UNIT –I

1.1 Introduction

Topics covered:

- Fundamentals of data mining
- Kinds of patterns can be mined
- Technologies used
- Applications and Issues in Data Mining

INTRODUCTION

Fundamentals of Data Mining

1.1.1 Why Data Mining?

We live in a world where vast amounts of data are collected daily. Analyzing such data is an important need. Data mining can be viewed as a result of the natural evolution of information technology.

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it.

Extraction of information is not the only process we need to perform; data mining also involves other processes such as Data Cleaning, Data Integration, Data Transformation, Data Mining, Pattern Evaluation and Data Presentation. Once all these processes are over, we would be able to use this information in many applications such as Fraud Detection, Market Analysis, Production Control, Science Exploration, etc.

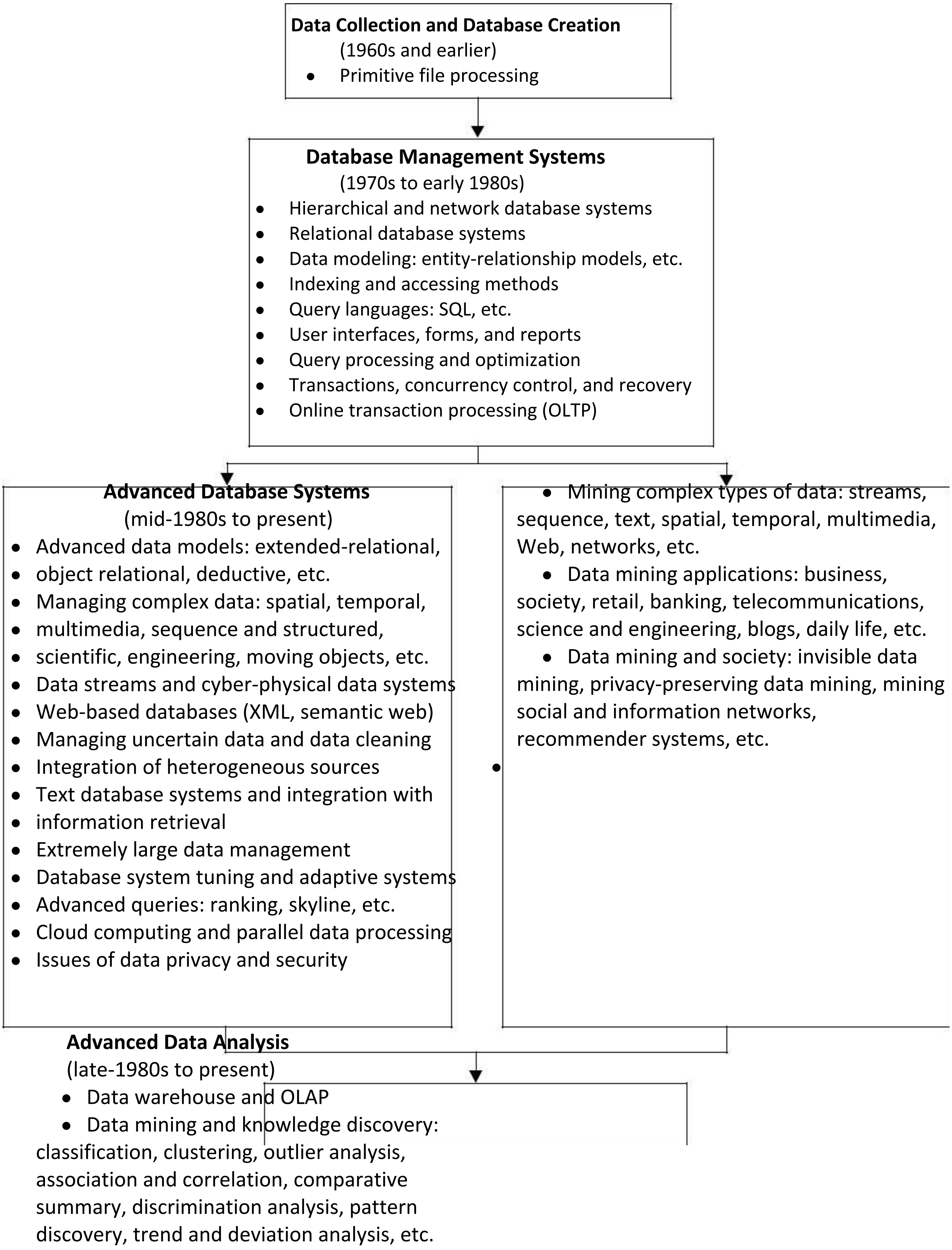
1.1.1.1 Moving toward the Information Age

“We are living in the information age” is a popular saying; however, we are actually living in the data age. Terabytes or petabytes of data pour into our computer networks, the World Wide Web, and various data storage devices every day from business,

> **Consider a** search engine (e.g., Google) receives hundreds of millions of queries every day. Each query can be viewed as a transaction where the user describes her/his information need. some patterns found in user search queries can disclose invaluable knowledge that cannot be obtained by reading individual data items alone. say, Google’s *Flu Trends* uses specific search terms as indicators of flu activity. It found a close relationship between the number of people who search for flu-related information and the number of people who actually have flu symptoms. A pattern emerges when all of the search queries related to flu are aggregated. Using aggregated Google search data, *Flu Trends* can estimate flu activity up to two weeks faster than traditional systems can. This example shows how data mining can turn a large collection of data into knowledge that can help meet a current global challenge.

1.1.1.2 Data Mining as the Evolution of Information Technology

Data mining can be viewed as a result of the natural evolution of information technology



Data Collection and Database Creation
(1960s and earlier)

- Primitive file processing

Database Management Systems
(1970s to early 1980s)

- Hierarchical and network database systems
- Relational database systems
- Data modeling: entity-relationship models, etc.
- Indexing and accessing methods
- Query languages: SQL, etc.
- User interfaces, forms, and reports
- Query processing and optimization
- Transactions, concurrency control, and recovery
- Online transaction processing (OLTP)

Advanced Database Systems
(mid-1980s to present)

- Advanced data models: extended-relational,
- object relational, deductive, etc.
- Managing complex data: spatial, temporal,
- multimedia, sequence and structured,
- scientific, engineering, moving objects, etc.
- Data streams and cyber-physical data systems
- Web-based databases (XML, semantic web)
- Managing uncertain data and data cleaning
- Integration of heterogeneous sources
- Text database systems and integration with
- information retrieval
- Extremely large data management
- Database system tuning and adaptive systems
- Advanced queries: ranking, skyline, etc.
- Cloud computing and parallel data processing
- Issues of data privacy and security

- Mining complex types of data: streams, sequence, text, spatial, temporal, multimedia, Web, networks, etc.
- Data mining applications: business, society, retail, banking, telecommunications, science and engineering, blogs, daily life, etc.
- Data mining and society: invisible data mining, privacy-preserving data mining, mining social and information networks, recommender systems, etc.

Advanced Data Analysis
(late-1980s to present)

- Data warehouse and OLAP
- Data mining and knowledge discovery: classification, clustering, outlier analysis, association and correlation, comparative summary, discrimination analysis, pattern discovery, trend and deviation analysis, etc.

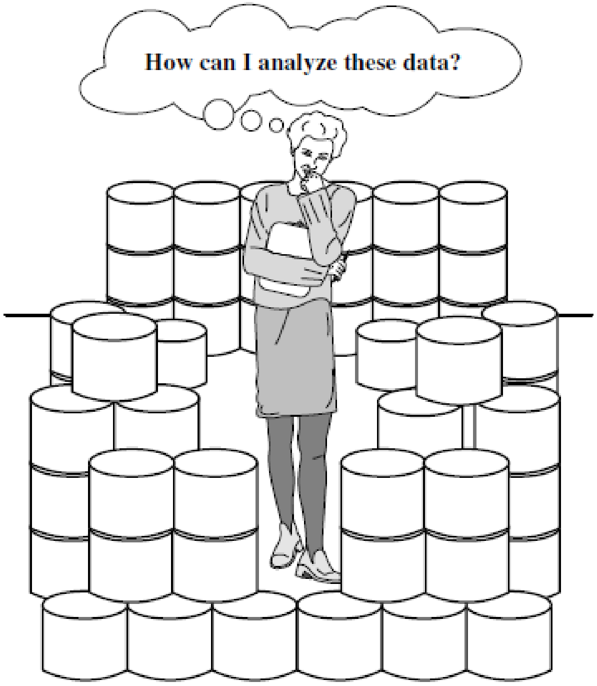
Future Generation of Information Systems

(Present to future)

