

UNIT-2

Introduction to Data Mining & Data Pre-processing

Syllabus: UNIT –I:

Introduction to Data Mining: Introduction, Definition, KDD, Challenges, Data Mining Functionalities. Data Objects and Attribute Types, Measuring Data Similarity and Dissimilarity, **Data Pre-processing:** Introduction, Data Pre-processing Overview, Data Cleaning, Data Integration and Transformation, Data Reduction, Discretization and Concept Hierarchy Generation.

Introduction

(Why Data Mining or Motivation for DM)

- We have in data rich but information poor situation. The fast-growing, tremendous amount of data, collected and stored in large and numerous data repositories, has far exceeded our human ability for comprehension without powerful tools. The widening gap between data and information calls for the systematic development of *data mining tools* that can turn data into knowledge.

Definition of Data Mining

(What is Data Mining?)

- Data mining is an iterative process of automatically discovering /extracting /uncovering interesting patterns (novel, non-trivial, implicit, previously unknown and potentially useful) and knowledge from *large* amounts of data via intelligent algorithms in order to support decisions/predictions/strategies.
- **Data mining**, also popularly referred to as *knowledge discovery from data (KDD)*, is the automated or convenient extraction of patterns representing knowledge implicitly stored or captured in large databases, data warehouses, the Web, other massive information repositories, or data streams.
- Data mining is an iterative process of extracting predictive and descriptive models by uncovering previously unknown trends and patterns in vast amounts of data in order to support decision making
- Technically, data mining is the computational process of analyzing data from different perspective, dimensions, angles and categorizing/summarizing it into meaningful information.
- The information gathered from Data Mining helps to predict hidden patterns, future trends and behaviours and allowing businesses to take decisions.
- As a multi disciplinary field, data mining draws on work including statistics, machine learning, pattern recognition, database technology, information retrieval, network science, knowledge-based systems,

artificial intelligence, high-performance computing, and data visualization.

- Many other terms have a similar meaning to data mining—for example, knowledge mining from data, knowledge extraction, data/pattern analysis, data archaeology, and data dredging.
- Data mining can be applied to any kind of data as long as the data are meaningful for a target application.
- The most basic forms of data for mining applications are: Database Data, Data warehouse Data, Transactional Data
- Data mining can also be applied to other forms of data: Data Streams, Time-related/Temporal/Ordered/Sequence Data, Graph or Networked Data, Spatial Data, Text Data, Multimedia Data and The WWW or Web Data

Steps in KDD (Knowledge Discovery from Data)

(Steps involved in DM when viewed as a process of knowledge discovery)

- We can view data mining as merely an essential step in the process of KDD.
- The term data mining often used to refer to the entire KDD process
- A KDD process typically involves data cleaning, data integration, data selection, data transformation, pattern discovery, pattern evaluation, and knowledge presentation.

1. Data cleaning

The process of removing noisy and inconsistent data and filling missing values is known as data cleaning

2. Data integration

The process of combining data from multiple heterogeneous data sources is known as data integration.

3. Data selection

In this step data relevant to the analysis task are retrieved from the database

4. Data transformation

In this step data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations

5. Data mining

It is essential process where intelligent methods are applied to extract data patterns

6. Pattern evaluation

In this step identify the truly interesting pattern representing knowledge based on interestingness measures.

7. Knowledge presentation

In this step visualization and knowledge representation techniques are used to present mined knowledge to users

→ The following diagram shows the process of KDD :

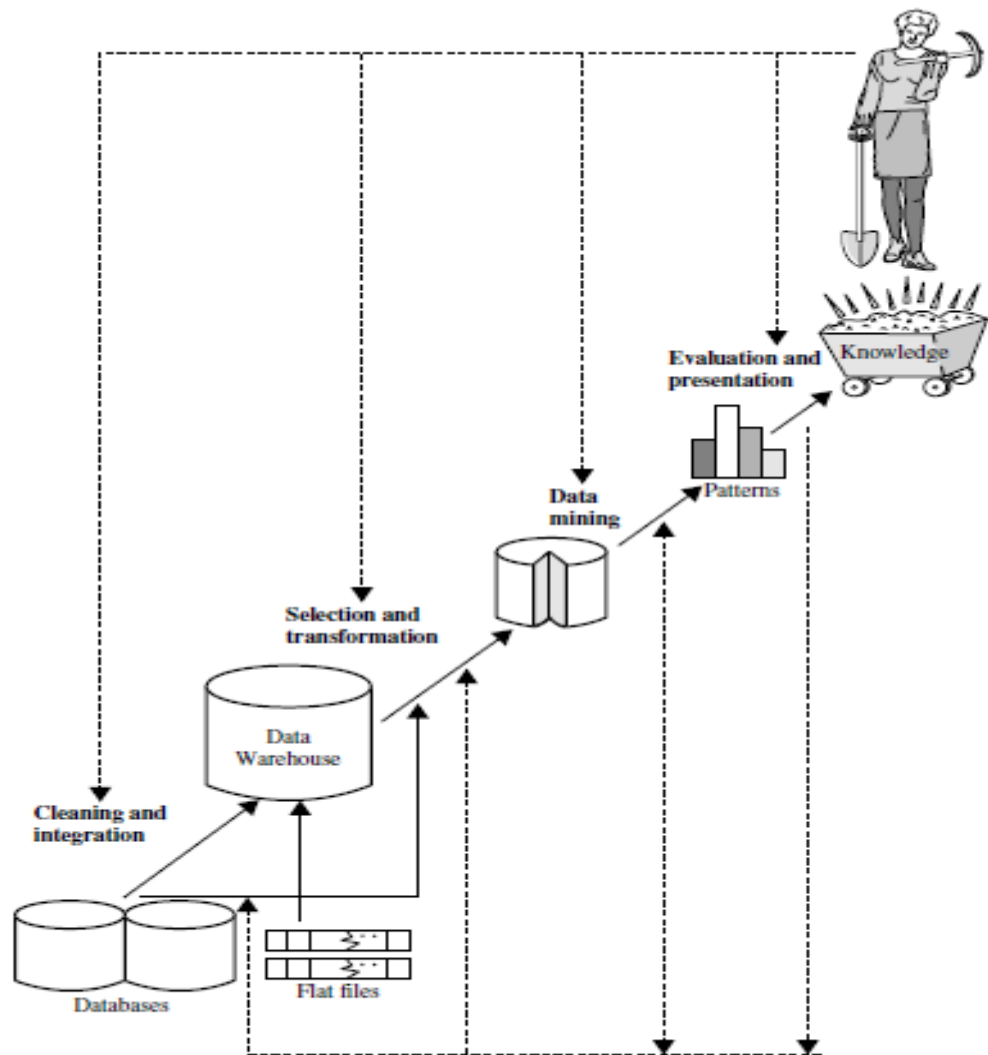


Figure 1.4 Data mining as a step in the process of knowledge discovery.

Challenges in Data Mining

(Major Issues in Data Mining)

- Data mining is a dynamic and fast-expanding field with great strengths.
- The major challenges in data mining are partitioned into five groups: mining methodology, user interaction, efficiency and scalability, diversity of data types, and data mining and society

Mining Methodology

- Various aspects of mining methodology are:

- **Mining various and new kinds of knowledge:** Different users may be interested in different kinds of knowledge. Therefore it is necessary for data mining to cover a broad range of knowledge discovery task.
- **Mining knowledge in multidimensional space:** When searching for knowledge in large data sets, we can explore the data in multidimensional space. That is, we can search for interesting patterns among combinations of dimensions (attributes) at varying levels of abstraction. Such mining is known as (*exploratory*) *multidimensional data mining*. In many cases, data can be aggregated or viewed as a multidimensional data cube. Mining knowledge in cube space can substantially enhance the power and flexibility of data mining.
- **Data mining—an interdisciplinary effort:** The power of data mining can be substantially enhanced by integrating new methods from multiple disciplines. For example to mine data with natural language text, it makes sense to fuse data mining methods with methods of information retrieval and natural language processing. As another example, consider the mining of software bugs in large programs. This form of mining, known as *bug mining*, benefits from the incorporation of software engineering knowledge into the data mining process.
- **Boosting the power of discovery in a networked environment:** Most data objects reside in a linked or interconnected environment, whether it be the Web, database relations, files, or documents. Semantic links across multiple data objects can be used to advantage in data mining. Knowledge derived in one set of objects can be used to boost the discovery of knowledge in a “related” or semantically linked set of objects.
- **Handling uncertainty, noise, or incompleteness of data:** Data often contain noise, errors, exceptions, or uncertainty, or are incomplete. Errors and noise may confuse the data mining process, leading to the derivation of erroneous patterns. Data cleaning, data preprocessing, outlier detection and removal, and uncertainty reasoning are examples of techniques that need to be integrated with the data mining process.
- **Pattern evaluation and pattern or constraint-guided mining:** Not all the patterns generated by data mining processes are interesting. What makes a pattern interesting may vary from user to user. Therefore, techniques are needed to assess the interestingness of discovered patterns based on subjective measures. These estimate the value of patterns with respect to a given user class, based on user beliefs or expectations. Moreover, by using interestingness measures or user-specified constraints to *guide* the discovery process, we may generate more interesting patterns and reduce the search space.

User Interaction

- The user plays an important role in the data mining process.
- Interesting areas of research include how to interact with a data mining system, how to incorporate a user's background knowledge in mining, and how to visualize and comprehend data mining results.
- **Interactive mining:** The data mining process should be highly *interactive*. Thus, it is important to build flexible user interfaces and an exploratory mining environment, facilitating the user's interaction with the system.
- **Incorporation of background knowledge:** Background knowledge, constraints, rules, and other information regarding the domain under study should be incorporated into the knowledge discovery process. Such knowledge can be used for pattern evaluation as well as to guide the search toward interesting patterns.
- **Ad hoc data mining and data mining query languages:** Data Mining Query language that allows the user to describe ad hoc mining tasks, should be integrated with a data warehouse query language and optimized for efficient and flexible data mining.
- **Presentation and visualization of data mining results:** – Once the patterns are discovered it needs to be expressed in high level languages, and visual representations. These representations should be easily understandable.

Efficiency and Scalability

- Efficiency and scalability are always considered when comparing data mining algorithms.
- As data amounts continue to multiply, these two factors are especially critical.
- **Efficiency and scalability of data mining algorithms:** Data mining algorithms must be efficient and scalable in order to effectively extract information from huge amounts of data in many data repositories or in dynamic data streams. In other words, the running time of a data mining algorithm must be predictable, short, and acceptable by applications. *Efficiency, scalability, performance, optimization*, and the ability to execute in *real time* are key criteria that drive the development of many new data mining algorithms.
- **Parallel, distributed, and incremental mining algorithms:** The factors such as huge size of databases, wide distribution of data, and complexity of data mining methods motivate the development of parallel and distributed data mining algorithms. These algorithms divide the data into partitions which is further processed in a parallel fashion. Then the results

from the partitions are merged. The **incremental algorithms**, update databases without mining the data again from scratch.

