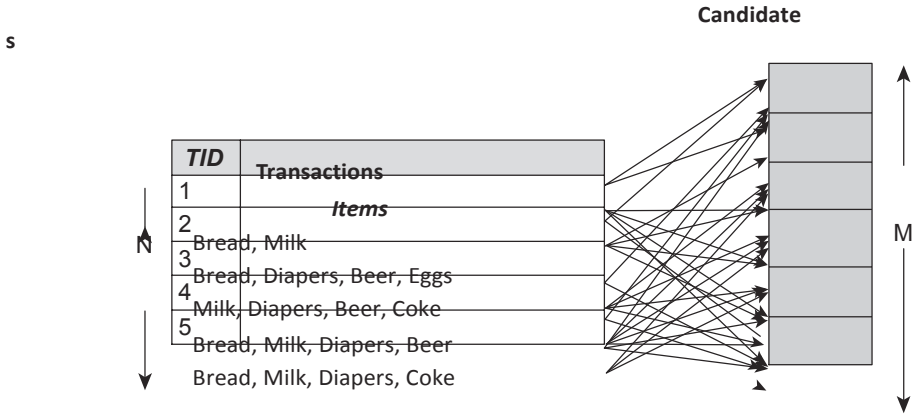


Figure 6.2. Counting the support of candidate itemsets.

There are several ways to reduce the computational complexity of frequent itemset generation.

1. Reduce the number of candidate itemsets (M). The Apriori principle, described in the next section, is an effective way to eliminate some of the candidate itemsets without counting their support values.

2. Reduce the number of comparisons. Instead of matching each candidate itemset against every transaction, we can reduce the number of comparisons by using more advanced data structures, either to store the candidate itemsets or to compress the data set.

4.2.1 The Apriori Principle

This section describes how the support measure helps to reduce the number of candidate itemsets explored during frequent itemset generation. The use of support for pruning candidate itemsets is guided by the following principle.

Theorem 4.1 (Apriori Principle). If an itemset is frequent, then all of its subsets must also be frequent. To illustrate the idea behind the Apriori principle, consider the itemset lattice shown in Figure 4.3. Suppose {c, d, e} is a frequent itemset. Clearly, any transaction that contains {c, d, e} must also contain its subsets, {c, d},{c, e}, {d, e}, {c}, {d}, and {e}. As a result, if {c, d, e} is frequent, then all subsets of {c, d, e} (i.e., the shaded itemsets in this figure) must also be frequent.