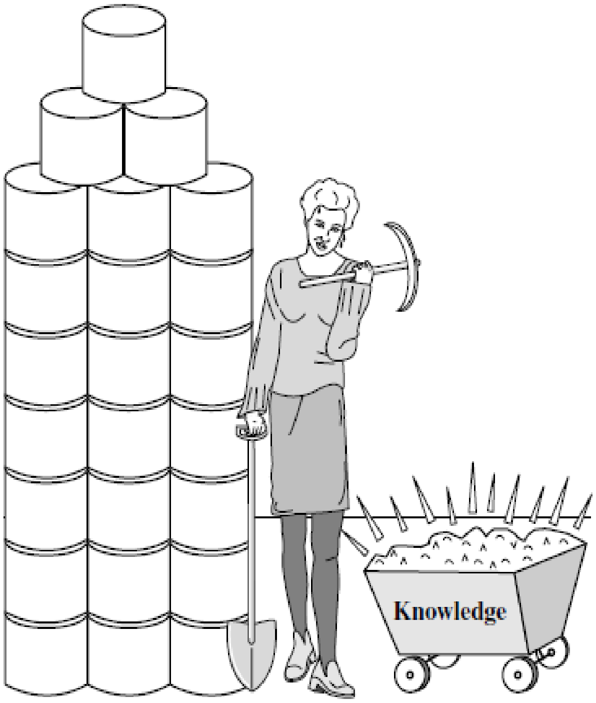
Fig 1.1The evolution of database system technology.

One emerging data repository architecture is the **data warehouse**. This is a repository of multiple heterogeneous data sources organized under a unified schema at a single site to facilitate management decision making. Data warehouse technology includes data cleaning, data integration, and online analytical processing (OLAP)—that is, analysis techniques with functionalities such as summarization, consolidation, and aggregation, as well as the ability to view information from different angles.

Huge volumes of data have been accumulated beyond databases and data warehouses. During the 1990s, the World Wide Web and web-based databases (e.g., XML databases) began to appear. Internet-based global information bases, such as the WWW and various kinds of interconnected, heterogeneous databases, have emerged and play a vital role in the information industry.



The world is data rich but information poor.



Data mining—searching for knowledge (interesting patterns) in data.

The large quantity of data, coupled with the need for powerful data analysis tools, has been described as a 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 the human ability for comprehension without powerful tools. As a result, data collected in large data repositories become “data tombs” that are seldom visited

1.1.2 What Is Data Mining?

Data mining is the *process* of discovering interesting patterns, previously unknown and knowledge from *large* amounts of data. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically.

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.***

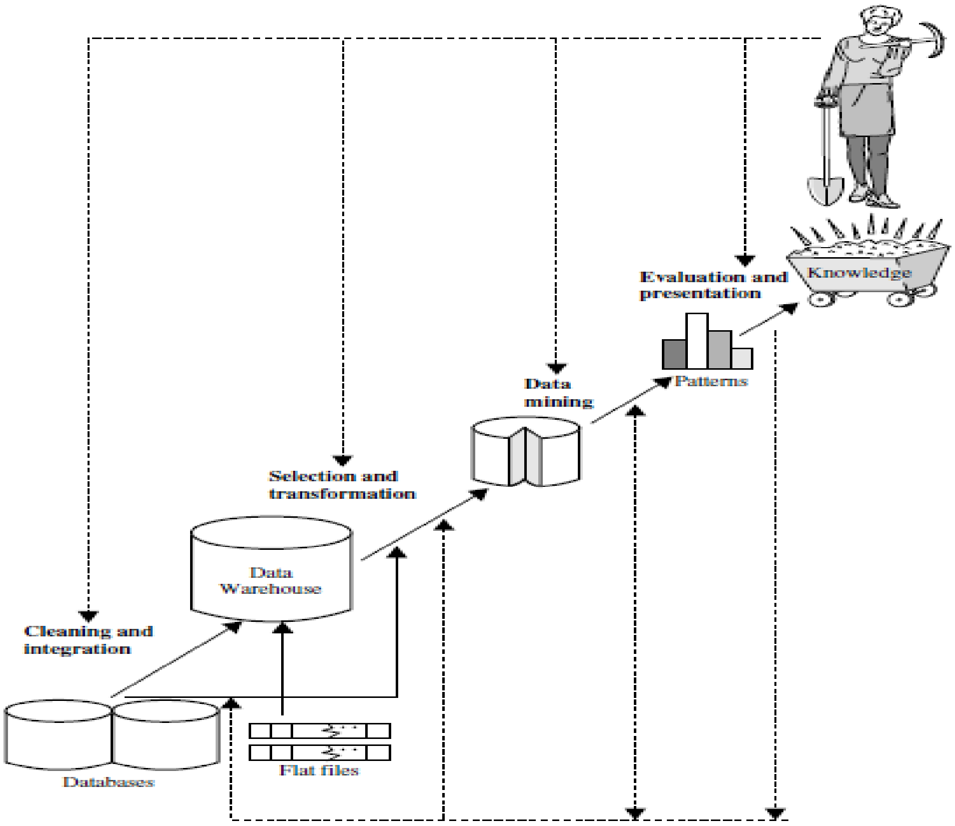
Data mining has a synonym popularly used term, **knowledge discovery from data**, or **KDD in which** data mining is just an essential step in the process of knowledge discovery.

Knowledge discovery from data - KDD

The knowledge discovery process is an iterative sequence of the following steps:

1. Data cleaning (to remove noise and inconsistent data)
2. Data Integration (where multiple data sources may be combined **4. Data transformation** (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)
3. Data selection (where data relevant to the analysis task are retrieved from the database)
4. Data transformation (where data are transformed and consolidated into forms appropriate for
mining by performing summary or aggregation operations)
5. Data mining (an essential process where intelligent methods are applied to extract data patterns)
6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on *interestingness measures*)
7. Knowledge presentation (where visualization and knowledge representation techniques are used
to present mined knowledge to users)

The KDD process can be diagramatially shown below



Steps 1 through 4 are different forms of data preprocessing, where data are prepared for mining.
The data mining step may interact with the user or a knowledge base.
The interesting patterns are presented to the user and may be stored as new knowledge in the knowledge base. The above diagram shows data mining as one step in the knowledge discovery process.

1.1.3 What Kinds of Data Can Be Mined?

As a general technology, 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

1. Database data
2. Data warehouse data and
3. transactional data

1.1.3.1 Database Data

A database system, also called a **database management system** (**DBMS**), consists of a collection of interrelated data, known as a **database**, and a set of software programs to manage and access the data. The software programs provide mechanisms for defining database structures and data storage; for specifying and managing concurrent, shared, or distributed data access; and for ensuring consistency and security of the information stored despite system crashes or attempts at unauthorized access.

A **relational database** is a collection of **tables**, each of which is assigned a unique name. Each table consists of a set of **attributes** (columns or fields) and usually stores a large set of **tuples** (records or rows). Each tuple in a relational table represents an object identified by a unique key and described by a set of attribute values. A semantic data model, such as an **entity-relationship (ER)** data model, is often constructed for relational databases. An ER data model represents the database as a set of entities and their relationships.

Example: consider a relational database for ***AllElectronics***. The company is described by the **following relation tables: customer, item, employee, and branch**.

The relation **customer** consists of a set of attributes describing the customer information, including a unique customer identity number (cust ID), customer name, address, age, occupation, annual income, credit information, and category.

Similarly, each of the relations item, employee, and branch consists of a set of attributes describing the properties of these entities.

Tables can also be used to represent the relationships between or among multiple entities. In our example, these include *purchases* (customer purchases items, creating a sales transaction handled by an employee), *items sold* (lists items sold in a given transaction), and *works at* (employee works at a branch of *AllElectronics*).

Customer (.cust ID, name, address, age, occupation, annual income, credit information, category, . . .)

Item (.item ID, brand, category, type, price, place made, supplier, cost, . . .)

Employee(empl ID, name, category, group, salary, commission, . . .)

branch (.branch ID, name, address, . . .)

purchases (.trans ID, cust ID, empl ID, date, time, method paid, amount)

item_ sold (trans ID, item ID, qty)

works_at (empl ID, branch ID)

Relational schema for a relational database, *AllElectronics*.

Relational data can be accessed by **database queries** written in a relational query language (e.g., SQL) or with the assistance of graphical user interfaces. A query allows retrieval of specified subsets of the data.