Diversity of Database Types

- The wide diversity of database types brings about challenges to data mining. These include:
- **Handling complex types of data:** Diverse applications generate a wide spectrum of new data types, from structured data such as relational and data warehouse data to semi-structured and unstructured data; from stable data repositories to dynamic data streams; from simple data objects to temporal data, biological sequences, sensor data, spatial data, hypertext data, multimedia data, software program code, Web data, and social network data. It is unrealistic to expect one data mining system to mine all kinds of data, given the diversity of data types and the different goals of data mining. Domain- or application-dedicated data mining systems are being constructed for in depth mining of specific kinds of data. The construction of effective and efficient data mining tools for diverse applications remains a challenging and active area of research.
- **Mining dynamic, networked, and global data repositories:** Multiple sources of data are connected by the Internet and various kinds of networks, forming gigantic, distributed, and heterogeneous global information systems and networks. The discovery of knowledge from different sources of structured, semi-structured, or unstructured yet interconnected data with diverse data semantics poses great challenges to data mining. Mining such gigantic, interconnected information networks may help disclose many more patterns and knowledge in heterogeneous data sets than can be discovered from a small set of isolated data repositories. Web mining, multisource data mining and information network mining have become challenging and fast-evolving data mining fields.

Data Mining and Society

- **Social impacts of data mining:** With data mining penetrating our everyday lives, it is important to study the impact of data mining on society. How can we use data mining technology to benefit society? How can we guard against its misuse? The improper disclosure or use of data and the potential violation of individual privacy and data protection rights are areas of concern that need to be addressed.
- **Privacy-preserving data mining:** Data mining will help scientific discovery, business management, economy recovery, and security protection (e.g., the real-time discovery of intruders and cyber attacks). However, it poses the risk of disclosing an individual's personal information. Studies on privacy-preserving data publishing and data

mining are ongoing. The philosophy is to observe data sensitivity and preserve people's privacy while performing successful data mining.

- **Invisible data mining:** We cannot expect everyone in society to learn and master data mining techniques. More and more systems should have data mining functions built within so that people can perform data mining or use data mining results simply by mouse clicking, without any knowledge of data mining algorithms.

Data Mining Functionalities

- Data mining functionalities are used to specify the kinds of patterns to be found in data mining tasks. In general, such tasks can be classified into two categories: **descriptive** and **predictive**.
- Descriptive data mining tasks describes data in a concise and summarative manner and presents interesting general properties of the data in a target data set. Predictive mining tasks analyze data in order to construct one or a set of models on the current data, and attempts to predict the behavior of new data sets.
- There are a number of *data mining functionalities*. These include:
 - Characterization and Discrimination
 - Association Analysis
 - Classification and Regression
 - Clustering Analysis
 - Outlier Analysis or Anomaly Mining or Deviation Detection
 - Evolution Analysis or Trend Analysis

Characterization and Discrimination

- Class/Concept description is the most basic form of descriptive data mining. It describes a given set of task-relevant data in a concise and summarative manner, presenting interesting general properties of the data. Concept (or class) description consists of characterization and comparison (or discrimination).
- **Data characterization** is a summarization of the general characteristics or features of a target class of data.
- **Eaxample1:** For example to summarize the characteristics of customers who spend more than \$5000 a year at company. The result is a general profile of these customers, such as that they are 40 to 50 years old, employed, and have excellent credit ratings.
- **Discrimination** is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes.
- **Example:** For example, to compare two groups of customers in a company—those who shop for computer products regularly (e.g., more

than twice a month) and those who rarely shop for such products (e.g., less than three times a year). The resulting description provides a general comparative profile of these customers, such as that 80% of the customers who frequently purchase computer products are between 20 and 40 years old and have a university education, whereas 60% of the customers who infrequently buy such products are either seniors or youths, and have no university degree.

- The output of data characterization can be presented in various forms. Examples include **pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables**, including crosstabs. The resulting descriptions can also be presented as **generalized relations** or in rule form (called **characteristic rules**).
- The forms of output presentation of discrimination are similar to those for characteristic descriptions, although discrimination descriptions should include comparative measures that help to distinguish between the target and contrasting classes. Discrimination descriptions expressed in the form of rules are referred to as **discriminant rules**.

Mining Frequent Patterns, Associations, and Correlations

- **Frequent patterns** are patterns that occur frequently in data. There are many kinds of frequent patterns, including frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures.
- A *frequent itemset* typically refers to a set of items that often appear together in a transactional data set—for example, milk and bread, which are frequently bought together in grocery stores by many customers.
- A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a digital camera, and then a memory card, is a (*frequent*) *sequential pattern*.
- A substructure can refer to different structural forms (e.g., graphs, trees, or lattices) that may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (*frequent*) *structured pattern*.
- Mining frequent patterns leads to the discovery of interesting associations and correlations within data.
- **Example:** Suppose that, as a marketing manager of company want to know which items are frequently purchased together (i.e., within the same transaction). An example of such a rule, mined from the company transactional database, is

buys(X, "computer") ⇒ buys(X, "software") [support = 1%, confidence = 50%],

where X is a variable representing a customer.

A **confidence**, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she will buy software as well.

A 1% **support** means that 1% of all the transactions under analysis show that computer and software are purchased together.

- Association rules that contain a single predicate are referred to as **single-dimensional association rules**. Dropping the predicate notation, the rule can be written simply as “*computer* \Rightarrow *software* [1%, 50%].”
- An association involving more than one attribute or predicate (i.e., *age*, *income*, and *buys*) are referred to as a **multidimensional association rule**. Example of a such rule is:

$$age(X, "20..29") \wedge income(X, "40K..49K") \Rightarrow buys(X, "laptop")$$

$$[support = 2\%, confidence = 60\%].$$

- Frequent itemset mining is a fundamental form of frequent pattern mining.

Classification and Regression

- **Classification** is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts.
- The model is derived based on the analysis of a set of **training data** (i.e., data objects for which the class labels are known).
- The model is used to predict the class label of objects for which the class label is unknown.
- The derived model may be represented in various forms, such as classification rules (i.e., IF-THEN rules), decision trees, mathematical formulae, or neural networks.
- There are many other methods for constructing classification models, such as **naïve** Bayesian classification, support vector machines, and *k*-nearest-neighbor classification.
- Classification predicts categorical (discrete, unordered) labels, where as **regression** models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction and class label prediction.
- **Regression analysis** is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Regression also encompasses the identification of distribution *trends* based on the available data.
- **Example1:** Suppose as a sales manager of *store* want to classify a large set of items in the store, based on three kinds of responses to a sales campaign: *good response*, *mild response* and *no response*. We want to derive a model for each of these three classes based on the descriptive

features of the items, such as *price*, *brand*, *place made*, *type*, and *category*.

- **Example2:** A common example of classification comes with detecting spam emails. To write a program to filter out spam emails, a computer programmer can train a machine learning algorithm with a set of spam-like emails labelled as spam and regular emails labelled as not-spam. The idea is to make an algorithm that can learn characteristics of spam emails from this training set so that it can filter out spam emails when it encounters new emails.

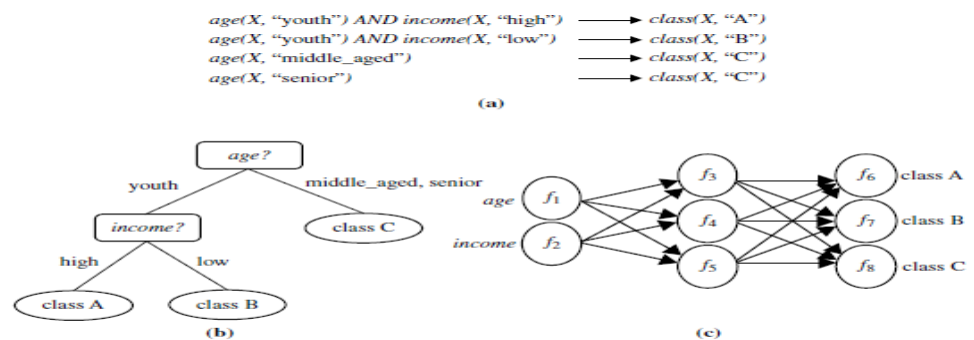


Figure 1.9 A classification model can be represented in various forms: (a) IF-THEN rules, (b) a decision tree, or (c) a neural network.

Cluster Analysis

- **Clustering** analyzes data objects without consulting class labels.
- Clustering can be used to generate class labels for a group of data.
- The objects are clustered or grouped based on the principle of *maximizing the intra class similarity and minimizing the interclass similarity*. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are rather dissimilar to objects in other clusters.
- Each cluster so formed can be viewed as a class of objects, from which rules can be derived.
- Clustering can also facilitate **taxonomy formation**, that is, the organization of observations into a hierarchy of classes that group similar events together.
- **Example:** Cluster analysis can be performed on *store* customer data to identify homogeneous subpopulations of customers. These clusters may represent individual target groups for marketing. Following figure shows a 2-D plot of customers with respect to customer locations in a city. Three clusters of data points are evident.

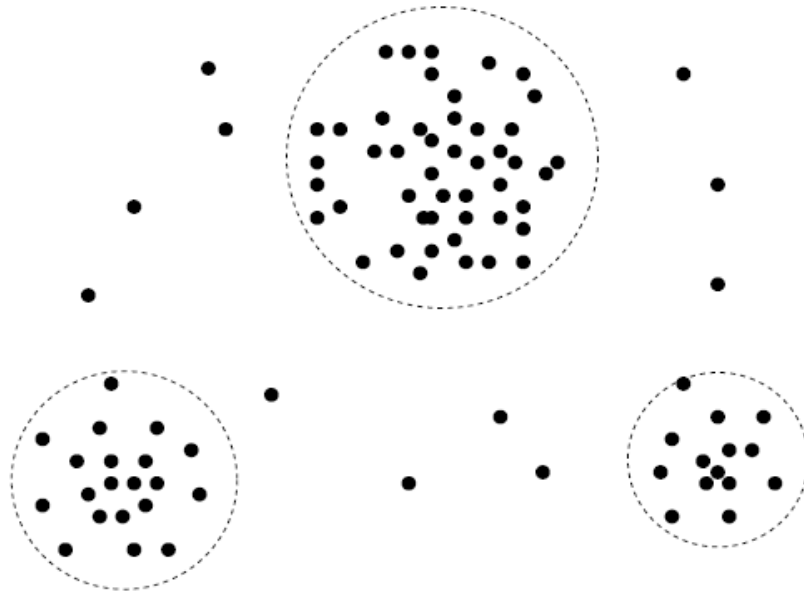


Figure 1.10 A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters.

Outlier Analysis

- Data objects in data set that do not comply with the general behaviour or model of the data are called **outliers or anomalies** or deviations
- Many data mining methods discard outliers as noise or exceptions. However, in some applications (e.g., fraud detection) the rare events can be more interesting than the more regularly occurring ones.
- The analysis of outlier data is referred to as **outlier analysis** or **anomaly mining** or **deviation detection**
- **Example:** Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of unusually large amounts for a given account number in comparison to regular charges incurred by the same account. Outlier values may also be detected with respect to the locations and types of purchase, or the purchase frequency.

Data Evolution Analysis

- Data evolution analysis describes and models regularities or trends for objects whose behaviour changes over time.
- Although this may include characterization, discrimination, association, classification, or clustering of time-related data, distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis.

Are All Patterns Interesting?

- A data mining system has the potential to generate thousands or even millions of patterns, or rules.