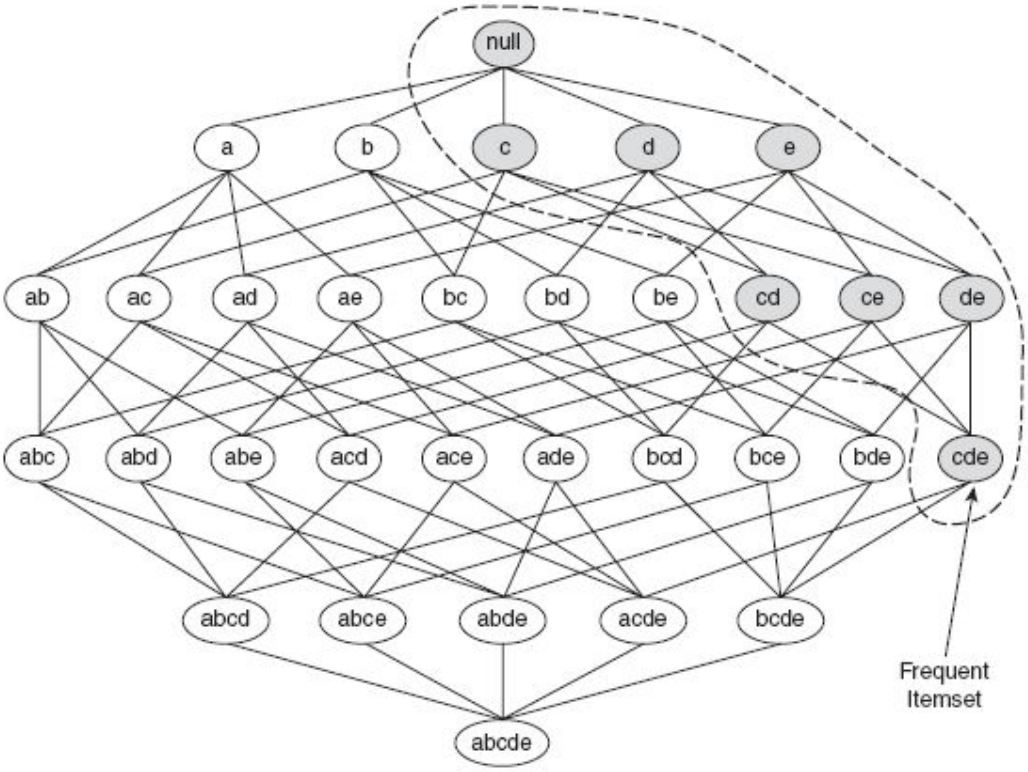


Figure 4.3. An illustration of the Apriori principle.

If {c, d, e} is frequent, then all subsets of this itemset are frequent.

Conversely, if an itemset such as {a, b} is infrequent, then all of its supersets must be infrequent too. As illustrated in Figure 6.4, the entire subgraph containing the supersets of {a, b} can be pruned immediately once {a, b} is found to be infrequent. This strategy of trimming the exponential search space based on the support measure is known as support-based pruning. Such a pruning strategy is made possible by a key property of the support measure, namely, that the support for an itemset never exceeds the support for its subsets. This property is also known as the anti-monotone property of the support measure.

Definition 4.2 (Monotonicity Property). Let I be a set of items, and J =2I be the power set of I. A measure f is monotone (or upward closed) if

$$\forall X, Y \in J : \ (X \subseteq Y) \longrightarrow f(X) \leq f(Y),$$

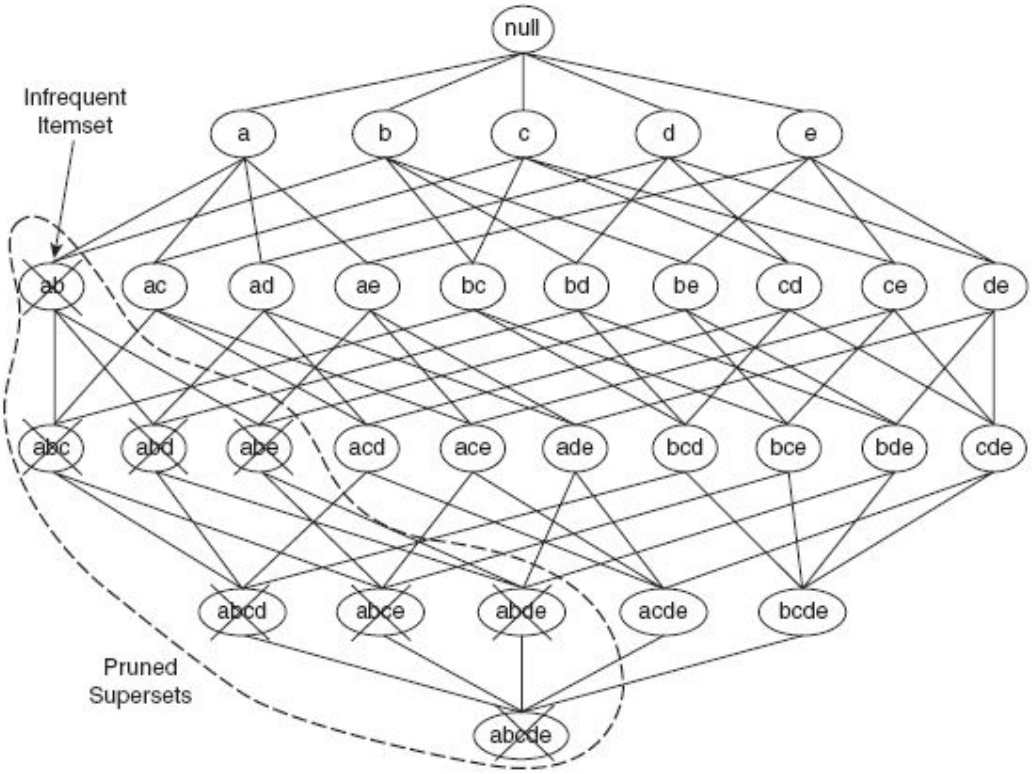


Figure 6.4. An illustration of support-based pruning.