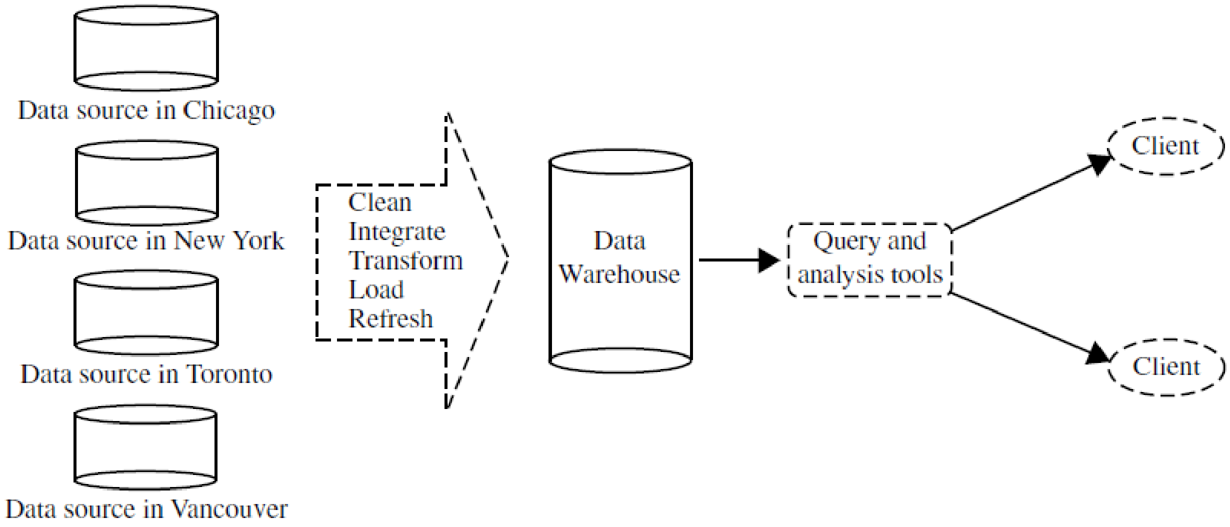
Relational languages also use aggregate functions such as sum, avg (average), count, max (maximum), and min (minimum). Using aggregates allows you to ask: ***"Show me the total sales of the last month, grouped by branch,"*** **or** ***"How many sales transactions occurred in the month of December?"*** **or** ***"Which salesperson had the highest sales?"***

When **mining relational databases**, we can go further by *searching for trends* or *data patterns*. For example, data mining systems can analyze **customer data to predict the credit risk of new customers based on their income, age, and previous credit information**. Data mining systems may also detect deviations—that is, **items with sales that are far from those expected in comparison with the previous year**. Such deviations can then be further investigated. For example, data mining may discover that there has been a change in packaging of an item or a significant increase in price.

Relational databases are one of the most commonly available and richest information repositories, and thus they are a major data form in the study of data mining

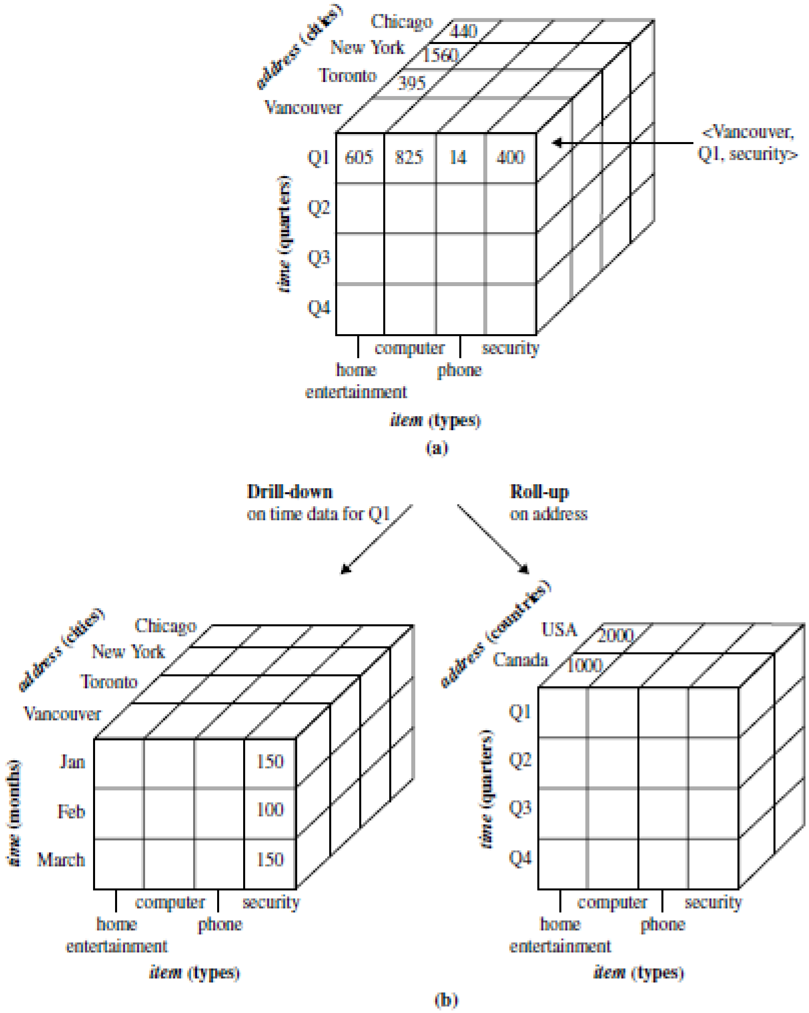
1.1.3.2. Data Warehouse Data

A **data warehouse** is a repository of information collected from multiple sources, stored under a unified schema, and usually residing at a single site. Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing.
To facilitate decision making, the data in a data warehouse are organized around *major subjects* (e.g., customer, item, supplier, and activity). The data are stored to provide information from a *historical perspective*, such as in the past 6 to 12 months, and are typically *summarized*.
For example, rather than storing the details of each sales transaction, the data warehouse may store a summary of the transactions per item type for each store or, summarized to a higher level, for each sales region.



Typical framework of a data warehouse for *AllElectronics.*

A data warehouse is usually modeled by a multidimensional data structure, called a **data cube**, in which each **dimension** corresponds to an attribute or a set of attributes in the schema, and each **cell** stores the value of some aggregate measure such as *count* or *sum.sales amount*.
A data cube provides a multidimensional view of data and allows the precomputation and fast access of summarized data.



A multidimensional data cube, commonly used for data warehousing, (a) showing summarized data for *AllElectronics* and (b) showing summarized data resulting fromdrill-down and roll-up operations on the cube in (a). For improved readability, only some of the cube cell values are shown.

1.1.3.3. Transactional Data

In general, each record in a **transactional database** captures a transaction, such as a customer's purchase, a flight booking, or a user's clicks on a web page.
A transaction typically includes a unique transaction identity number (*trans ID*) and a list of the **items** making up the transaction, such as the items purchased in the transaction.
A transactional database may have additional tables, which contain other information related to the transactions, such as item description, information about the salesperson or the branch, and so on.
Example transaction list of Items : which items are sold together – frequent itemsets/ market basket analysis.

trans ID	*list of item IDs*
T100	I1, I3, I8, I16
T200	I2, I8
….	….

1.1.3.4 **Other Kinds of Data**

Besides relational database data, data warehouse data, and transaction data, there are many other kinds of data that have versatile forms and structures and rather different semantic meanings. Such kinds of data can be seen in many applications: time-related or sequence data (e.g., historical records, stock exchange data, and time-series and biological sequence data), data streams (e.g., video surveillance and sensor data, which are continuously transmitted), spatial data (e.g., maps), engineering design data (e.g., the design of buildings, system components, or integrated circuits), hypertext and multimedia data (including text, image, video, and audio data), graph and networked data (e.g., social and information networks), and the Web (a huge, widely distributed information repository made available by the Internet). These applications bring about new challenges, like how to handle data carrying special structures (e.g., sequences, trees, graphs, and networks) and specific semantics (such as ordering, image, audio and video contents, and connectivity), and how to mine patterns that carry rich structures and semantics.

1.1.4 What Kinds of Patterns Can Be Mined

There are a number of ***data mining functionalities***. These include

- **characterization and discrimination,**
- **The mining of frequent patterns, associations, and correlations,**
- **classification and regression,**
- **clustering analysis and outlier analysis.**

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:

1. **descriptive** and
2. **predictive.**

Descriptive mining tasks characterize properties of the data in a target data set and the **Predictive mining** tasks perform induction on the current data in order to make predictions.

Data mining functionalities, and the kinds of patterns they can discover, are described below.

1.1.4.1 Class/Concept Description: Characterization and Discrimination

Data entries can be associated with classes or concepts. For example, in the *AllElectronics* store, classes of items for sale include *computers* and *printers*, and concepts of customers include *bigSpenders* and *budgetSpenders*. It can be useful to describe individual classes and concepts in summarized, concise, and yet precise terms. Such descriptions of a class or a concept are called **class/concept descriptions**.