- “Are all of the patterns interesting?” Typically, the answer is no—only a small fraction of the patterns potentially generated would actually be of interest to a given user.
- “What makes a pattern interesting? A pattern is **interesting** if it is
 - (1) Easily understood by humans
 - (2) Valid on new or test data with some degree of certainty
 - (3) Potentially useful, and
 - (4) Novel.
- A pattern is also interesting if it validates a hypothesis that the user sought to confirm.
- An interesting pattern represents **knowledge**.
- Several **objective measures of pattern interestingness** exist. These are based on the structure of discovered patterns and the statistics underlying them. Ex: *support* and *confidence* for association rules
- **Subjective interestingness measures** are based on user beliefs in the data. These measures find patterns interesting if the patterns are **unexpected** (contradicting a user’s belief) or offer strategic information on which the user can act.
- “Can a data mining system generate all of the interesting patterns? Refers to the **completeness** of a data mining algorithm. It is often unrealistic and inefficient for data mining systems to generate all possible patterns
- “Can a data mining system generate only interesting patterns?” is an optimization problem in data mining. It is highly desirable for data mining systems to generate only interesting patterns. This would be efficient for users and data mining systems because neither would have to search through the patterns generated to identify the truly interesting ones.

Data Objects and Attribute Types

- Data sets are made up of data objects.
- A **data object** represents an entity.
- Examples:
 - in a sales database, the objects may be customers, store items, and sales
 - in a medical database, the objects may be patients
 - in a university database, the objects may be students, professors, and courses.
- Data objects are described by attributes.
- Data objects can also be referred to as *samples*, *examples*, *instances*, *data points*, or *objects*.

- If the data objects are stored in a database, they are *data tuples*. That is, the rows of a database correspond to the data objects, and the columns correspond to the attributes.

Patient_ID	Age	Height	Weight	Gender
1569	30	1.76m	70 kg	male
2594	24	1.65m	58kg	female

Data Object

Attributes

What Is an Attribute?

- An **attribute** is a data field, representing a characteristic or feature of a data object.
- The nouns *attribute*, *dimension*, *feature*, and *variable* are often used interchangeably in the literature.
- The term *dimension* is commonly used in data warehousing. Machine learning literature tends to use the term *feature*, while statisticians prefer the term *variable*. Data mining and database professionals commonly use the term *attribute*.
- Attributes describing a customer object can include, for example, *customer ID*, *name*, and *address*.
- Observed values for a given attribute are known as *observations*.
- A set of attributes used to describe a given object is called an *attribute vector* (or *feature vector*). The distribution of data involving one attribute (or variable) is called *univariate*. A *bivariate* distribution involves two attributes, and so on.

Name	Alternative Names
Data set	Table , Matrix, example set, data frame
Rows	Data objects, samples, examples, instances, data points, entities, records, observations, transactions, tuples, vectors, patterns, events, cases, feature-vectors, objects.
Columns	attributes, dimensions, features, fields, characteristics, properties, variables

Different Attribute Types

- The **type** of an attribute is determined by the set of possible values—nominal, binary, ordinal, or numeric—the attribute can have.
- Attributes can be:
 - ❖ Nominal(Categorical)
 - ❖ Binary
 - ❖ Ordinal
 - ❖ Numeric

Nominal Attributes

- Nominal means “relating to names.”
- The values of a **nominal** (or **categorical**) **attribute** are symbols or names of things, where each value represents some kind of category, code, or state.
- The values do not have any meaningful order.
- In computer science, the values are also known as *enumerations*.
- **Examples:** Suppose that *hair color*, *marital status* and *occupation* are three attributes describing *person* objects.
 - ❖ *hair color* with values *black*, *brown*, *blond*, *red*, *auburn*, *gray*, and *white*.
 - ❖ *marital status* with values *single*, *married*, *divorced*, and *widowed*.
 - ❖ *occupation*, with the values *teacher*, *dentist*, *programmer*, *farmer*, and so on.

Binary Attributes

- A **binary attribute** is a nominal attribute with only two categories or states: 0 or 1, where 0 typically means that the attribute is absent and 1 means that it is present.
- Binary attributes are referred to as **Boolean** if the two states correspond to *true* and *false*.
- **Examples:**
 - Given the attribute *smoker* describing a *patient* object, 1 indicates that the patient smokes, while 0 indicates that the patient does not.
 - Similarly, suppose the patient undergoes a medical test that has two possible outcomes. The attribute *medical test* is binary, where a value of 1 means the result of the test for the patient is positive, while 0 means the result is negative.
- A binary attribute is **symmetric** if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute *gender* having the states *male* and *female*.
- A binary attribute is **asymmetric** if the outcomes of the states are not equally important, such as the *positive* and *negative* outcomes of a medical test for HIV. By convention, we code the most important outcome, which is usually the rarest one, by 1 (e.g., *HIV positive*) and the other by 0 (e.g., *HIV negative*).

Ordinal Attributes

- An **ordinal attribute** is an attribute with possible values that have a meaningful order or *ranking* among them, but the magnitude between successive values is not known.

→ **Examples:**

- Suppose that *drink size* corresponds to the size of drinks available at a fast-food restaurant. This nominal attribute has three possible values: *small*, *medium*, and *large*. The values have a meaningful sequence (which corresponds to increasing drink size); however, we cannot tell from the values *how much* bigger, say, a medium is than a large.
 - Other examples of ordinal attributes include *grade* (e.g., *A+*, *A*, *A-*, *B+*) and *professional rank*. Professional ranks can be enumerated in a sequential order: for example, *assistant*, *associate*, and *full* for professors, and *private*, *private first class*, *specialist*, *corporal*, and *sergeant* for army ranks.
- Ordinal attributes are useful for registering subjective assessments of qualities that cannot be measured objectively; thus ordinal attributes are often used in surveys for ratings. In one survey, participants were asked to rate how satisfied they were as customers.

Numeric Attributes

- A **numeric attribute** is *quantitative*; that is, it is a measurable quantity, represented in integer or real values.
- Numeric attributes can be *interval-scaled* or *ratio-scaled*.

Interval-Scaled Attributes

- **Interval-scaled attributes** are measured on a scale of equal-size units.
- The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition to providing a ranking of values, such attributes allow us to compare and quantify the *difference* between values.
- Example: A *temperature* attribute is interval-scaled. Suppose that we have the outdoor *temperature* value for a number of different days, where each day is an object. By ordering the values, we obtain a ranking of the objects with respect to *temperature*. In addition, we can quantify the difference between values.

Ratio-Scaled Attributes

- A **ratio-scaled attribute** is a numeric attribute with an inherent zero-point. That is, if a measurement is ratio-scaled, we can speak of a value as being a multiple (or ratio) of another value.
- In addition, the values are ordered, and we can also compute the difference between values, as well as the mean, median, and mode.
- **Example:** Attributes to measure weight, height, latitude and longitude coordinates (e.g., when clustering houses), and monetary quantities (e.g., you are 100 times richer with \$100 than with \$1).

Note:

- Nominal, binary, and ordinal attributes are qualitative but numeric attribute is quantitative

Discrete versus Continuous Attributes

- A **discrete attribute** has a finite or countably infinite set of values, which may or may not be represented as integers.
- Example: The attributes *hair color*, *smoker*, *medical test*, and *drink size* each have a finite number of values, and so are discrete.
- An attribute is *countably infinite* if the set of possible values is infinite but the values can be put in a one-to-one correspondence with natural numbers. For example, the attribute *customer ID* is countably infinite. The number of customers can grow to infinity, but in reality, the actual set of values is countable (where the values can be put in one-to-one correspondence with the set of integers). Zip codes are another example.
- If an attribute is not discrete, it is **continuous**. They have real numbers as attribute values, Ex: temperature, height, or weight. Continuous attributes are typically represented as floating-point variables.

Measuring Data Similarity and Dissimilarity

(Measures of *proximity*)

- Measures of object **similarity** and **dissimilarity** are used in data mining applications such as clustering, outlier analysis, and nearest-neighbor classification.
- A similarity measure for two objects, i and j , will typically return the value 0 if the objects are unlike. The higher the similarity value, the greater the similarity between objects. (Typically, a value of 1 indicates complete similarity, that is, the objects are identical.)
- A dissimilarity measure works the opposite way. It returns a value of 0 if the objects are the same (and therefore, far from being dissimilar). The higher the dissimilarity value, the more dissimilar the two objects are.
- Proximity refers to a similarity or dissimilarity

Data Matrix versus Dissimilarity Matrix

(Data structures used in Measures of proximity)

- Two data structures that are commonly used in the measures of proximity:
 - ❖ the *data matrix* (used to store the data objects) and
 - ❖ the *dissimilarity matrix* (used to store dissimilarity values for pairs of objects).

Data matrix (or *object-by-attribute structure*):

- This structure stores the n data objects in the form of a relational table, or n -by- p matrix (n objects \times p attributes)
- Each row corresponds to an object.

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

Dissimilarity matrix (or *object-by-object structure*):

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

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

where $d(i, j)$ is the measured **dissimilarity** or “difference” between objects i and j .

- In general, $d(i, j)$ is a non-negative number that is close to 0 when objects i and j are highly similar or “near” each other, and becomes larger the more they differ.
- Note that $d(i, i) = 0$; that is, the difference between an object and itself is 0. Furthermore, $d(i, j) = d(j, i)$.
- Measures of similarity can often be expressed as a function of measures of dissimilarity. For example, for nominal data, $sim(i, j) = 1 - d(i, j)$ where $sim(i, j)$ is the similarity between objects i and j .
- A data matrix is made up of two entities or “things,” namely rows (for objects) and columns (for attributes). Therefore, the data matrix is often called a **two-mode** matrix.
- The dissimilarity matrix contains one kind of entity (dissimilarities) and so is called a **one-mode** matrix.

Proximity Measures for Nominal Attributes

- A nominal attribute can take on two or more states for example; *map color* is a nominal attribute that may have, say, five states: *red*, *yellow*, *green*, *pink*, and *blue*.
- Let the number of states of a nominal attribute be M . The states can be denoted by letters, symbols, or a set of integers, such as 1, 2, ..., M .
- The dissimilarity between two objects i and j can be computed based on the ratio of mismatches:

$$d(i, j) = \frac{p - m}{p},$$

where m is the number of *matches* (i.e., the number of attributes for which i and j are in the same state), and p is the total number of attributes describing the objects.

Table 2.2 A Sample Data Table Containing Attributes of Mixed Type

Object Identifier	test-1 (nominal)	test-2 (ordinal)	test-3 (numeric)
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

Example 2.17 Dissimilarity between nominal attributes. Suppose that we have the sample data of Table 2.2, except that only the *object-identifier* and the attribute *test-1* are available, where *test-1* is nominal. (We will use *test-2* and *test-3* in later examples.) Let's compute the dissimilarity matrix (Eq. 2.9), that is,

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}.$$

Since here we have one nominal attribute, *test-1*, we set $p = 1$ in Eq. (2.11) so that $d(i, j)$ evaluates to 0 if objects i and j match, and 1 if the objects differ. Thus, we get

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

From this, we see that all objects are dissimilar except objects 1 and 4 (i.e., $d(4,1) = 0$).

Proximity Measures for Binary Attributes

(How can we compute the dissimilarity between two binary attributes?)