If {a, b} is infrequent, then all supersets of {a, b} are infrequent, which means that if X is a subset of Y , then f(X) must not exceed f(Y). On the other hand, f is anti-monotone (or downward closed) if which means that if X is a subset of Y , then f(Y) must not exceed f(X).

$$\forall X, Y \in J : \ (X \subseteq Y) \longrightarrow f(Y) \leq f(X),$$

Any measure that possesses an anti-monotone property can be incorporated directly into the mining algorithm to effectively prune the exponential search space of candidate itemsets, as will be shown in the next section.

4.2.2 Frequent Itemset Generation in the Apriori Algorithm

Apriori is the first association rule mining algorithm that pioneered the use of support-based pruning to systematically control the exponential growth of candidate itemsets. Figure 4.5 provides a high-level illustration of the frequent itemset generation part of the Apriori algorithm for the transactions shown in

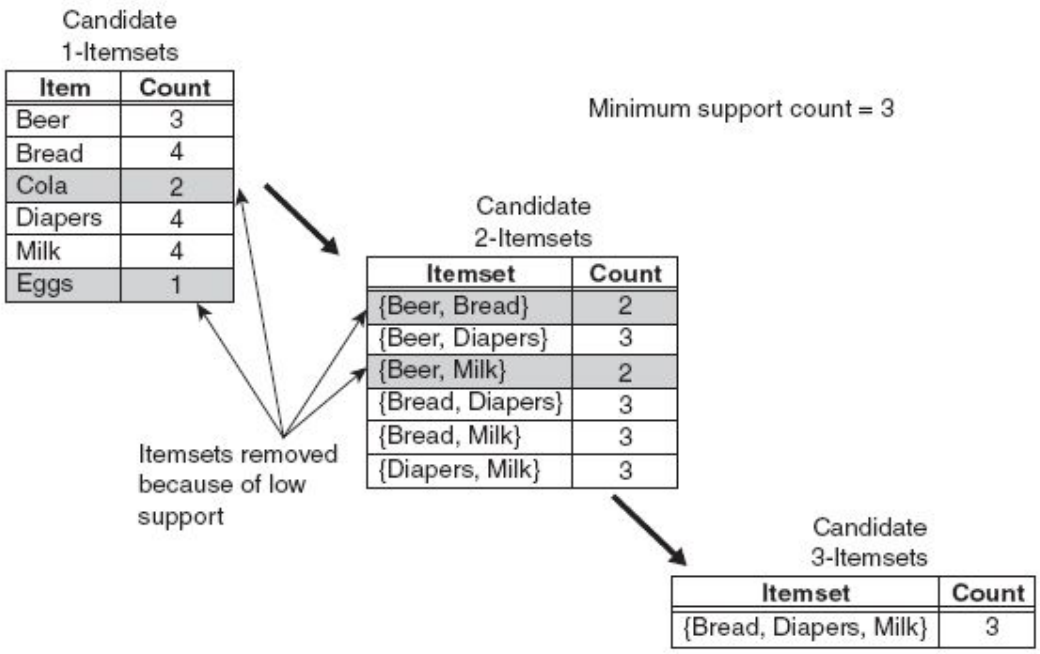


Figure 6.5. Illustration of frequent itemset generation using the Apriori algorithm.

4.3 Rule Generation

This section describes how to extract association rules efficiently from a given frequent itemset. Each frequent k-itemset, Y , can produce up to 2k-2 association rules, ignoring rules that have empty antecedents or consequents(0→Yor Y → 0). An association rule can be extracted by partitioning the itemset Y into two non-empty subsets, X and Y -X, such that X → Y - X satisfies the confidence threshold. Note that all such rules must have already met the support threshold because they are generated from a frequent itemset.

Example 4.2. Let X = {1, 2, 3} be a frequent itemset. There are six candidate association rules that can be generated from X: {1, 2} →{3}, {1, 3} →{2}, {2, 3}→{1}, {1}→{2, 3}, {2}→{1, 3}, and {3}→{1, 2}. As each of their support is identical to the support for X, the rules must satisfy the support threshold.

Computing the confidence of an association rule does not require additional scans of the

transaction data set. Consider the rule {1, 2} →{3}, which is generated from the frequent itemset X = {1, 2, 3}. The confidence for this rule is σ({1, 2, 3})/σ({1, 2}). Because {1, 2, 3} is frequent, the anti-monotone property of support ensures that {1, 2} must be frequent, too. Since the support counts for both itemsets were already found during frequent itemset generation, there is no need to read the entire data set again.