These descriptions can be derived using (1) *data characterization*, by summarizing the data of the class under study (often called the **target class**) in general terms, or (2) *data discrimination*, by comparison of the target class with one or a set of comparative classes (often called the **contrasting classes**), or (3) both data characterization and discrimination.

Data characterization is a summarization of the general characteristics or features of a target class of data. The data corresponding to the user-specified class are typically collected by a query. For example, to study the characteristics of software products with sales that increased by 10% in the previous year, the data related to such products can be collected by executing an SQL query on the sales database.

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**).

Example :Data characterization. A customer relationship manager at *AllElectronics* may order the
following data mining task: *Summarize the characteristics of customers who spend more than $5000 a year at AllElectronics*. 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.

Data discrimination is a comparison of the general features of the target class data objects against the general features of objects from one or multiple contrasting classes. The target and contrasting classes can be specified by a user, and the corresponding data objects can be retrieved through database queries.

For example, a user may want to compare the general features of software products with sales that increased by 10% last year against those with sales that decreased by at least 30% during the same period. The methods used for data discrimination are similar to those used for data characterization.

Discrimination descriptions expressed in the form of rules are referred to as **discriminant rules**.

Example Data discrimination. A customer relationship manager at *AllElectronics* may want to compare two groups of customers—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.

1.1.4.2 Mining Frequent Patterns, Associations, and Correlations

Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including frequent itemsets, frequent sub-sequences (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 Association analysis. Suppose that, as a marketing manager at *AllElectronics*, you want to know which items are frequently purchased together (i.e., within the same transaction). An example of such a rule, mined fromthe *AllElectronics* 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. This association rule involves a single attribute or predicate (i.e., *buys*) that repeats. 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 => software* [1%, 50%].”

Suppose, instead, that we are given the *AllElectronics* relational database related to purchases. A data mining system may find association rules like

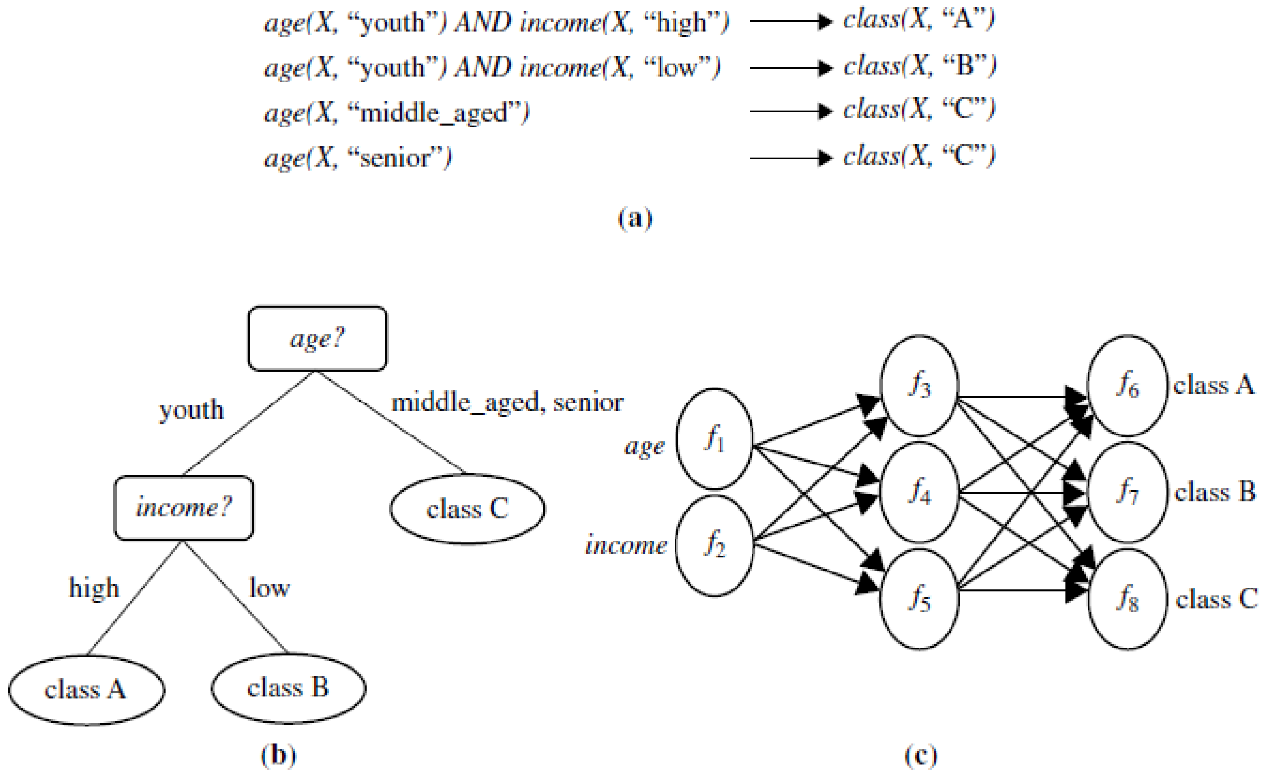
age.X, “20..29”) ∧ *income(X*, “40K..49K”) => *buys(X*, “laptop”)[*support =* 2*%*, *confidence*=60*%*].

The rule indicates that of the *AllElectronics* customers under study, 2% are 20 to 29 years old with an income of $40,000 to $49,000 and have purchased a laptop (computer) at *AllElectronics*. There is a 60% probability that a customer in this age and income group will purchase a laptop. Note that this is an association involving more than one attribute or predicate (i.e., *age, income*, and *buys*). Adopting the terminology used in multidimensional databases, where each attribute is referred to as a dimension, the above rule can be referred to as a **multidimensional association rule**.

Typically, association rules are discarded as uninteresting if they do not satisfy both a **minimum support threshold** and a **minimum confidence threshold**. Additional analysis can be performed to uncover interesting statistical **correlations** between associated attribute–value pairs. *Frequent itemset mining* is a fundamental form of frequent pattern mining.

1.1.4.3 Classification and Regression for Predictive Analysis

Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts. The model are 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 the class label is unknown.
"How is the derived model presented?" 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.



age(*X*, "youth") *AND income*(*X*, "high") ⟶ *class*(*X*, "A")
age(*X*, "youth") *AND income*(*X*, "low") ⟶ *class*(*X*, "B")
age(*X*, "middle_aged") ⟶ *class*(*X*, "C")
age(*X*, "senior") ⟶ *class*(*X*, "C")

(a)

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

A **decision tree** is a flowchart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easily be converted to classification rules.

A **neural network**, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. There are many other methods for constructing classification models, such as na¨ıve Bayesian classification, support vector machines, and *k*-nearest-neighbor classification.

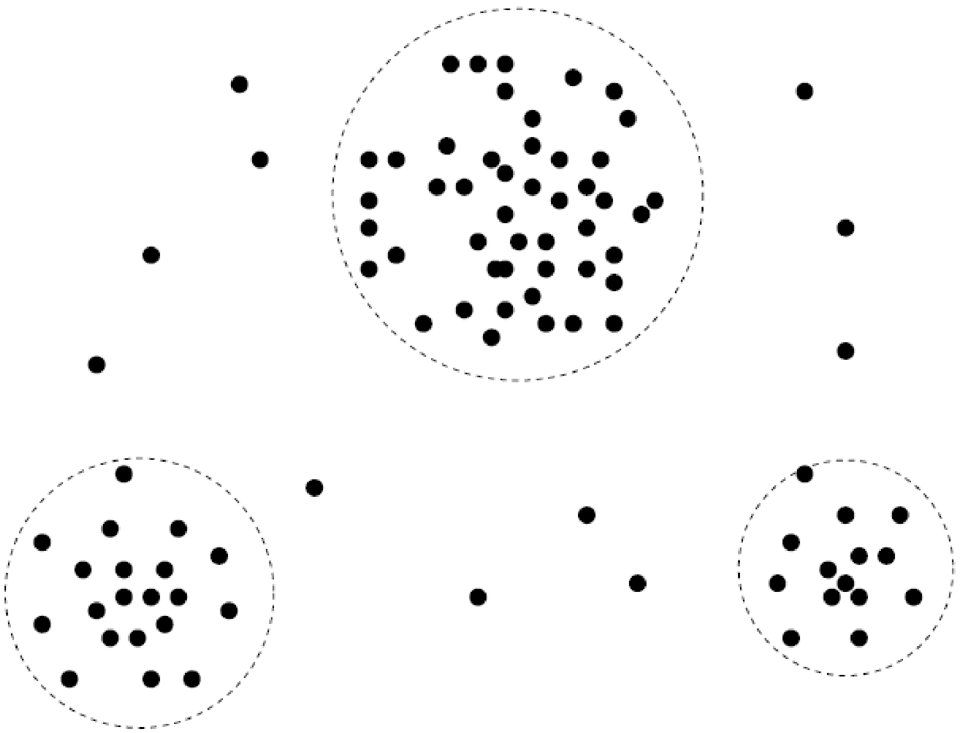
Whereas classification predicts categorical (discrete, unordered) labels, **regression** models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction 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.