- A binary attribute has only one of two states: 0 and 1, where 0 means that the attribute is absent, and 1 means that it is present .
- Given the attribute *smoker* describing a patient, for instance, 1 indicates that the patient smokes, while 0 indicates that the patient does not.
- One approach involves computing a dissimilarity matrix from the given binary data. If all binary attributes are thought of as having the same weight, we have the 2×2 contingency table of Table 2.3, where q is the number of attributes that equal 1 for both objects i and j , r is the number of attributes that equal 1 for object i but equal 0 for object j , s is the number of attributes that equal 0 for object i but equal 1 for object j , and t

is the number of attributes that equal 0 for both objects i and j . The total number of attributes is p , where $p = q + r + s + t$.

Table 2.3 Contingency Table for Binary Attributes

		Object j		
		1	0	sum
Object i	1	q	r	$q + r$
	0	s	t	$s + t$
	sum	$q + s$	$r + t$	p

→ If objects i and j are described by symmetric binary attributes, then the dissimilarity between i and j is

$$d(i, j) = \frac{r + s}{q + r + s + t}. \quad (2.13)$$

For asymmetric binary attributes, the two states are not equally important, such as the *positive* (1) and *negative* (0) outcomes of a disease test. Given two asymmetric binary attributes, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match). Therefore, such binary attributes are often considered “monary” (having one state). The dissimilarity based on these attributes is called **asymmetric binary dissimilarity**, where the number of negative matches, t , is considered unimportant and is thus ignored in the following computation:

$$d(i, j) = \frac{r + s}{q + r + s}. \quad (2.14)$$

Complementarily, we can measure the difference between two binary attributes based on the notion of similarity instead of dissimilarity. For example, the **asymmetric binary similarity** between the objects i and j can be computed as

$$\text{sim}(i, j) = \frac{q}{q + r + s} = 1 - d(i, j). \quad (2.15)$$

The coefficient $\text{sim}(i, j)$ of Eq. (2.15) is called the **Jaccard coefficient** and is popularly referenced in the literature.

Example 2.18 Dissimilarity between binary attributes. Suppose that a patient record table (Table 2.4) contains the attributes *name*, *gender*, *fever*, *cough*, *test-1*, *test-2*, *test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects

Table 2.4 Relational Table Where Patients Are Described by Binary Attributes

<i>name</i>	<i>gender</i>	<i>fever</i>	<i>cough</i>	<i>test-1</i>	<i>test-2</i>	<i>test-3</i>	<i>test-4</i>
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

(patients) is computed based only on the asymmetric attributes. According to Eq. (2.14), the distance between each pair of the three patients—Jack, Mary, and Jim—is

$$d(\text{Jack}, \text{Jim}) = \frac{1+1}{1+1+1} = 0.67,$$

$$d(\text{Jack}, \text{Mary}) = \frac{0+1}{2+0+1} = 0.33,$$

$$d(\text{Jim}, \text{Mary}) = \frac{1+2}{1+1+2} = 0.75.$$

These measurements suggest that Jim and Mary are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease. ■

Dissimilarity of Numeric Data

(Distance measures for numeric data)

→ Different types of distance measures for numeric data include:

- ❖ Euclidean Distance
- ❖ Manhattan Distance (also referred as city block distance)
- ❖ Minkowski Distance (also referred as L_p norm)
- ❖ Supremum Distance (also referred to as L_{max} , L_1 norm and as the Chebyshev distance)

Euclidean Distance

The most popular distance measure is **Euclidean distance** (i.e., straight line or “as the crow flies”). Let $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ be two objects described by p numeric attributes. The Euclidean distance between objects i and j is defined as

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ip} - x_{jp})^2}. \quad (2.16)$$

Manhattan Distance

Another well-known measure is the **Manhattan (or city block) distance**, named so because it is the distance in blocks between any two points in a city (such as 2 blocks down and 3 blocks over for a total of 5 blocks). It is defined as

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|. \quad (2.17)$$

Both the Euclidean and the Manhattan distance satisfy the following mathematical properties:

Non-negativity: $d(i, j) \geq 0$: Distance is a non-negative number.

Identity of indiscernibles: $d(i, i) = 0$: The distance of an object to itself is 0.

Symmetry: $d(i, j) = d(j, i)$: Distance is a symmetric function.

Triangle inequality: $d(i, j) \leq d(i, k) + d(k, j)$: Going directly from object i to object j in space is no more than making a detour over any other object k .

A measure that satisfies these conditions is known as **metric**. Please note that the non-negativity property is implied by the other three properties.

Minkowski Distance

Minkowski distance is a generalization of the Euclidean and Manhattan distances. It is defined as

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}, \quad (2.18)$$

where h is a real number such that $h \geq 1$. (Such a distance is also called L_p **norm** in some literature, where the symbol p refers to our notation of h . We have kept p as the number of attributes to be consistent with the rest of this chapter.) It represents the Manhattan distance when $h = 1$ (i.e., L_1 norm) and Euclidean distance when $h = 2$ (i.e., L_2 norm).

Supremum Distance

The **supremum distance** (also referred to as L_{\max} , L_{∞} **norm** and as the **Chebyshev distance**) is a generalization of the Minkowski distance for $h \rightarrow \infty$. To compute it, we find the attribute f that gives the maximum difference in values between the two objects. This difference is the supremum distance, defined more formally as:

$$d(i, j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f |x_{if} - x_{jf}|. \quad (2.19)$$

The L_{∞} norm is also known as the *uniform norm*.

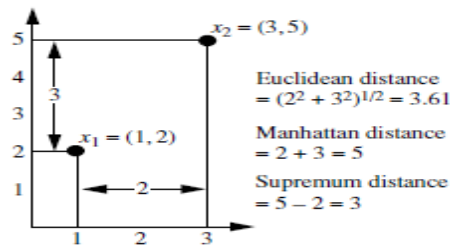


Figure 2.23 Euclidean, Manhattan, and supremum distances between two objects.

Euclidean distance and Manhattan distance. Let $x_1 = (1, 2)$ and $x_2 = (3, 5)$ represent two objects as shown in Figure 2.23. The Euclidean distance between the two is $\sqrt{2^2 + 3^2} = 3.61$. The Manhattan distance between the two is $2 + 3 = 5$. ■

Supremum distance. Let's use the same two objects, $x_1 = (1, 2)$ and $x_2 = (3, 5)$, as in Figure 2.23. The second attribute gives the greatest difference between values for the objects, which is $5 - 2 = 3$. This is the supremum distance between both objects. ■

Note:

The Euclidean distance of two n -dimensional vectors x and $y = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$

The Manhattan distance of two n -dimensional vectors x and $y = \sum_{i=1}^n |x_i - y_i|$

The Minkowski distance of two n -dimensional vectors x and $y = \sqrt[h]{\sum_{i=1}^n |x_i - y_i|^h}$

The Supremum distance of two n -dimensional vectors x and $y = \max_{i=1}^n |x_i - y_i|$

Proximity Measures for Ordinal Attributes

- The values of an ordinal attribute have a meaningful order or ranking about them, yet the magnitude between successive values is unknown
- An example includes the sequence *small, medium, large* for a *size* attribute.
- Ordinal attributes may also be obtained from the discretization of numeric attributes by splitting the value range into a finite number of categories. These categories are organized into ranks. That is, the range of a numeric attribute can be mapped to an ordinal attribute f having M_f states. For example, the range of the interval-scaled attribute *temperature* (in Celsius) can be organized into the following states: -30 to -10, -10 to 10, 10 to 30, representing the categories *cold temperature*, *moderate temperature*, and *warm temperature*, respectively. Let M represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking $1, \dots, M_f$.

→ “How are ordinal attributes handled?” The treatment of ordinal attributes is quite similar to that of numeric attributes when computing dissimilarity between objects.

→ Suppose that f is an attribute from a set of ordinal attributes describing n objects. The dissimilarity computation with respect to f involves the following steps:

1. The value of f for the i th object is x_{if} , and f has M_f ordered states, representing the ranking $1, \dots, M_f$. Replace each x_{if} by its corresponding rank, $r_{if} \in \{1, \dots, M_f\}$.
2. Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto $[0.0, 1.0]$ so that each attribute has equal weight. We perform such data normalization by replacing the rank r_{if} of the i th object in the f th attribute by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \quad (2.21)$$

3. Dissimilarity can then be computed using any of the distance measures described in Section 2.4.4 for numeric attributes, using z_{if} to represent the f value for the i th object.

Example 2.21 Dissimilarity between ordinal attributes. Suppose that we have the sample data shown earlier in Table 2.2, except that this time only the *object-identifier* and the continuous ordinal attribute, *test-2*, are available. There are three states for *test-2*: *fair*, *good*, and *excellent*, that is, $M_f = 3$. For step 1, if we replace each value for *test-2* by its rank, the four objects are assigned the ranks 3, 1, 2, and 3, respectively. Step 2 normalizes the ranking by mapping rank 1 to 0.0, rank 2 to 0.5, and rank 3 to 1.0. For step 3, we can use, say, the Euclidean distance (Eq. 2.16), which results in the following dissimilarity matrix:

$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}.$$

Therefore, objects 1 and 2 are the most dissimilar, as are objects 2 and 4 (i.e., $d(2, 1) = 1.0$ and $d(4, 2) = 1.0$). This makes intuitive sense since objects 1 and 4 are both *excellent*. Object 2 is *fair*, which is at the opposite end of the range of values for *test-2*. ■

Similarity values for ordinal attributes can be interpreted from dissimilarity as $\text{sim}(i, j) = 1 - d(i, j)$.

Dissimilarity for Attributes of Mixed Types

(How can we compute the dissimilarity between objects of mixed attribute types?)

→ One approach is to group each type of attribute together, performing separate data mining (e.g., clustering) analysis for each type. This is feasible if these analyses derive compatible results. However, in real applications, it is unlikely that a separate analysis per attribute type will generate compatible results.

- A more preferable approach is to process all attribute types together, performing a single analysis. One such technique combines the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0].
- Suppose that the data set contains p attributes of mixed type. The dissimilarity $d(i, j)$ between objects i and j is defined as

$$d(i, j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}},$$

where the indicator $\delta_{ij}^{(f)} = 0$ if either (1) x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute f for object i or object j), or (2) $x_{if} = x_{jf} = 0$ and attribute f is asymmetric binary; otherwise, $\delta_{ij}^{(f)} = 1$. The contribution of attribute f to the dissimilarity between i and j (i.e., $d_{ij}^{(f)}$) is computed dependent on its type:

- If f is numeric: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$, where h runs over all nonmissing objects for attribute f .
- If f is nominal or binary: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; otherwise, $d_{ij}^{(f)} = 1$.
- If f is ordinal: compute the ranks r_{if} and $z_{if} = \frac{r_{if} - 1}{M_f - 1}$, and treat z_{if} as numeric.

These steps are identical to what we have already seen for each of the individual attribute types. The only difference is for numeric attributes, where we normalize so that the values map to the interval [0.0, 1.0]. Thus, the dissimilarity between objects can be computed even when the attributes describing the objects are of different types.