4.3.1 Confidence-Based Pruning

Unlike the support measure, confidence does not have any monotone property. For example, the confidence for $X \longrightarrow Y$ can be larger, smaller, or equal to the confidence for another rule $\tilde{X} \longrightarrow \tilde{Y}$, where $\tilde{X} \subseteq X$ and $\tilde{Y} \subseteq Y$ (see Exercise 3 on page 405). Nevertheless, if we compare rules generated from the same frequent itemset Y, the following theorem holds for the confidence measure.

Theorem .2. *If a rule $X \longrightarrow Y - X$ does not satisfy the confidence threshold, then any rule $X' \longrightarrow Y - X'$, where X' is a subset of X, must not satisfy the confidence threshold as well.*

To prove this theorem, consider the following two rules: $X' \longrightarrow Y - X'$ and $X \longrightarrow Y - X$, where $X' \subset X$. The confidence of the rules are $\sigma(Y)/\sigma(X')$ and $\sigma(Y)/\sigma(X)$, respectively. Since X' is a subset of X, $\sigma(X') \geq \sigma(X)$. Therefore, the former rule cannot have a higher confidence than the latter rule.

4.3.2 Rule Generation in *Apriori* Algorithm

The Apriori algorithm uses a level-wise approach for generating association rules, where each level corresponds to the number of items that belong to the rule consequent. Initially, all the high-confidence rules that have only one item in the rule consequent are extracted. These rules are then used to generate new candidate rules. For example, if {acd}→{b} and {abd}→{c} are high-confidence rules, then the candidate rule {ad} →{bc} is generated by merging the consequents of both rules. Figure 4.15 shows a lattice structure for the association rules generated from the frequent itemset {a, b, c, d}.

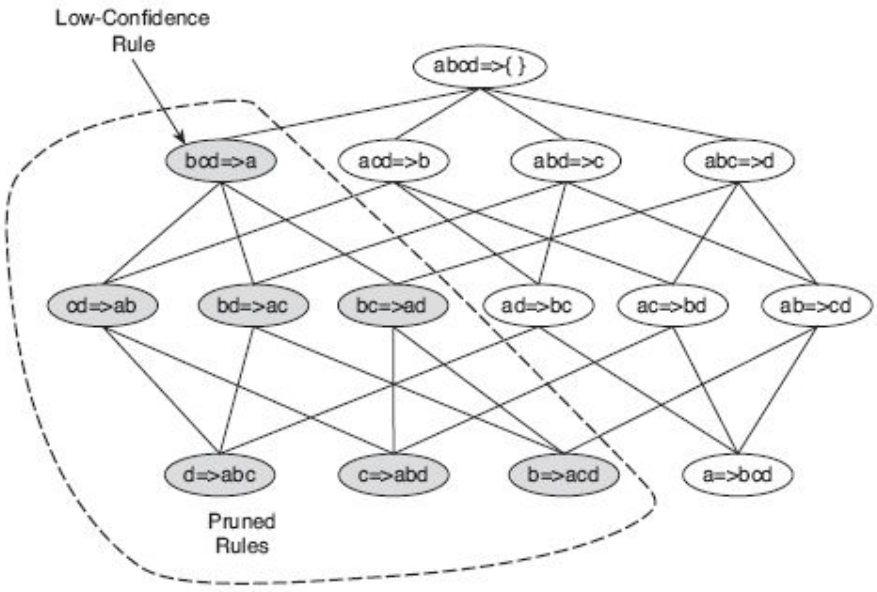


Figure 6.15. Pruning of association rules using the confidence measure.

Suppose the confidence for {bcd} →{a} is low. All the rules containing item a in its consequent, including {cd} →{ab}, {bd}→{ac}, {bc} →{ad}, and {d} →{abc} can be discarded.

The only difference is that, in rule generation, we do not have to make additional passes over the data set to compute the confidence of the candidate rules. Instead, we determine

the confidence of each rule by using the support counts computed during frequent itemset generation.