Example Classification and regression. Suppose as a sales manager of *AllElectronics* you 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*. You 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*. The resulting classification should maximally distinguish each class from the others, presenting an organized picture of the data set. to predict the amount of revenue that each item will generate during an upcoming sale at *AllElectronics*, based on the previous sales data is an example of regression analysis because the regression model constructed will predict a continuous function (or ordered value.)

1.1.4.4 Cluster Analysis

Clustering analyzes data objects without consulting class labels. In many cases, class-labeled data may simply not exist at the beginning.

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* ***intraclass 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.



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

0

Example Cluster analysis. Cluster analysis can be performed on *AllElectronics* customer data to identify homogeneous subpopulations of customers.

1.1.4.5 Outlier Analysis

A data set may contain objects that do not comply with the general behavior or model of the data. These data objects are **outliers**. 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**.

Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are remote from any other cluster are considered outliers.

Example Outlier analysis. 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.

1.1.4.6 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? Can a data mining system generate all of the interesting patterns? Or, Can the system generate only the interesting ones?”*

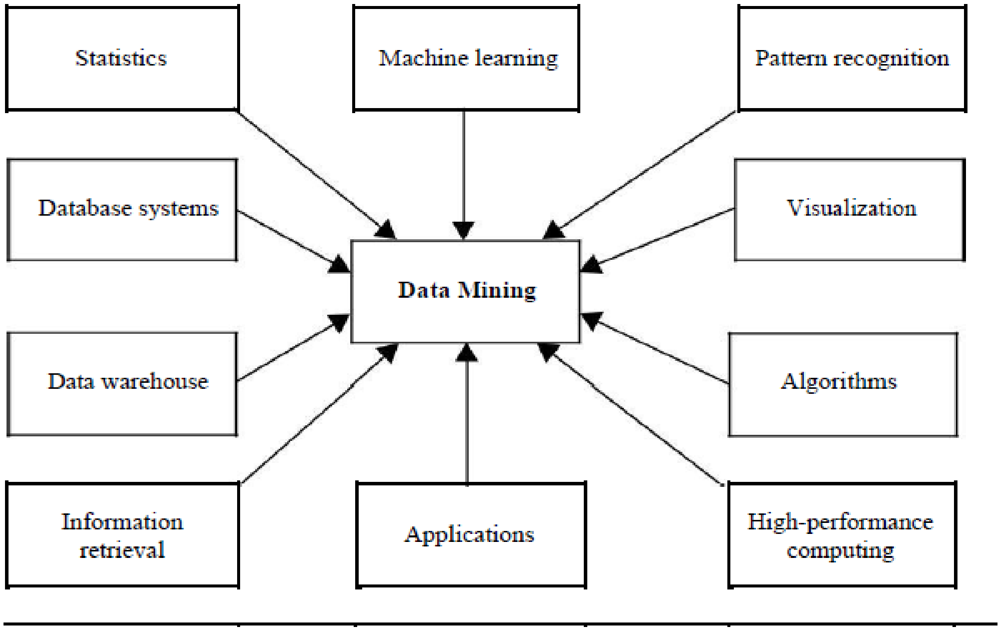
1) Is a pattern is **interesting,** if it is (1) *easily understood* by humans, (2) *valid* on new or test data with some degree of *certainty*, (3) potentially *useful*, and (4) *novel*. A pattern is also interesting if it validates a hypothesis that the user *sought to confirm*. An interesting pattern represents **knowledge**.
Several **objective measures of pattern interestingness** exist. These are based on the structure of discovered patterns and the statistics underlying them. (support, confidence, accuracy, coverage)
Subjective interestingness measures are based on user beliefs in the data. These measures find patterns interesting if the patterns are **unexpected or expected.**

2) “*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. Instead, userprovided constraints and interestingness measures should be used to focus the search.

3)*“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.

1.5 Which Technologies Are Used?

As a highly application-driven domain, data mining has incorporated many techniques from other domains such as **statistics, machine learning, pattern recognition, database and data warehouse systems, information retrieval, visualization, algorithms, high-performance computing, and many application domains**



Data mining adopts techniques from many domains.

1. Statistics

Statistics studies the collection, analysis, interpretation or explanation, and presentation of data. Data mining has an inherent connection with statistics.
A **statistical model** is a set of mathematical functions that describe the behavior of the objects in a target class in terms of random variables and their associated probability distributions. Statistical models are widely used to model data and data classes.
Statistical methods can be used to summarize or describe a collection of data. Basic **statistical descriptions** of data. Statistics is useful for mining various patterns from data as well as for understanding the underlying mechanisms generating and affecting the patterns. **Inferential statistics** (or **predictive statistics**) models data in a way that accounts for randomness and uncertainty in the observations.

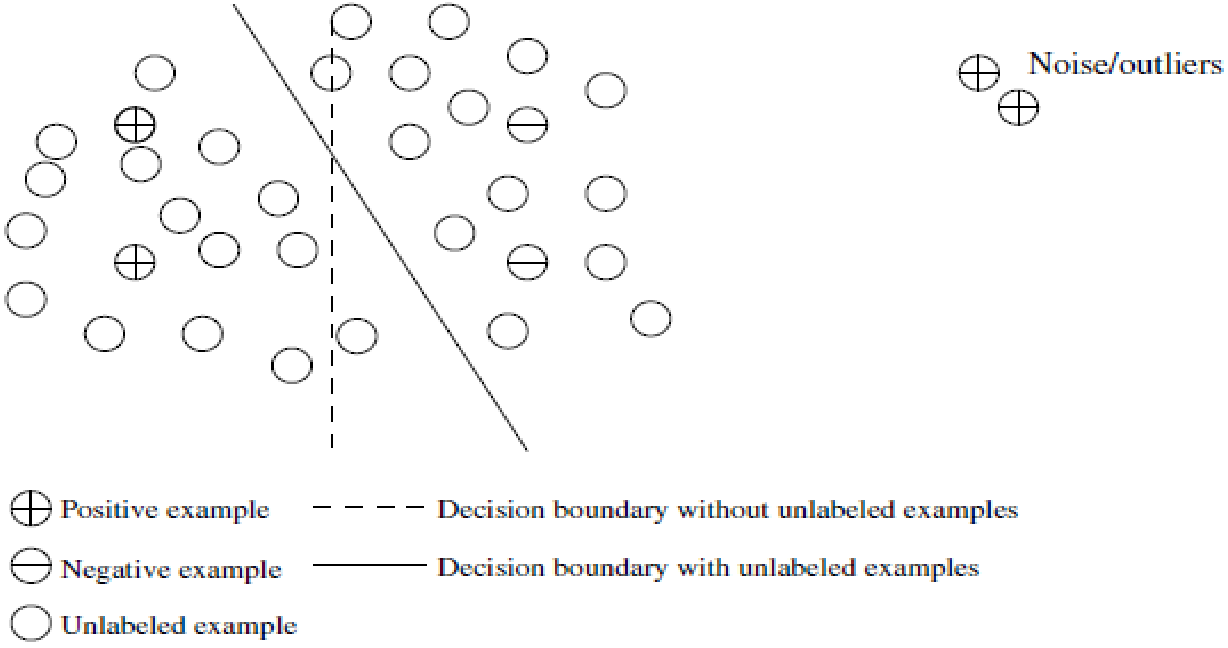
2. Machine Learning

Machine learning investigates how computers can learn (or improve their performance) based on data. For example, a typical machine learning problem is to program a computer so that it can automatically recognize handwritten postal codes on mail after learning from a set of examples.

Machine learning is a fast-growing discipline. Here, we illustrate classic problems in machine learning that are highly related to data mining.

- **Supervised learning** is basically a synonym for classification. The supervision in the learning comes from the labeled examples in the training data set. For example, in the postal code recognition problem, a set of handwritten postal code images and their corresponding machine-readable translations are used as the training examples, which supervise the learning of the classification model.