Example 2.22 Dissimilarity between attributes of mixed type. Let's compute a dissimilarity matrix for the objects in Table 2.2. Now we will consider *all* of the attributes, which are of different types. In Examples 2.17 and 2.21, we worked out the dissimilarity matrices for each of the individual attributes. The procedures we followed for *test-1* (which is nominal) and *test-2* (which is ordinal) are the same as outlined earlier for processing attributes of mixed types. Therefore, we can use the dissimilarity matrices obtained for *test-1* and *test-2* later when we compute Eq. (2.22). First, however, we need to compute the dissimilarity matrix for the third attribute, *test-3* (which is numeric). That is, we must compute $d_{ij}^{(3)}$. Following the case for numeric attributes, we let $\max_h x_{3h} = 64$ and $\min_h x_{3h} = 22$. The difference between the two is used in Eq. (2.22) to normalize the values of the dissimilarity matrix. The resulting dissimilarity matrix for *test-3* is

$$\begin{bmatrix} 0 & & & \\ 0.55 & 0 & & \\ 0.45 & 1.00 & 0 & \\ 0.40 & 0.14 & 0.86 & 0 \end{bmatrix}.$$

We can now use the dissimilarity matrices for the three attributes in our computation of Eq. (2.22). The indicator $\delta_{ij}^{(f)} = 1$ for each of the three attributes, f . We get, for example, $d(3, 1) = \frac{1(1) + 1(0.50) + 1(0.45)}{3} = 0.65$. The resulting dissimilarity matrix obtained for the

data described by the three attributes of mixed types is:

$$\begin{bmatrix} 0 \\ 0.85 & 0 \\ 0.65 & 0.83 & 0 \\ 0.13 & 0.71 & 0.79 & 0 \end{bmatrix}.$$

From Table 2.2, we can intuitively guess that objects 1 and 4 are the most similar, based on their values for *test-1* and *test-2*. This is confirmed by the dissimilarity matrix, where $d(4, 1)$ is the lowest value for any pair of different objects. Similarly, the matrix indicates that objects 1 and 2 are the least similar. ■

Cosine Similarity

- A document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document. Thus, each document is an object represented by what is called a *term-frequency vector*.
- For example, in following table, we see that *Document1* contains five instances of the word *team*, while *hockey* occurs three times. The word *coach* is absent from the entire document, as indicated by a count value of 0.

Document Vector or Term-Frequency Vector

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
<i>Document1</i>	5	0	3	0	2	0	0	2	0	0
<i>Document2</i>	3	0	2	0	1	1	0	1	0	1
<i>Document3</i>	0	7	0	2	1	0	0	3	0	0
<i>Document4</i>	0	1	0	0	1	2	2	0	3	0

- **Cosine similarity** is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words.

Let x and y be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$\text{sim}(x, y) = \frac{x \cdot y}{\|x\| \|y\|}, \quad (2.23)$$

where $\|x\|$ is the Euclidean norm of vector $x = (x_1, x_2, \dots, x_p)$, defined as $\sqrt{x_1^2 + x_2^2 + \dots + x_p^2}$. Conceptually, it is the length of the vector. Similarly, $\|y\|$ is the Euclidean norm of vector y . The measure computes the cosine of the angle between vectors x and y . A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle and the greater the match between vectors. Note that because the cosine similarity measure does not obey all of the properties of Section 2.4.4 defining metric measures, it is referred to as a *nonmetric measure*.

Example 2.23 Cosine similarity between two term-frequency vectors. Suppose that x and y are the first two term-frequency vectors in Table 2.5. That is, $x = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$ and $y = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$. How similar are x and y ? Using Eq. (2.23) to compute the cosine similarity between the two vectors, we get:

$$\begin{aligned} x^t \cdot y &= 5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 1 + 0 \times 1 + 0 \times 0 + 2 \times 1 \\ &\quad + 0 \times 0 + 0 \times 1 = 25 \\ \|x\| &= \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48 \\ \|y\| &= \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2} = 4.12 \\ \text{sim}(x, y) &= 0.94 \end{aligned}$$

Therefore, if we were using the cosine similarity measure to compare these documents, they would be considered quite similar. ■

When attributes are binary-valued, the cosine similarity function can be interpreted in terms of shared features or attributes. Suppose an object x possesses the i th attribute if $x_i = 1$. Then $x^t \cdot y$ is the number of attributes possessed (i.e., shared) by both x and y , and $\|x\| \|y\|$ is the *geometric mean* of the number of attributes possessed by x and the number possessed by y . Thus, $\text{sim}(x, y)$ is a measure of relative possession of common attributes.

A simple variation of cosine similarity for the preceding scenario is

$$\text{sim}(x, y) = \frac{x \cdot y}{x \cdot x + y \cdot y - x \cdot y}, \quad (2.24)$$

which is the ratio of the number of attributes shared by x and y to the number of attributes possessed by x or y . This function, known as the **Tanimoto coefficient** or **Tanimoto distance**, is frequently used in information retrieval and biology taxonomy.

Pre-processing: An Overview

- Real-world data tend to be incorrect (inaccurate or noisy or dirty), incomplete (missing), and inconsistent due to their typically huge size and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results.
- Process of transforming the raw data into appropriate, useful and efficient format for subsequent analysis is called data preprocessing.
- Data Preprocessing prepares raw data for further processing.
- Data processing techniques substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

Why Preprocess the Data?

(Need of Data Pre-processing)

- Real world data tend to be inaccurate (incorrect or noisy or dirty), incomplete and inconsistent.
 - *incomplete* : lacking attribute values or certain attributes of interest, or containing only aggregate data
 - *inaccurate or noisy* :containing errors, or values that deviate from the expected
 - *inconsistent*: containing discrepancies in the department codes used to categorize items
- Data preprocessing techniques improve the quality, accuracy and efficiency of the subsequent mining.
- Data preprocessing is important step in the knowledge discovery process because quality decision must base on quality data.
- Detecting data anomalies, rectifying them early and reducing the data to be analyzed can lead to huge payoffs for decision making.

Major Tasks in Data Pre-processing

(Data Pre-processing Techniques (or) Forms of data Pre-processing)

- The major steps involved in data preprocessing are:
 - Data Cleaning
 - Data Integration
 - Data Reduction
 - Data Transformation.
- *Data cleaning* can be applied to remove noise and correct inconsistencies in data. *Data integration* merges data from multiple sources into a coherent data store such as a data warehouse. *Data reduction* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. *Data transformations* (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0.

- **Data cleaning** routines work to “clean” the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- **Data integration** is the merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data 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.
- Data reduction strategies include *dimensionality reduction* and *numerosity reduction*.
 - In **dimensionality reduction**, data encoding schemes are applied so as to obtain a reduced or “compressed” representation of the original data. Examples include data compression techniques (e.g., *wavelet transforms* and *principal components analysis*), *attribute subset selection* (e.g., removing irrelevant attributes), and *attribute construction* (e.g., where a small set of more useful attributes is derived from the original set).
 - In **numerosity reduction**, the data are replaced by alternative, smaller representations using parametric models (e.g., *regression* or *log-linear models*) or nonparametric models (e.g., *histograms*, *clusters*, *sampling*, or *data aggregation*).
- **Data Transformation:** In data transformation, the data are transformed or consolidated into forms appropriate for mining. In this preprocessing step, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand.

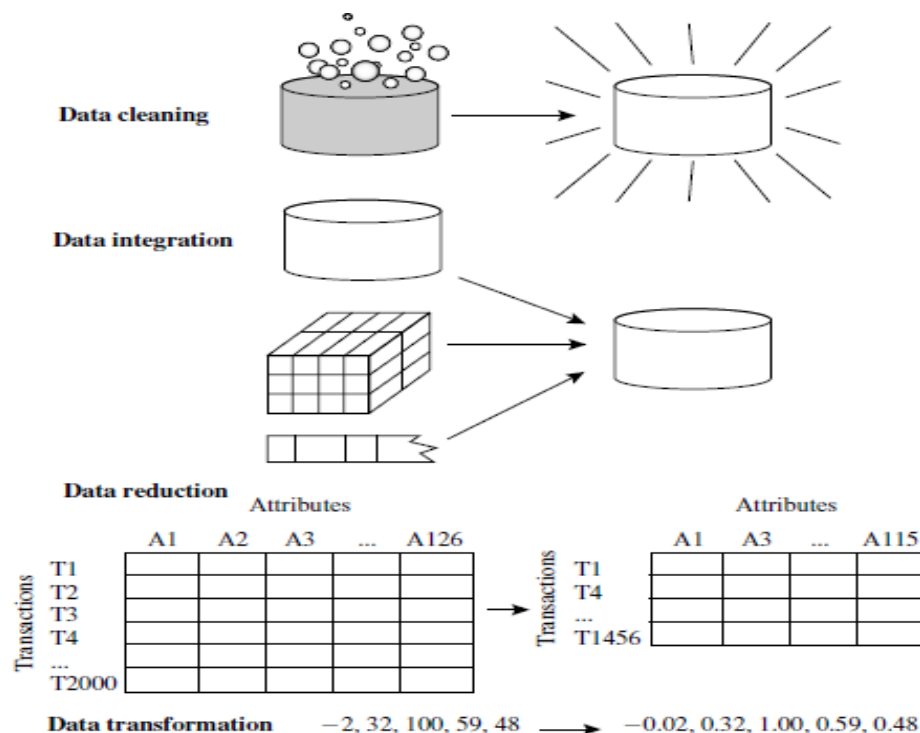


Figure 3.1 Forms of data preprocessing.

Data Cleaning

- Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.
- Basic methods for data cleaning are.
 - ❖ Handling missing values
 - ❖ Data smoothing technique

Handling Missing Values

- No recorded value for attribute of tuple is called missing value. Reasons for missing values may include:
 - The person originally asked to provide a value for the attribute refuses and/or finds that the information requested is not applicable (e.g., a *license number* attribute left blank by non drivers)
 - the data entry person does not know the correct value
 - the value is to be provided by a later step of the process.
- **Methods for filling in the missing values:**
 - **Ignore the tuple:** This is usually done when the class label is missing (assuming the mining task involves classification). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.
 - **Fill in the missing value manually:** In general, this approach is time consuming and may not be feasible given a large data set with many missing values.
 - **Use a global constant to fill in the missing value:** Replace all missing attribute values by the same constant such as a label like “*Unknown*” or If missing values are replaced by, say, “*Unknown*,” then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of “*Unknown*.” Hence, although this method is simple, it is not foolproof.
 - **Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value:** For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median
 - **Use the attribute mean or median for all samples belonging to the same class as the given tuple:** For example, if classifying customers according to *credit risk*, we may replace the missing value with the mean *income* value for customers in the same credit risk category as that of the given tuple. If the data distribution for a given class is skewed, the median value is a better choice.

- **Use the most probable value to fill in the missing value:** This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction.

Noisy Data

- **Noise** is a random error or variance in a measured variable.
- We can “smooth” out the data to remove the noise.
- Data smoothing techniques.
 - Binning
 - Regression
 - Outlier Analysis

Binning

- Binning methods smooth a sorted data value by consulting its “neighborhood,” that is, the values around it. The sorted values are distributed into a number of “buckets,” or *bins*. Because binning methods consult the neighborhood of values, they perform *local* smoothing.
- In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin
- **Smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median.
- In **smoothing by bin boundaries**, the minimum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value.

Sorted data for *price* (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34

Partition into (equal-frequency) bins:	
Bin 1:	4, 8, 15
Bin 2:	21, 21, 24
Bin 3:	25, 28, 34
Smoothing by bin means:	
Bin 1:	9, 9, 9
Bin 2:	22, 22, 22
Bin 3:	29, 29, 29
Smoothing by bin boundaries:	
Bin 1:	4, 4, 15
Bin 2:	21, 21, 24
Bin 3:	25, 25, 34

Figure 3.2 Binning methods for data smoothing.