Algorithm 4.2 Rule generation of the *Apriori* algorithm.

```
1: for each frequent k-itemset f_k, k ≥ 2 do
2:    H_1 = {i | i ∈ f_k}        {1-item consequents of the rule.}
3:    call ap-genrules(f_k, H_1.)
4: end for
```

Algorithm 4.3 Procedure ap-genrules(f_k, H_m).

```
1: k = |f_k|    {size of frequent itemset.}
2: m = |H_m|    {size of rule consequent.}
3: if k > m + 1 then
4:    H_{m+1} = apriori-gen(H_m).
5:    for each h_{m+1} ∈ H_{m+1} do
6:       conf = σ(f_k)/σ(f_k - h_{m+1}).
7:       if conf ≥ minconf then
8:          output the rule (f_k - h_{m+1}) --> h_{m+1}.
9:       else
10:         delete h_{m+1} from H_{m+1}.
11:      end if
12:   end for
13:   call ap-genrules(f_k, H_{m+1}.)
14: end if
```

4.4 Compact representation of frequent item sets

In practice, the number of frequent itemsets produced from a transaction data set can be very large. It is useful to identify a small representative set of itemsets from which all other frequent itemsets can be derived. Two such representations are presented in this section in the form of maximal and closed frequent itemsets.

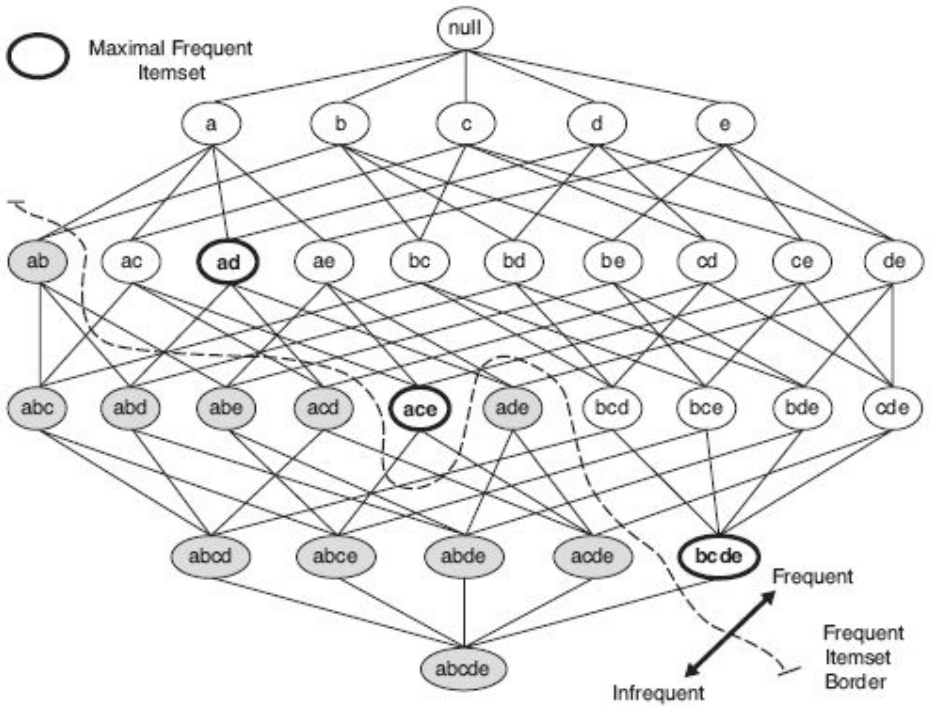


Figure 4.16. Maximal frequent itemset.

Definition 4.3 (Maximal Frequent Itemset). A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent.

To illustrate this concept, consider the itemset lattice shown in Figure 4.16. The itemsets in the lattice are divided into two groups: those that are frequent and those that are infrequent. A frequent itemset border, which is represented by a dashed line, is also illustrated in the diagram. Every itemset located above the border is frequent, while those located below the border (the shaded nodes) are infrequent. Among the itemsets residing near the border, {a, d}, {a, c, e}, and {b, c, d, e} are considered to be maximal frequent itemsets because their immediate supersets are infrequent. An itemset such as {a, d} is maximal frequent because all of its immediate supersets, {a, b, d}, {a, c, d}, and {a, d, e}, are infrequent. In contrast, {a, c} is non-maximal because one of its immediate supersets, {a, c, e}, is frequent.

5

For example, the frequent itemsets shown in Figure 4.16 can be divided into two groups:

• Frequent itemsets that begin with item a and that may contain items c, d, or e. This group includes itemsets such as {a}, {a, c}, {a, d}, {a, e}, and {a, c, e}.