- **Unsupervised learning** is essentially a synonym for clustering. The learning process is unsupervised since the input examples are not class labeled. Typically, we may use clustering to discover classes within the data. For example, an unsupervised learning method can take, as input, a set of images of handwritten digits. Suppose that it finds 10 clusters of data. These clusters may correspond to the 10 distinct digits of 0 to 9, respectively. However, since the training data are not labeled, the learned model cannot tell us the semantic meaning of the clusters found.
- **Semi-supervised learning** is a class of machine learning techniques that make use of both labeled and unlabeled examples when learning a model. In one approach, labeled examples are used to learn class models and unlabeled examples are used to refine the boundaries between classes. For a two-class problem, we can think of the set of examples belonging to one class as the *positive examples* and those belonging to the other class as the *negative examples*. In Figure, if we do not consider the unlabeled examples, the dashed line is the decision boundary that best partitions the positive examples from the negative examples. Using the unlabeled examples, we can refine the decision boundary to the solid line. Moreover, we can detect that the two positive examples at the top right corner, though labeled, are likely noise or outliers.
- **Active learning** is a machine learning approach that lets users play an active role in the learning process. An active learning approach can ask a user (e.g., a domain expert) to label an example, which may be from a set of unlabeled examples or synthesized by the learning program. The goal is to optimize the model quality by actively acquiring knowledge from human users, given a constraint on how many examples they can be asked to label.


Noise/outliers
Positive example
Negative example
Unlabeled example
Decision boundary without unlabeled examples
Decision boundary with unlabeled examples

: Semi-supervised learning.

3. Database Systems and Data Warehouses

Database systems high scalable in processing very large, relatively structured data sets. Many data mining tasks need to handle large data sets or even real-time, fast streaming data. Therefore, data mining can make good use of scalable database technologies to achieve high efficiency and scalability on large data sets. Moreover, data mining tasks can be used to extend the capability of existing database systems to satisfy advanced users sophisticated data analysis requirements.

A **data warehouse** integrates data originating from multiple sources and various timeframes. It consolidates data in multidimensional space to form partially materialized data cubes.

4. Information Retrieval

Information retrieval (**IR**) is the science of searching for documents or information in documents. Documents can be text or multimedia, and may reside on the Web. Information retrieval assumes that (1) the data under search are unstructured; and (2) the queries are formed mainly by keywords, which do not have complex structures.

The typical approaches in information retrieval adopt probabilistic models.

For example, a text document can be regarded as a bag of words, that is, a multiset of words appearing in the document. The document's **language model** is the probability density function that generates the bag of words in the document. The similarity between two documents can be measured by the similarity between their corresponding language models.

1.1.6 Major Issues in Data Mining

The major issues in data mining research, partitioning them into five groups: *mining methodology, user interaction, efficiency and scalability, diversity of data types*, and *data mining and society*.

1.1.6.1 Mining Methodology

This involves the investigation of new kinds of knowledge, mining in multidimensional space, integrating methods from other disciplines, and the consideration of semantic ties among data objects. Some mining methods explore how user-specified measures can be used to assess the interestingness of discovered patterns as well as guide the discovery process.

These are various aspects of mining methodology.
1. Mining various and new kinds of knowledge*:* Data mining covers a wide spectrum of data analysis and knowledge discovery tasks, from data characterization and discrimination to association and correlation analysis, classification, regression, clustering, outlier analysis, sequence analysis, and trend and evolution analysis.

2. 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*..

3. Data mining - an interdisciplinary effort*:* The power of data mining can be substantially enhanced by integrating new methods from multiple disciplines.

4. 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. 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.

5. 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.

6. 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..

1.1.6.2. User Interaction

The user plays an important role in the data mining process.

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.
Interactive mining should allow users to dynamically change the focus of a search, to refine mining requests based on returned results, and to drill, dice, and pivot through the data and knowledge space interactively, dynamically exploring "cube space" while mining.

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*:* Query languages (e.g., SQL) have played an important role in flexible searching because they allow users to pose ad-hoc queries. Similarly, high-level data mining query languages or other high-level flexible user interfaces will give users the freedom to define ad-hoc data mining tasks.

Presentation and visualization of data mining results*:* presenting the data mining results, vividly and flexibly, so that the discovered knowledge can be easily understood and directly usable by humans.

1.1.6.3. Efficiency and Scalability

Efficiency and scalability are always considered when comparing data mining Algorithms.
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.

Parallel, distributed, and incremental mining algorithms*:* The large size of many data sets, the wide distribution of data, and the computational complexity of some data mining methods are factors that motivate the development of **parallel and distributed data-intensive mining algorithms**.

Cloud computing **and** ***cluster computing***, which use computers in a distributed and collaborative way to tackle very large-scale computational tasks.

1.1.6.4. 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; fromstable data repositories to dynamic data streams; fromsimple 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

- ***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. Web mining, multisource data mining, and information network mining have become challenging and fast-evolving data mining fields.

1.1.6.5. Data Mining and Society

How does data mining impact society? What steps can data mining take to preserve the privacy of individuals? Do we use data mining in our daily lives without even knowing that we do? These questions raise the following issues:

Social impacts of data mining*: 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 poses the risk of disclosing an individual's personal information. Studies on privacy-preserving data publishing and data mining are ongoing. The idea 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 systems should have data mining functions built within so that people can use data mining results simply by mouse clicking, without any knowledge of data mining algorithms. Intelligent search engines and Internet-based stores perform such *invisible data.*

UNIT –I

1.2 Types of Data

Topics covered:

- Attribute types
- Basic Statistical descriptions of data
- Measuring data similarity and dissimilarity

1.2.1 Data Objects and Attribute Types

Data sets are made up of data objects.

A **data object** represents an entity . The objects may be customers, store items, and sales in a sale database; in a medical database, the objects may be patients in a university database, the objects may be students, professors, and courses.
Data objects are typically described by attributes.

1.2.1.1 Attribute

An **attribute** is a data field, representing a characteristic or feature of a data object. The term *dimension* is commonly used in data warehousing. Data mining and database 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.

The **type** of an attribute is determined by the set of possible values—**nominal, binary, ordinal, or numeric**—the attribute can have.

1. Nominal Attributes

Nominal means "relating to names." The values of a **nominal attribute** are symbols or *names of things*. Each value represents some kind of category, code, or state, and so nominal attributes are also referred to as **categorical**. The values do not have any meaningful order. In computer science, the values are also known as ***enumerations*.**

Example Nominal attributes. The attribute *hair color* have possible values are *black*, *brown*, *red*, *auburn*, *gray*, and *white*. The attribute *marital status* can take on the values *single, married, divorced*, and *widowed*. Both *hair color* and *marital status* are nominal attributes

2. 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*.

Example Binary attributes. 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 eg: gender (male,female), and 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.

3. 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.

Example Ordinal attributes. 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) *grade* (e.g., *A+, A, A-, B+*,and so on) and *professional rank(assistant*, *associate*, and *full* for professors) Customer satisfaction had the following ordinal categories: *0: very dissatisfied, 1: somewhat dissatisfied, 2: neutral, 3: satisfied*, and *4: very satisfied.* Nominal, binary, and ordinal attributes are *qualitative*. That is, they *describe* a feature of an object without giving an actual size or quantity.

4. 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*.

a. 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 Interval-scaled attributes. 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*. 20°C is five degrees higher than a temperature of 15°C.Calendar dates are another example. For instance, the years 2002 and 2010 are eight years apart. Temperatures in Celsius and Fahrenheit do not have a true zero-point, that is, neither
0_C nor 0_F indicates “no temperature.”

b. 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 Ratio-scaled attributes. Attributes include *count* attributes such as *years of experience* (e.g., the objects are employees) and *number of words* (e.g., the objects are documents). Additional examples include attributes to measure weight, height, latitude and longitude

5. Discrete versus Continuous Attributes

The types are not mutually exclusive.

Classification algorithms developed from the field of machine learning often talk of attributes as being either *discrete* or *continuous*. Each type may be processed differently.

A **discrete attribute** has a finite or countably infinite set of values, which may or may not be represented as integers. The attributes *hair color*, *smoker*, *medical test*, and *drink size* each have a finite number of values, and so are discrete. Note that discrete attributes may have numeric values, such as 0 and 1 for binary attributes or, the values 0 to 110 for the attribute *age*. 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**. The terms *numeric attribute* and *continuous attribute* are often used interchangeably in the literature. In practice, real values are represented using a finite number of digits. Continuous attributes are typically represented as floating-point variables.

1.2.2 Basic Statistical Descriptions of Data

For data preprocessing to be successful, it is essential to have an overall picture of the data. Basic statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers.

1.2.2.1 Measuring the Central Tendency: Mean, Median, and Mode

Suppose that we have some attribute *X*, like *salary*, which has been recorded for a set of objects.
Let $x_1, x_2, \ldots, x_N$ be the set of *N* observed values or *observations* for *X*. Here, these values may also be referred to as the data set (for *X*). If we were to plot the observations for *salary*, where would most of the values fall? This gives us an idea of the central tendency of the data.