- Above figure illustrates some binning techniques. In this example, the data for *price* are first sorted and then partitioned into *equal-frequency* bins of size 3 (i.e., each bin contains three values). In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9.

Regression:

- Data smoothing can also be done by regression, a technique that conforms data values to a function.
- *Linear regression* involves finding the “best” line to fit two attributes (or variables) so that one attribute can be used to predict the other.
- *Multiple linear regressions* is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

Outlier analysis:

- Outliers may be detected by clustering, for example, where similar values are organized into groups, or “clusters.” Intuitively, values that fall outside of the set of clusters may be considered outliers.

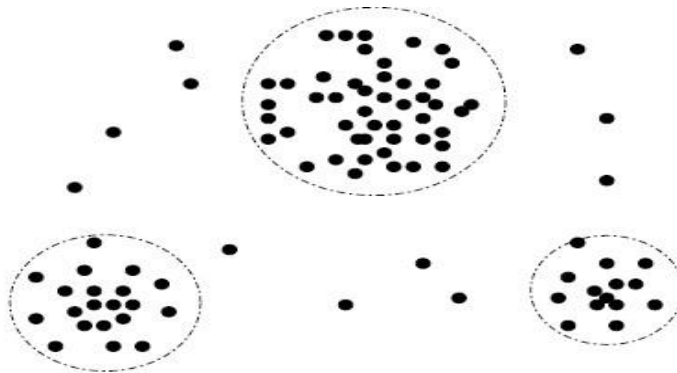


Figure 3.3 A 2-D customer data plot with respect to customer locations in a city, showing three data clusters. Outliers may be detected as values that fall outside of the cluster sets.

Data Integration

- **Data integration** combines data from multiple sources to form a coherent data store.
- Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data mining process.
- The resolution of semantic heterogeneity, metadata, correlation analysis, tuple duplication detection, and data conflict detection contribute to smooth data integration.
- Issues that must be considered during such integration include:
 - **Schema integration:** The metadata from the different data sources must be integrated in order to match up equivalent real-world entities. This is referred to as the entity identification problem.
 - **Handling redundant data:** Derived attributes may be redundant, and inconsistent attribute naming may also lead to redundancies in the resulting data set. Some redundancies can be detected by **correlation analysis**

- **Tuple Duplication Detection:** Duplications at the tuple level may occur and thus need to be detected and resolved.
- **Detection and resolution of data value conflicts:** Differences in representation, scaling, or encoding may cause the same real-world entity attribute values to differ in the data sources being integrated.

Data Transformation

→ In data transformation, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand. i.e.; in *data transformation*, the data are transformed or consolidated into forms appropriate.

Data Transformation Strategies:

Strategies for data transformation include the following:

1. **Smoothing**, which works to remove noise from the data. Techniques include binning, regression, and clustering.
2. **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.
3. **Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.
4. **Normalization**, where the attribute data are scaled so as to fall within a smaller range, such as -1.0 to 1.0, or 0.0 to 1.0.
5. **Discretization**, where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth*, *adult*, *senior*).
6. **Concept hierarchy generation for nominal data**, where attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*. Many hierarchies for nominal attributes are implicit within the database schema and can be automatically defined at the schema definition level.

Data Transformation by Normalization

- Normalization involves transforming the data to fall within a smaller or common range such as [-1, 1] or [0.0, 1.0].
- Normalizing the data attempts to give all attributes an equal weight.
- Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering. It is also useful when given no prior knowledge of the data.

Various methods for data normalization are:

- ❖ Min-max normalization
- ❖ z-score normalization using the standard deviation
- ❖ Normalization by decimal scaling.

Let A be a numeric attribute with n observed values, v_1, v_2, \dots, v_n .

- **Min-max normalization** performs a linear transformation on the original data. Suppose that $\min A$ and $\max A$ are the minimum and maximum values of an attribute, A . Min-max normalization maps a value, v_i , of A to v_i' in the range $[\text{new_min}_A, \text{new_max}_A]$ by computing

$$v_i' = \frac{v_i - \min A}{\max A - \min A} (\text{new_max}_A - \text{new_min}_A) + \text{new_min}_A.$$

Min-max normalization preserves the relationships among the original data values. It will encounter an “out-of-bounds” error if a future input case for normalization falls outside of the original data range for A .

Example 3.4 Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range $[0.0, 1.0]$. By min-max normalization, a value of \$73,600 for *income* is transformed to $\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$. ■

- In **z-score normalization** (or *zero-mean normalization*), the values for an attribute, A , are normalized based on the mean (i.e., average) and standard deviation of A . A value, v_i , of A is normalized to v_i' by computing

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A},$$

Where σ_A and \bar{A} are the mean and standard deviation, respectively, of attribute A .

This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.

Example 3.5 z-score normalization. Suppose that the mean and standard deviation of the values for the attribute *income* are \$54,000 and \$16,000, respectively. With z-score normalization, a value of \$73,600 for *income* is transformed to $\frac{73,600 - 54,000}{16,000} = 1.225$. ■

- **Normalization by decimal scaling** normalizes by moving the decimal point of values of attribute A . The number of decimal points moved depends on the maximum absolute value of A . A value, v_i , of A is normalized to v'_i by computing

$$v'_i = \frac{v_i}{10^j},$$

where j is the smallest integer such that $\max(|v'_i|) < 1$. i.e., number of digits in maximum absolute value of A .

Example: Suppose that the recorded values of A range from -986 to 917. The Maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1,000 (i.e., $j = 3$) so that -986 normalizes to -0.986 and 917 normalizes to 0.917.

Data Reduction

- Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data.
- That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.
- *Data reduction* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering.

Data Reduction Strategies

- **Dimensionality reduction** reduces the number of random variables or attributes under consideration. Methods include *wavelet transforms*, *principal components analysis*, *attribute subset selection*, and *attribute creation*.
- **Numerosity reduction** methods use parametric or non parametric models to obtain smaller representations of the original data. Parametric models store only the model parameters instead of the actual data. Examples include regression and log-linear models. Non parametric methods include histograms, clustering, sampling, and data cube aggregation.
- **Data compression** methods apply transformations to obtain a reduced or “compressed” representation of the original data. The data reduction is lossless if the original data can be reconstructed from the compressed data without any loss of information; otherwise, it is lossy.

Data Reduction Strategies:**▪ Dimensionality Reduction**

- Wavelet transforms
- Principal components analysis
- Attribute subset selection

▪ Numerosity Reduction

- Parametric methods
 - Regression
 - Log-linear models.
- Non parametric methods
 - Histograms
 - Clustering
 - Sampling
 - Data cube aggregation.

▪ Data Compression

Wavelet Transforms

- The **discrete wavelet transform (DWT)** is a linear signal processing technique that, when applied to a data vector X , transforms it to a numerically different vector, X' , of **wavelet coefficients**. The two vectors are of the same length.
- The usefulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients.
- The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines
- Wavelet transforms can be applied to multidimensional data such as a data cube. This is done by first applying the transform to the first dimension, then to the second, and so on.
- Wavelet transforms have many real world applications, including the compression of fingerprint images, computer vision, analysis of time- series data, and data cleaning.

Principal Components Analysis (PCA)

- Suppose that the data to be reduced consist of tuples or data vectors described by n attributes or dimensions. **Principal components analysis (PCA)**; also called the Karhunen-Loeve, or K-L, method) searches for k n - dimensional orthogonal vectors that can best be used to represent the data, where $k \leq n$. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction.

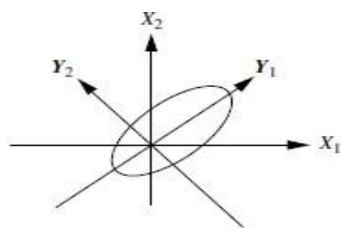


Figure 3.5 Principal components analysis. Y_1 and Y_2 are the first two principal components for the given data.

The basic procedure is as follows:

1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.
2. PCA computes k orthogonal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the *principal components*. The input data are a linear combination of the principal components.
3. The principal components are sorted in order of decreasing “significance” or strength. The principal components essentially serve as a new set of axes for

the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, below Figure shows the first two principal components, Y_1 and Y_2 , for the given set of data originally mapped to the axes X_1 and X_2 . This information helps identify groups or patterns within the data.

4. Because the components are sorted in decreasing order of “significance,” the data size can be reduced by eliminating the weaker components, that is, those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.
 - PCA can be applied to ordered and unordered attributes, and can handle sparse data and skewed data.
 - Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions.

Attribute Subset Selection

- Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant.
- **Attribute subset selection** reduces the data set size by removing irrelevant or redundant attributes (or dimensions).
- The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.
- Basic heuristic methods of attribute subset selection include the following:

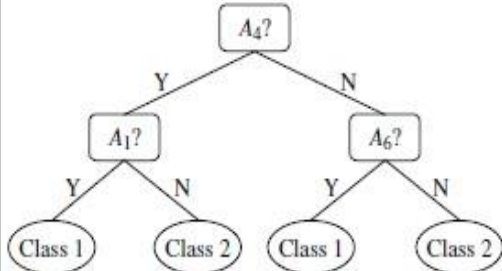
Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ $\Rightarrow \{A_1\}$ $\Rightarrow \{A_1, A_4\}$ \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_4, A_5, A_6\}$ \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$

Figure 3.6 Greedy (heuristic) methods for attribute subset selection.

1. **Stepwise forward selection:** The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

2. **Stepwise backward elimination:** The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
3. **Combination of forward selection and backward elimination:** The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
4. **Decision tree induction:** Decision tree algorithms (e.g., ID3, C4.5, and CART) were originally intended for classification. Decision tree induction constructs a flowchart like structure where each internal (non leaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the “best” attribute to partition the data into individual classes. When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

Regression and Log-Linear Models:

(Parametric Data Reduction)

- Regression and log-linear models can be used to approximate the given data.
- In (simple) **linear regression**, the data are modeled to fit a straight line. For example, a random variable, y (called a *response variable*), can be modeled as a linear function of another random variable, x (called a *predictor variable*), with the equation $y = wx + b$, where the variance of y is assumed to be constant.
- In the context of data mining, x and y are numeric database attributes. The coefficients, w and b (called *regression coefficients*), specify the slope of the line and the y -intercept, respectively. These coefficients can be solved for by the *method of least squares*, which minimizes the error between the actual line separating the data and the estimate of the line.
- **Multiple linear regression** is an extension of (simple) linear regression, which allows a response variable, y , to be modeled as a linear function of two or more predictor variables.
- **Log-linear models** approximate discrete multidimensional probability distributions. Given a set of tuples in n dimensions (e.g., described by n attributes), we can consider each tuple as a point in an n -dimensional space. Log-linear models can be used to estimate the probability of each point in a multidimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations

Histograms:

- Histograms use binning to approximate data distributions and are a popular form of data reduction.
- A **histogram** for an attribute, A , partitions the data distribution of A into

disjoint subsets, referred to as *buckets* or *bins*. If each bucket represents only a single attribute–value/frequency pair, the buckets are called *singleton buckets*. Often, buckets instead represent continuous ranges for the given attribute.

- **Example:** The following data are a list of prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 28, 28, 30, 30, 30.
- Figure 3.7 shows a histogram for the data using singleton buckets. To further reduce the data, it is common to have each bucket denote a continuous value range for the given attribute. In Figure 3.8, each bucket represents a different \$10 range for *price*.