• Frequent itemsets that begin with items b, c, d, or e. This group includes itemsets such as {b}, {b, c}, {c, d},{b, c, d, e}, etc.

4.4.2 Closed Frequent Itemsets

Closed itemsets provide a minimal representation of itemsets without losing their support information. A formal definition of a closed itemset is presented below.

Definition 4.4 (Closed Itemset). An itemset X is closed if none of its immediate supersets has exactly the same support count as X. Put another way, X is not closed if at least one of its immediate supersets has the same support count as X. Examples of closed itemsets are shown in

Figure 4.17 illustrate the support count of each itemset, we have associated each node (itemset) in the lattice with a list of its corresponding transaction IDs.

Definition 4.5 (Closed Frequent Itemset). An itemset is a closed frequent itemset if it is closed and its support is greater than or equal to minsup. Algorithms are available to explicitly extract closed frequent itemsets from a given data set. Interested readers may refer to the bibliographic notes at the end of this chapter for further discussions of these algorithms. We can use the closed frequent itemsets to determine the support counts for the non-closed Representation of Frequent Itemsets

Algorithm 4.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets
2 Let $k_{\max}$ denote the maximum size of closed frequent itemsets
3: $F_{k_{\max}} = \{f \mid f \in C, |f| = k_{\max}\}$ {Find all frequent itemsets of size $k_{\max}$.}
4: **for** $k = k_{\max} - 1$ downto 1 **do**
5: **for** each $f \in F_k$ **do** {Find all frequent itemsets of size k.}
6: $F_k = \{f \mid f \in F_{k+1}, |f| = k\}$
7: **if** $f \notin C$ **then**
8: $f.support = \max\{f'.support \mid f' \in F_{k+1}, f \subset f'\}$
end if

10: **end for**
11: **end for**

The algorithm proceeds in a specific-to-general fashion, i.e., from the largest to the smallest frequent itemsets. This is because, in order to find the support for a non-closed frequent itemset, the support for all of its supersets must be known.

TID	a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3	b_4	b_5	c_1	c_2	c_3	c_4	c_5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

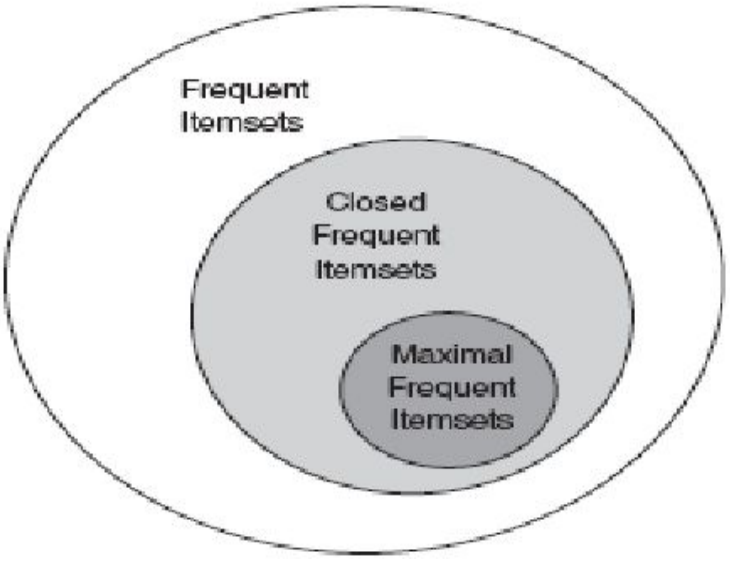


Figure 4.17 relationship among frequent, maximum frequent, and closed frequent itemset.

Closed frequent itemsets are useful for removing some of the redundant association

rules. An association rule X → Y is redundant if there exists another rule X → Y , where X is a subset of X and Y is a subset of Y , such that the support and confidence for both rules are identical. In the example shown in Figure 4.17, {b} is not a closed frequent itemset while {b, c} is closed.

The association rule {b} →{d, e} is therefore redundant because it has the same support and confidence as {b, c} →{d, e}. Such redundant rules are not generated if closed frequent itemsets are used for rule generation.

4.6 Alternative methods for generating frequent item sets.

Apriori is one of the earliest algorithms to have successfully addressed the combinatorial explosion of frequent itemset generation. It achieves this by applying the Apriori principle to prune the exponential search space. Despite its significant performance improvement, the algorithm still incurs considerable I/O overhead since it requires making several passes over the transaction data set.