Measures of central tendency include the mean, median, mode, and midrange.
The most common and effective numeric measure of the "center" of a set of data is the *(arithmetic) mean*. Let $x_1, x_2, \ldots, x_N$ be a set of *N* values or *observations*, such as for some numeric attribute *X*, like *salary*. The **mean** of this set of values is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}. \tag{2.1}$$

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

Example . Suppose we have the following values for *salary* (in thousands of dollars), shown in increasing order: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110. Using Eq. (2.1), we have

$$\bar{x} = \frac{30+36+47+50+52+52+56+60+63+70+70+110}{12}$$
$$= \frac{696}{12} = 58.$$

Thus, the mean salary is $58,000.

Sometimes, each value *xi* in a set may be associated with a weight *wi* for *i* = 1,... ,*N*. The weights reflect the significance, importance, or occurrence frequency attached to their respective values. In this case, we can compute

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1x_1 + w_2x_2 + \cdots + w_Nx_N}{w_1 + w_2 + \cdots + w_N}. \tag{2.2}$$

This is called the **weighted arithmetic mean** or the **weighted average**

For skewed (asymmetric) data, a better measure of the center of data is the **median**, which is the middle value in a set of ordered data values. It is the value that separates the higher half of a data set from the lower half.

Example Median: The median is expensive to compute when we have a large number of observations. For numeric attributes, however, we can easily *approximate* the value. Assume that data are grouped in intervals according to their *xi* data values and that the frequency (i.e., number of data values) of each interval is known. For example, employees may be grouped according to their annual salary in intervals such as $10–20,000, $20–30,000, and so on. Let the interval that contains the median frequency be the *median interval*. We can approximate the median of the entire data set (e.g., the median salary) by interpolation using the formula

$$median = L_1 + \left(\frac{N/2 - (\sum freq)_l}{freq_{median}}\right) width,$$

where L_1 is the lower boundary of the median interval, N is the number of values in the entire data set, $(\sum freq)_l$ is the sum of the frequencies of all of the intervals

The *mode* is another measure of central tendency. The **mode** for a set of data is the value that occurs most frequently in the set. Therefore, it can be determined for qualitative and quantitative attributes. It is possible for the greatest frequency to correspond to several different values, which results in more than one mode. Data sets with one, two, or three modes are respectively called **unimodal**, **bimodal**, and **trimodal**. In general, a data set with two or more modes is**multimodal**. At the other extreme, if each data value occurs only once, then there is no mode.

Example Mode. The data from previous Example are bimodal. The two modes are $52,000 and $70,000.
For unimodal numeric data that are moderately skewed (asymmetrical), we have the following empirical relation:

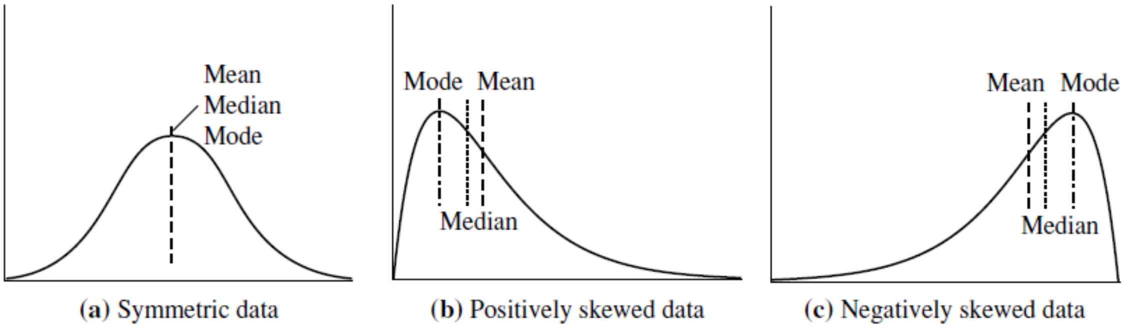
$$\textit{mean-mode} \approx 3 * (\textit{mean} - \textit{median}). \qquad (2.4)$$

This implies that the mode for unimodal frequency curves that are moderately skewed can easily be approximated if the mean and median values are known.

The **midrange** can also be used to assess the central tendency of a numeric data set. It is the average of the largest and smallest values in the set. This measure is easy to compute using the SQL aggregate functions, max() and min().

Example Midrange. The midrange of the data of is (30,000 +110,000) /2 = $70,000.

In a unimodal frequency curve with perfect **symmetric** data distribution, the mean, median, and mode are all at the same center value, as shown in Figure 2.1(a). Data in most real applications are not symmetric. They may instead be either **positively skewed**, where the mode occurs at a value that is smaller than the median (Figure 2.1b), or **negatively skewed**, where the mode occurs at a value greater than the median (Figure 2.1c).



(a) Symmetric data (b) Positively skewed data (c) Negatively skewed data

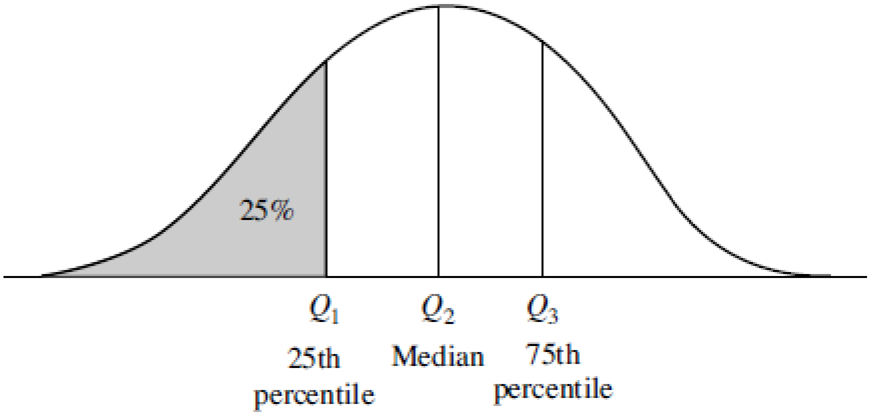
Figure 2.1 Mean, median, and mode of symmetric versus positively and negatively skewed data.

Measuring the Dispersion of Data: Range, Quartiles, Variance, Standard Deviation, and Interquartile Range

Range, Quartiles, and Interquartile Range

The **range** of the set is the difference between the largest (max()) and smallest (min()) values.

Quantiles are points taken at regular intervals of a data distribution, dividing it into essentially equalsize consecutive sets. The 100-quantiles are more commonly referred to as **percentiles**; they divide the data distribution into 100 equal-sized consecutive sets. The median, quartiles, and percentiles are the most widely used forms of quantiles.



A plot of the data distribution for some attribute X. The quantiles plotted are quartiles. The three quartiles divide the distribution into four equal-size consecutive subsets. The second quartile corresponds to the median.

The distance between the first and third quartiles is a simple measure of spread that gives the range covered by the middle half of the data. This distance is called the **interquartile range** (**IQR**) and is defined as

$$IQR = Q3 - Q1. \qquad (2.5)$$

Example 2.10 Interquartile range. The quartiles are the three values that split the sorted data set into four equal parts. The data of Example 2.6 contain 12 observations, already sorted in increasing order. Thus, the quartiles for this data are the third, sixth, and ninth values, respectively, in the sorted list. Therefore, $Q1$ = $47,000 and $Q3$ is $63,000. Thus, the interquartile range is IQR = 63 -47 = $16,000. (Note that the sixth value is a median, $52,000, although this data set has two medians since the number of data values is even.)

Five-Number Summary, Boxplots, and Outliers

No single numeric measure of spread (e.g., *IQR*) is very useful for describing skewed distributions. it is more informative to also provide the two quartiles *Q*1 and *Q*3, along with the median. A common rule of thumb for identifying suspected **outliers** is to single out values falling at least 1.5 X *IQR* above the third quartile or below the first quartile.

The **five-number summary** of a distribution consists of the median (*Q*2), the quartiles *Q*1 and *Q*3, and the smallest and largest individual observations, written in the order of *Minimum*, *Q*1, *Median*, *Q*3, *Maximum*.

Boxplots are a popular way of visualizing a distribution. A boxplot incorporates the five-number summary as follows:

- Typically, the ends of the box are at the quartiles so that the box length is the interquartile range.
- The median is marked by a line within the box.
- Two lines (called *whiskers*) outside the box extend to the smallest (*Minimum*) and largest (*Maximum*) observations.