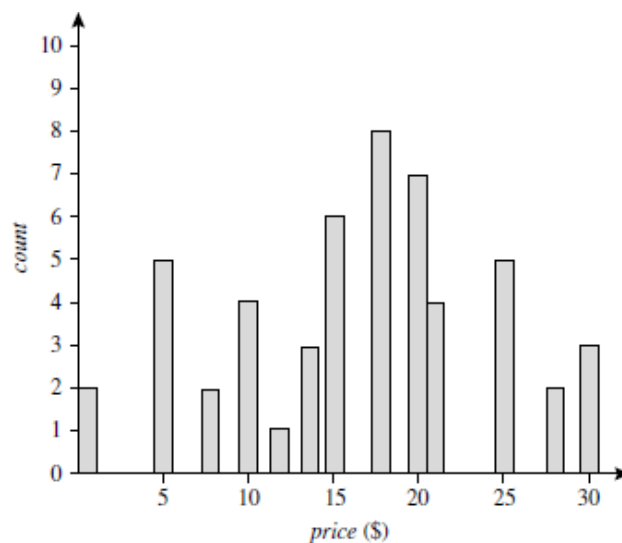


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

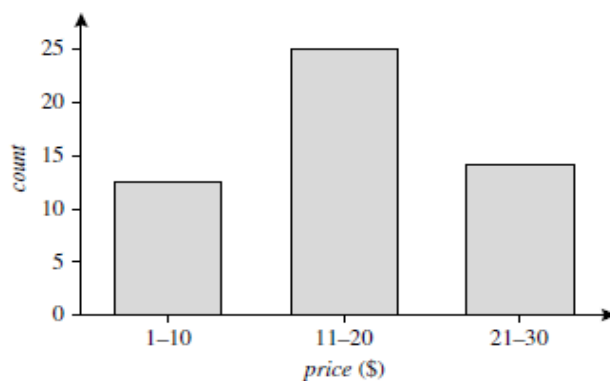


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of \$10.

There are several partitioning rules, including the following:

- ❖ **Equal-width:** In an equal-width histogram, the width of each bucket range is uniform (e.g., the width of \$10 for the buckets in Figure 3.8).
 - ❖ **Equal-frequency(or equal-depth):** In an equal-frequency histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (i.e., each bucket contains roughly the same number of contiguous data samples).
- Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data.
 - The histograms described before for single attributes can be extended for multiple attributes.
 - *Multidimensional histograms* can capture dependencies between attributes. These histograms have been found effective in approximating data with up to five attributes.
 - Singleton buckets are useful for storing high-frequency outliers.

Clustering:

- Clustering techniques consider data tuples as objects. They partition the objects into groups, or *clusters*, so that objects within a cluster are “similar” to one another and “dissimilar” to objects in other clusters.
- Similarity is commonly defined in terms of how “close” the objects are in space, based on a distance function.
- The “quality” of a cluster may be represented by its *diameter*, the maximum distance between any two objects in the cluster.
- **Centroid distance** is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the “average object,” or average point in space for the cluster).
- In data reduction, the cluster representations of the data are used to replace the actual data.
- The effectiveness of this technique depends on the data’s nature. It is much more effective for data that can be organized into distinct clusters than for smeared data.

Sampling:

- Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random data sample (or subset).
- Suppose that a large data set, D , contains N tuples. The most common ways that we could sample D for data reduction, as illustrated in following figure.

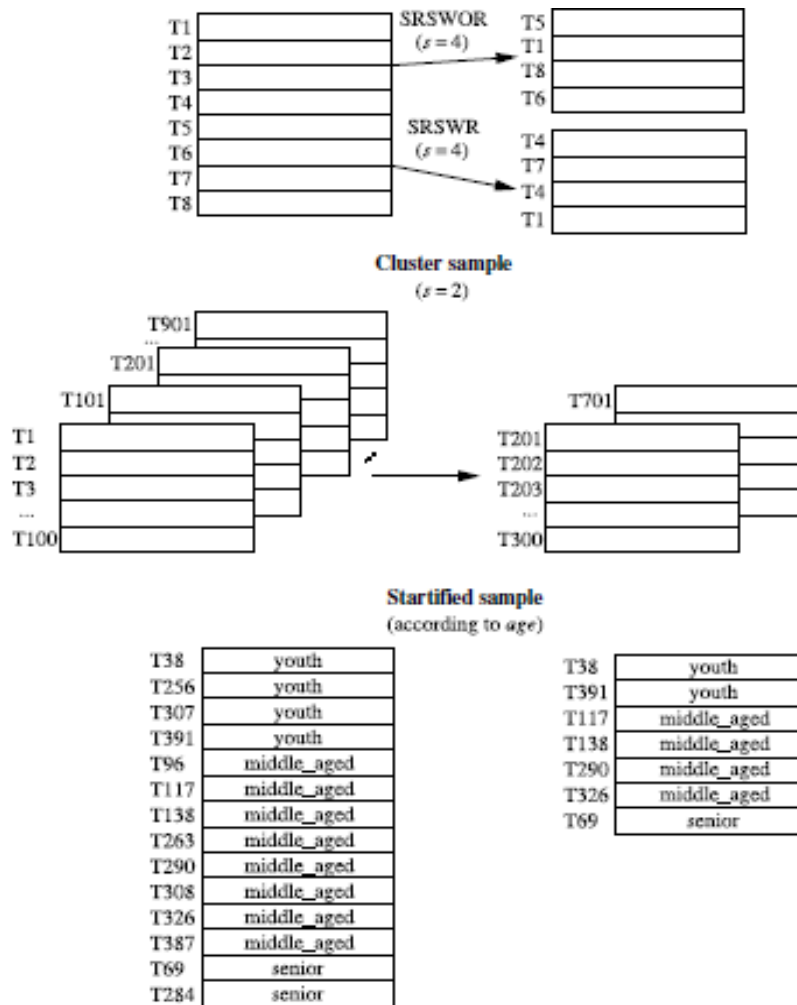


Figure 3.9 Sampling can be used for data reduction.

Simple random sample without replacement (SRSWOR):

→ This is created by drawing s of the N tuples from D ($s < N$), where the probability of drawing any tuple in D is $1/N$, that is, all tuples are equally likely to be sampled.

Simple random sample with replacement (SRSWR):

→ This is similar to SRSWOR, except that each time a tuple is drawn from D , it is recorded and then *replaced*. That is, after a tuple is drawn, it is placed back in D so that it may be drawn again.

Cluster sample:

→ If the tuples in D are grouped into M mutually disjoint “clusters,” then an SRS of s clusters can be obtained, where $s < M$. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

Data Cube Aggregation

- Data cubes store multidimensional aggregated information. Each cell holds an aggregate data value, corresponding to the data point multidimensional space.

The diagram illustrates the process of data cube aggregation. On the left, three stacked tables represent quarterly sales data for the years 2008, 2009, and 2010. Each table has columns for 'Quarter' and 'Sales'. The 2008 table shows specific values for Q1 through Q4. An arrow points from these tables to a single table on the right, which represents the aggregated annual sales data, with columns for 'Year' and 'Sales'.

Year 2010	
Quarter	Sales
Q1	0
Q2	0
Q3	0
Q4	0

Year 2009	
Quarter	Sales
Q1	0
Q2	0
Q3	0
Q4	0

Year 2008	
Quarter	Sales
Q1	\$224,000
Q2	\$408,000
Q3	\$350,000
Q4	\$586,000

Year	Sales
2008	\$1,568,000
2009	\$2,356,000
2010	\$3,594,000

Figure 3.10 Sales data for a given branch of *AllElectronics* for the years 2008 through 2010. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.

For example, Figure 3.11 shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch.

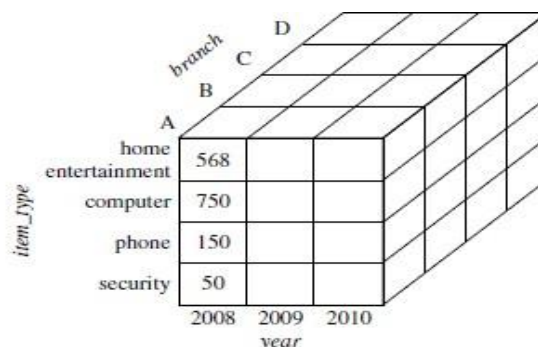


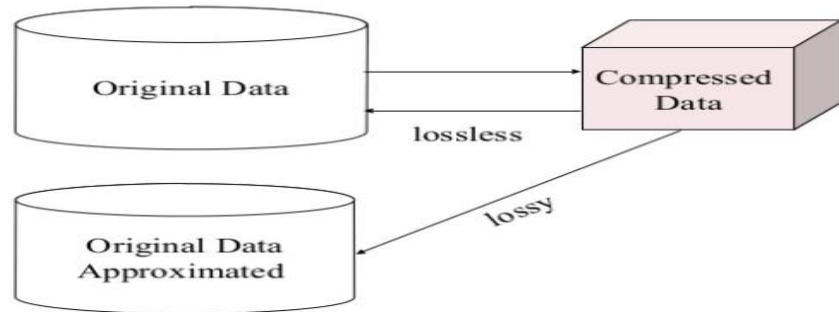
Figure 3.11 A data cube for sales at *AllElectronics*.

- Data cubes provide fast access to precomputed, summarized data, thereby benefiting online analytical processing as well as data mining.
- The cube created at the lowest abstraction level is referred to as the **base cuboid**. The base cuboid should correspond to an individual entity of interest such as *sales* or *customer*. In other words, the lowest level should be usable, or useful for the analysis.
- A cube at the highest level of abstraction is the **apex cuboid**. For the sales data in Figure 3.11, the apex cuboid would give one total—the total *sales* for all three years, for all item types, and for all branches.
- Data cubes created for varying levels of abstraction are often referred to as *cuboids*, so that a data cube may instead refer to a *lattice of cuboids*.

Data Compression

- **Data Compression** specifically refers to a **data reduction** method by which files are shrunk at the bit level.
- **Data Compression** works by using formulas or algorithms to **reduce** the number of bits needed to represent the **data**.
- The data reduction is *lossless* if the original data can be reconstructed from the compressed data without any loss of information; otherwise, it is *lossy*.

Data Compression



String Compression:

- There are extensive theories & well-tuned algorithms
- Typically lossless
- But only limited manipulation is possible

Audio/video Compression:

- Typically lossy compression with progressive refinement
- Some time small fragments of signal can be constructed without reconstruction of the whole.
- Dimensionality Reduction and Numerosity Reduction techniques can also be consider form of data compression.

Data Discretization and Concept Hierarchy Generation for Numerical Data

- **Data discretization** transforms numeric data by mapping values to interval or concept labels. Such methods can be used to automatically generate *concept hierarchies* for the data, which allows for mining at multiple levels of granularity.
- Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up).
- If the discretization process uses class information, then we say it is *supervised discretization*. Otherwise, it is *unsupervised*.
- If the process starts by first finding one or a few points (called *split points* or *cut points*) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called *top-down discretization* or *splitting*. This contrasts with *bottom-up discretization* or *merging*, which starts by

considering all of the continuous values as potential split- points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals.