• General-to-Specific versus Specific-to-General: The Apriori algorithm uses a general-to-specific search strategy, where pairs of frequent (k-1)-itemsets are merged to obtain candidate k-itemsets. This general-to-specific search strategy is effective, provided the maximum length of a frequent itemset is not too long. The configuration of frequent itemsets that works best with this strategy is shown in Figure 4.19(a), where the darker nodes represent infrequent itemsets. Alternatively, a specific-to-general search strategy looks for more specific frequent itemsets first, before finding the more general frequent itemsets. This strategy is use-ful to discover maximal frequent itemsets in dense transactions, where the frequent itemset border is located near the bottom of the lattice, as shown in Figure 4.19(b). The Apriori principle can be applied to prune all subsets of maximal frequent itemsets. Specifically, if a candidate k-itemset is maximal frequent, we do not have to examine any of its subsets of size k - 1. However, if the candidate k-itemset is infrequent, we need to check all of its k - 1 subsets in the next iteration. Another approach is to combine both general-to-specific and specific-to-general search strategies. This bidirectional

approach requires more space to

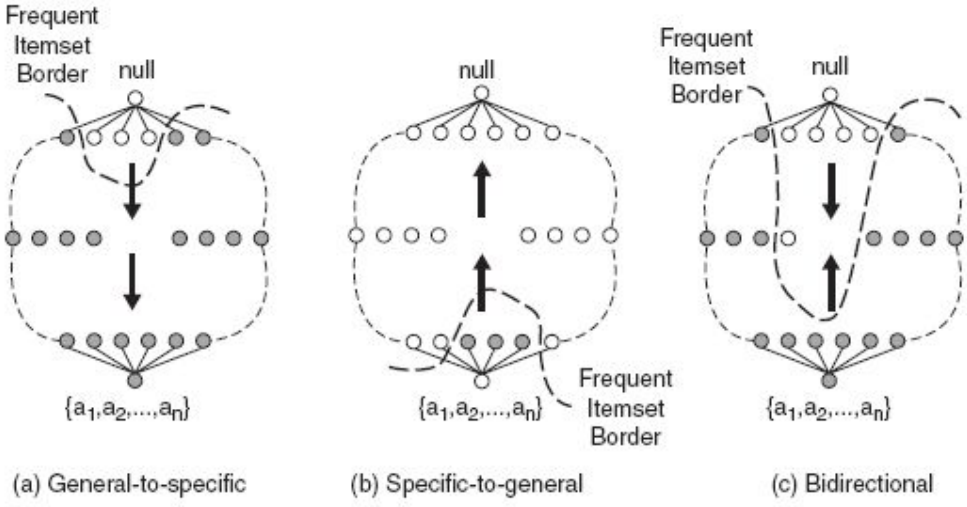


Figure 6.19. General-to-specific, specific-to-general, and bidirectional search.

• **Equivalence Classes:** Another way to envision the traversal is to first partition the lattice into disjoint groups of nodes (or equivalence classes). A frequent itemset generation algorithm searches for frequent itemsets within a particular equivalence class first before moving to another equivalence class. As an example, the level-wise strategy used in the Apriori algorithm can be considered to be partitioning the lattice on the basis of itemset sizes;

• **Breadth-First versus Depth-First:** The Apriori algorithm traverses the lattice in a breadth-first manner, as shown in Figure 6.21(a). It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated.