- A concept hierarchy for a given numeric attribute defines a discretization of the attribute. i.e organize attributes or attributes values into different level of abstraction.
- *Concept hierarchies* may exist for each attribute, allowing the analysis of data at multiple abstraction levels. For example, a hierarchy for *branch* could allow branches to be grouped into regions, based on their address.
- Concept hierarchies can be used to reduce the data collecting and replacing low-level concepts(such as numeric value for the attribute age) by higher level concepts (such as young, middle-aged, or senior).
- Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert.
- Several discretization methods can be used to automatically generate or dynamically refine concept hierarchies for numerical attributes.
- Concept hierarchy generation is also form of data reduction.
- Discretization techniques include:
 - ❖ Binning
 - ❖ Histogram Analysis
 - ❖ Cluster Analysis
 - ❖ Decision Tree Analysis
 - ❖ Correlation Analysis.

Discretization by Binning:

- Binning is a top-down splitting technique based on a specified number of bins.
- Binning methods for data smoothing are also used as discretization methods for data reduction and concept hierarchy generation.
- For example, attribute values can be discretized by applyig equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions to generate concept hierarchies.
- Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers.

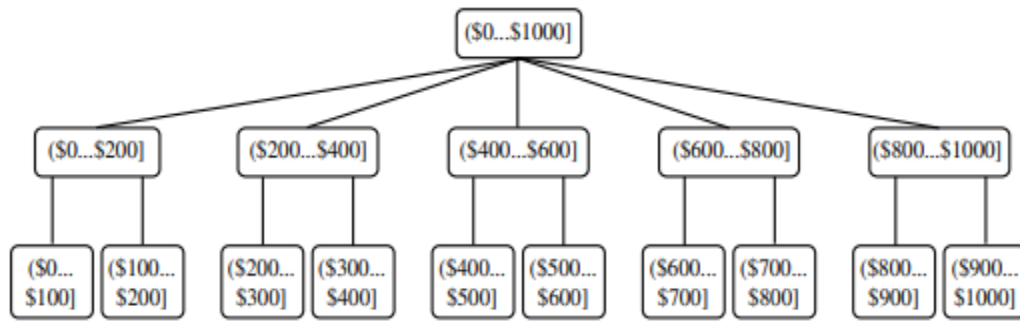
Discretization by Histogram Analysis:

- Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information.
- A histogram partitions the values of an attribute, *A*, into disjoint ranges called *buckets* or *bins*. Various partitioning rules can be used to define histograms.

- In an *equal-width* histogram, for example, the values are partitioned into equal-size partitions or ranges.
- With an *equal-frequency* histogram, the values are partitioned so that, ideally, each partition contains the same number of data tuples.
- The histogram analysis algorithm can be applied recursively to each partition in order to automatically generate a multilevel concept hierarchy, with the procedure terminating once a pre specified number of concept levels has been reached. A *minimum interval size* can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level.
- Histograms can also be partitioned based on cluster analysis of the data distribution, as described next.

Discretization by Cluster, Decision Tree, and Correlation Analyses:

- Clustering, decision tree analysis, and correlation analysis can be used for data discretization.
- Cluster analysis is a popular data discretization method. A clustering algorithm can be applied to discretize a numeric attribute, *A*, by partitioning the values of *A* into clusters or groups.
- Clustering takes the distribution of *A* into consideration, as well as the closeness of data points, and therefore is able to produce high-quality discretization results.
- Clustering can be used to generate a concept hierarchy for *A* by following either a top-down splitting strategy or a bottom-up merging strategy, where each cluster forms a node of the concept hierarchy. In the former, each initial cluster or partition may be further decomposed into several subclusters, forming a lower level of the hierarchy. In the latter, clusters are formed by repeatedly grouping neighboring clusters in order to form higher-level concepts.
- Techniques to generate decision trees for classification can be applied to discretization. Such techniques employ a top-down splitting approach.
- Measures of correlation can be used for discretization. *ChiMerge* is χ^2 - based discretization method. This method employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively.



A concept hierarchy for the attribute *price*, where an interval $(\$X \dots \$Y]$ denotes the range from $\$X$ (exclusive) to $\$Y$ (inclusive).

Concept Hierarchy Generation for Categorical Data

- Nominal (categorical) attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include *geographic location*, *job category*, and *item type*.
- In concept hierarchy, attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*.
- Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, many hierarchies are implicit within the database schema and can be automatically defined at the schema definition level.
- The concept hierarchies can be used to transform the data into multiple levels of granularity.
- Information at the schema level and on attribute–value counts can be used to generate concept hierarchies for nominal data.
- Transforming nominal data with the use of concept hierarchies allows higher-level knowledge patterns to be found. It allows mining at multiple levels of abstraction, which is a common requirement for data mining applications.
- For example, data mining patterns regarding sales may be found relating to specific regions or countries, in addition to individual branch locations

Methods for the generation of concept hierarchies for nominal data:

1.Specification of a partial ordering of attributes explicitly at the schema level by users or experts: Concept hierarchies for nominal attributes or dimensions typically involve a group of attributes. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level. For example, suppose that a relational database contains the following group of attributes: *street*, *city*, *province or state*, and *country*. Similarly, a data warehouse *location* dimension may contain the same attributes. A hierarchy can be defined by specifying the total ordering among

these attributes at the schema level such as *street* < *city* < *province or state* < *country*.

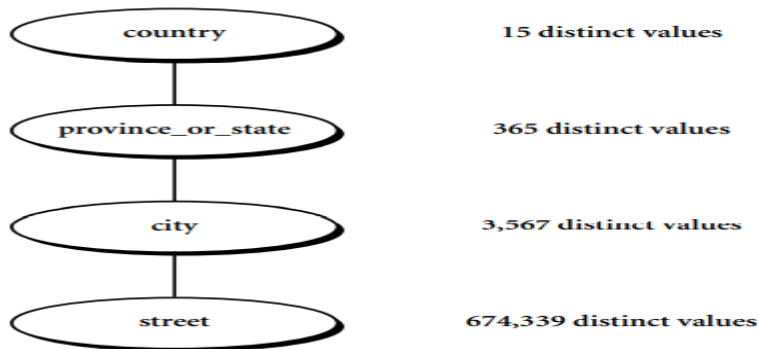
2.Specification of a portion of a hierarchy by explicit data grouping: This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it is unrealistic to define an entire concept hierarchy by explicit value enumeration. On the contrary, we can easily specify explicit groupings for a small portion of intermediate-level data. For example, after specifying that *province* and *country* form a hierarchy at the schema level, a user could define some intermediate levels manually, such as “{*Alberta, Saskatchewan, Manitoba*} □ *prairies- Canada*” and “{*British Columbia, prairies-Canada*} □ *Western -Canada*.”

3.Specification of a set of attributes, but not of their partial ordering: A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly state their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy. Consider the observation that since higher- level concepts generally cover several subordinate lower-level concepts, an attribute defining a high concept level (e.g., *country*) will usually contain a smaller number of distinct values than an attribute defining a lower concept level (e.g., *street*). Based on this observation, a concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set. The attribute with the most distinct values is placed at the lowest hierarchy level. The lower the number of distinct values an attribute has, the higher it is in the generated concept hierarchy.

Example 3.7 Concept hierarchy generation based on the number of distinct values per attribute.

Suppose a user selects a set of location-oriented attributes—*street*, *country*, *province_or_state*, and *city*—from the *AllElectronics* database, but does not specify the hierarchical ordering among the attributes.

A concept hierarchy for *location* can be generated automatically, as illustrated in Figure 3.13. First, sort the attributes in ascending order based on the number of distinct values in each attribute. This results in the following (where the number of distinct values per attribute is shown in parentheses): *country* (15), *province_or_state* (365), *city* (3567), and *street* (674,339). Second, generate the hierarchy from the top down according to the sorted order, with the first attribute at the top level and the last attribute at the bottom level. Finally, the user can examine the generated hierarchy, and when necessary, modify it to reflect desired semantic relationships among the attributes. In this example, it is obvious that there is no need to modify the generated hierarchy. ■



Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

- 4. Specification of only a partial set of attributes:** Sometimes a user can be careless when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in the hierarchy specification. For example, instead of including all of the hierarchically relevant attributes for *location*, the user may have specified only *street and city*. To handle such partially specified hierarchies, it is important to embed data semantics in the database schema so that attributes with tight semantic connections can be pinned together. In this way, the specification of one attribute may trigger a whole group of semantically tightly linked attributes to be “dragged in” to form a complete hierarchy.

Tutorial Questions

1. What is data mining? Why we require data mining? Explain data mining as a step in the process of knowledge discovery with neat diagram.
(or) Define data mining? Why do we mine data? Discuss in detail about the steps of knowledge discovery with diagram.
(or) **What is data mining? Why do we mine data? Describe the steps involved in data mining when viewed as a process of KDD.**
(or) Explain the steps involved in the Data Mining Process. Give the sketch of the KDD process.
2. **List and briefly describe the different data mining functionalities with real life example of each.**
3. **Discuss the major challenging issues in data mining research.**
(or) Explain major issues in data mining
(or) Briefly outline the major challenges in data mining
4. Define data object and attribute. Describe different attribute types with examples.
(or) What is data set? Describe different characteristics and types of data sets used in data mining.

5. **With an example, briefly outline similarity/dissimilarity techniques for different types of data.**
(or) Explain the different measures of proximity for different types of attributes.
(or) Briefly outline how to compute the similarity/dissimilarity between objects of different types.
6. Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):
 - (a) Compute the *Euclidean distance* between the two objects.
 - (b) Compute the *Manhattan distance* between the two objects.
 - (c) Compute the *Minkowski distance* between the two objects, using $q=3$.
 - (d) Compute the *supremum distance* between the two objects.
7. **Why preprocess the data? Briefly describe the major steps involved in data pre-processing.**
8. Explain various data pre-processing methods with appropriate examples.
9. What is the need for data cleaning? Explain the steps in the process of data cleaning. Discuss briefly about data cleaning techniques.
10. Describe the various approaches to remove the noisy data from the original data.
(or) What is noisy data? Explain the binning methods for data smoothening.
11. What is data integration? Discuss the issues to consider during data integration.
12. **What is data normalization? Explain different normalization methods with examples.(or)** Illustrate the Data Transformation by Normalization.
13. What is data reduction? Describe briefly the strategies for data reduction.
14. **What are the techniques used to produce smaller forms of data representation using numerosity reduction? Explain each with an example**
15. What is attribute subset selection? Describe heuristic methods of attribute subset selection.
16. What is concept hierarchy generation? Describe the various methods for automatic generation of concept hierarchies for categorical data.
17. What is need of dimensionality reduction? Describe briefly different methods for dimensionality reduction.
18. Why correlation analysis is useful? How correlation coefficient is computed?
19. **What is data discretization? Describe different techniques for data discretization.**

20. Use the three methods below to normalize the following group of data: **200, 300, 400, 600, 1000**
- Min-max normalization by setting $\min = 0$ and $\max = 1$
 - z-score normalization
 - Normalization by decimal scaling
21. Suppose a group of 12 *sales price* records has been sorted as follows: **5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215**. Partition them into three bins by each of the following methods:
- equal-frequency (equal-depth) partitioning
 - equal-width partitioning
 - clustering
22. Using sorted data for *price* (in dollars): **4, 8, 15, 21, 21, 24, 25, 28, 34**
- Use *smoothing by bin means* to smooth these data, using a bin depth of 3.
 - Use *smoothing by bin medians* to smooth these data, using a bin depth of 3.
 - Use *smoothing by bin boundaries* to smooth these data, using a bin depth of 3