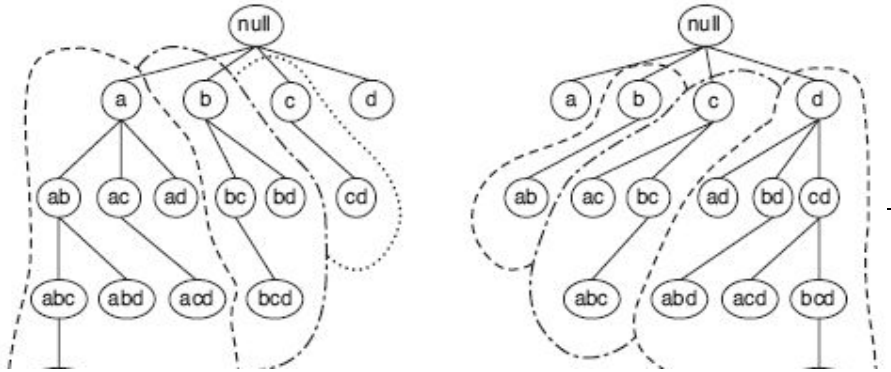
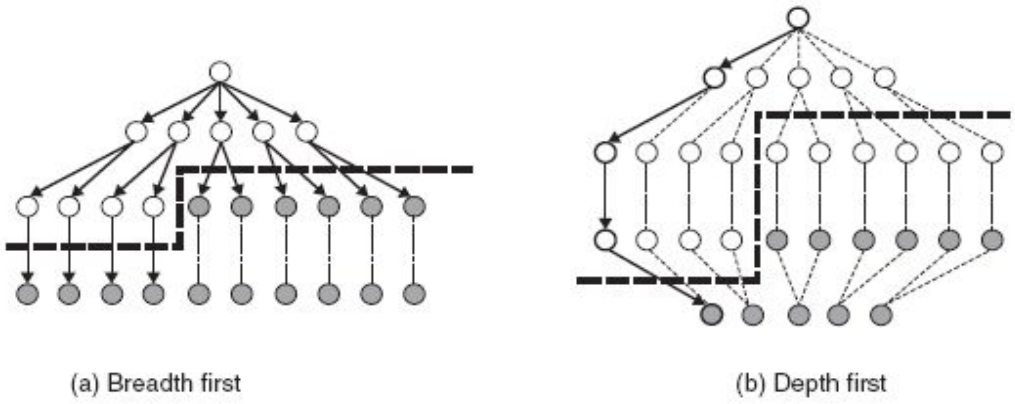


Figure 6.20. Equivalence classes based on prefix and suffix labels of item sets



Representation of Transaction Data Set There are many ways to represent a transaction data set. The choice of representation can affect the I/O costs incurred when computing the support of candidate itemsets. Figure 6.23 shows two different ways of representing market basket transactions. The representation on the left is called a **horizontal** data layout, which is adopted by many association rule mining algorithms, including Apriori. Another possibility is to store the list of transaction identifiers (TID-list) associated with each item. Such a representation is known as the **vertical** data layout. The support for each candidate itemset is obtained by intersecting the TID-lists of its subset items. The length of the TID-lists shrinks as we progress to larger sized itemsets.

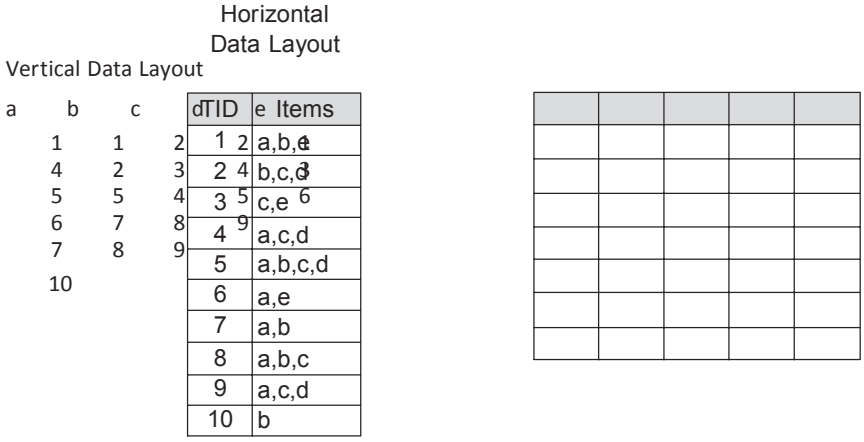


Figure 6.23. Horizontal and vertical data format.

However, one problem with this approach is that the initial set of TID-lists may be too large to fit into main memory, thus requiring more sophisticated techniques to compress the TID-lists. We describe another effective approach to represent the data in the next section.

Recommended Questions:

1. Explain Market Basket Analysis.
2. What is association analysis? Give an example of an association rule.
3. Explain the concept of Support and Confidence.
4. Explain ***Apriori*** Principle and the algorithm.
5. What is Brute-Force Method?
6. Explain Confidence-based Pruning.
7. Differentiate between Maximal Frequent Itemsets and Closed Frequent Itemsets.
8. Explain FP-Growth Algorithm in detail.