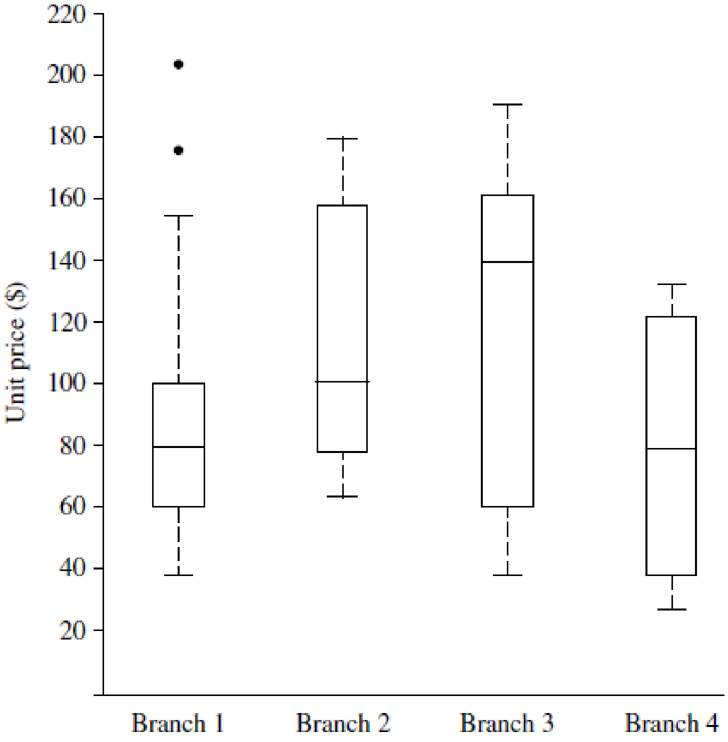


Figure 2.3 Boxplot for the unit price data for items sold at four branches of *AllElectronics* during a given time period.

When dealing with a moderate number of observations, it is worthwhile to plot potential outliers individually. To do this in a boxplot, the whiskers are extended to the extreme low and high observations *only if* these values are less than 1.5 * *IQR* beyond the quartiles.Otherwise, the whiskers terminate at the most extreme observations occurring within 1.5 * *IQR* of the quartiles. The remaining cases are plotted individually. Boxplots can be used in the comparisons of several sets of compatible data.

Example Boxplot. For branch 1, we see that the median price of items sold is \$80, *Q*1 is \$60, and *Q*3 is \$100. Notice that two outlying observations for this branch were plotted individually, as their values of 175 and 202 are more than 1.5 times the IQR here of 40.

Variance and Standard Deviation

Variance and standard deviation are measures of data dispersion. They indicate how spread out a data distribution is. A low standard deviation means that the data observations tend to be very close to the mean, while a high standard deviation indicates that the data are spread out over a large range of values.

The **variance** of N observations, $x_1, x_2, \ldots, x_N$, for a numeric attribute X is

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right) - \bar{x}^2, \tag{2.6}$$

where $\bar{x}$ is the mean value of the observations, as defined in Eq. (2.1). The **standard deviation**, σ, of the observations is the square root of the variance, σ^2.

Variance and standard deviation. In Example 2.6, we found $\bar{x} = \$58{,}000$ using Eq. (2.1) for the mean. To determine the variance and standard deviation of the data from that example, we set $N = 12$ and use Eq. (2.6) to obtain

$$\sigma^2 = \frac{1}{12}(30^2 + 36^2 + 47^2 \ldots + 110^2) - 58^2$$

$$\approx 379.17$$

$$\sigma \approx \sqrt{379.17} \approx 19.47.$$

■

1.2.2.3 Graphic Displays of Basic Statistical Descriptions of Data

These include *quantile plots*, *quantile–quantile plots*, *histograms*, and *scatter plots*. Such graphs are helpful for the visual inspection of data, which is useful for data preprocessing. The first three of these show univariate distributions (i.e., data for one attribute), while scatter plots show bivariate distributions (i.e., involving two attributes).

Quantile Plot

A **quantile plot** is a simple and effective way to have a first look at a univariate data distribution. First, it displays all of the data for the given attribute Second, it plots quantile information . Let *xi* , for *i* = 1 to *N*, be the data sorted in increasing order so that *x*1 is the smallest observation and *xN* is the largest for some ordinal or numeric attribute *X*. Each observation, *xi* , is paired with a percentage, *fi* , which indicates that approximately *fi* *100% of the data are below the value, *xi* .We say “approximately” because there may not be a value with exactly a fraction, *fi* , of the data below *xi* . Note that the 0.25 percentile corresponds to quartile *Q*1, the 0.50 percentile is the median, and the 0.75 percentile is *Q*3.

Let *fi* = *(i* - 0.5) / *N* . (2.7)

These numbers increase in equal steps of 1/*N*, ranging from 1/2*N* (which is slightlyabove 0) to 1 – 1/2*N* (which is slightly below 1). On a quantile plot, *xi* is graphed against *fi* . This allows us to compare different distributions based on their quantiles. For example, given the quantile plots of sales data for two different time periods, we can compare their *Q*1, median, *Q*3, and other *fi* values at a glance.

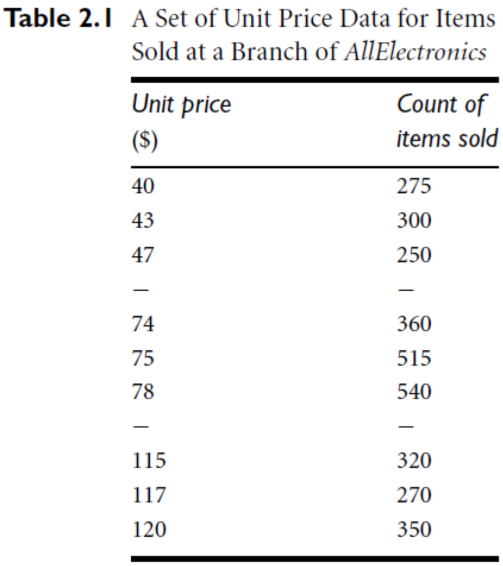
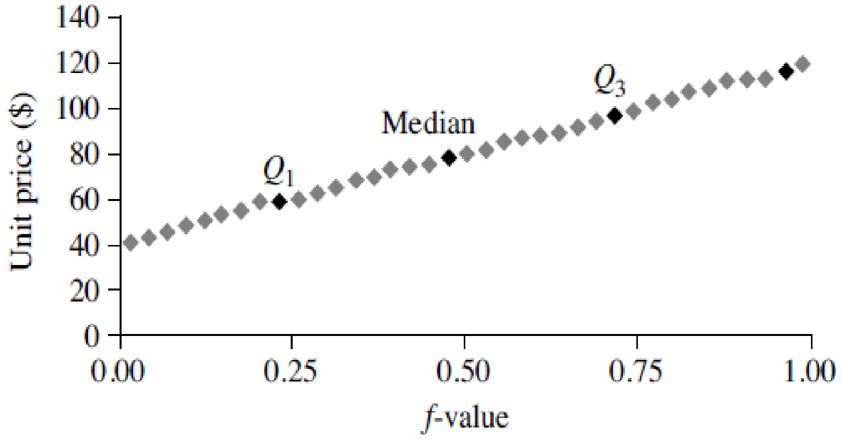


Table 2.1 A Set of Unit Price Data for Items Sold at a Branch of *AllElectronics*

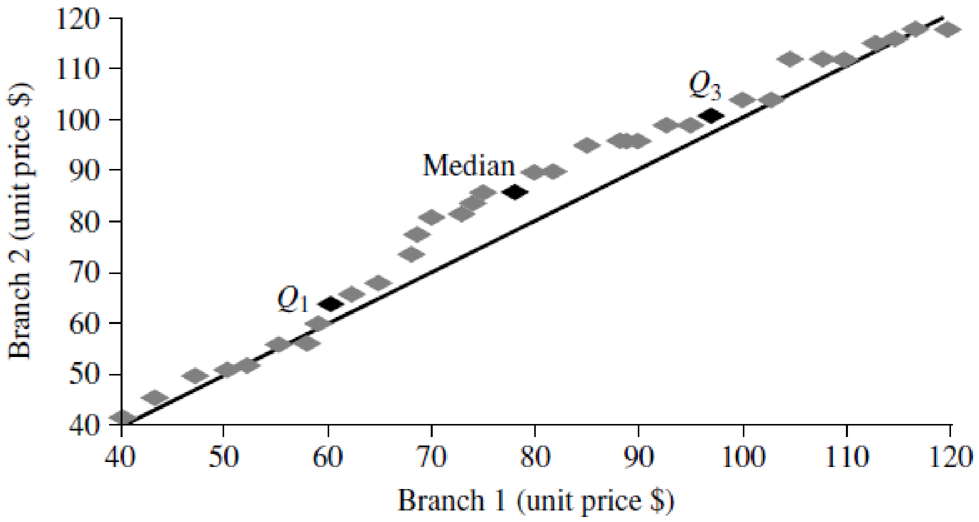
Unit price ($)	*Count of items sold*
40	275
43	300
47	250
—	—
74	360
75	515
78	540
—	—
115	320
117	270
120	350



A quantile plot for the unit price data of Table 2.1.

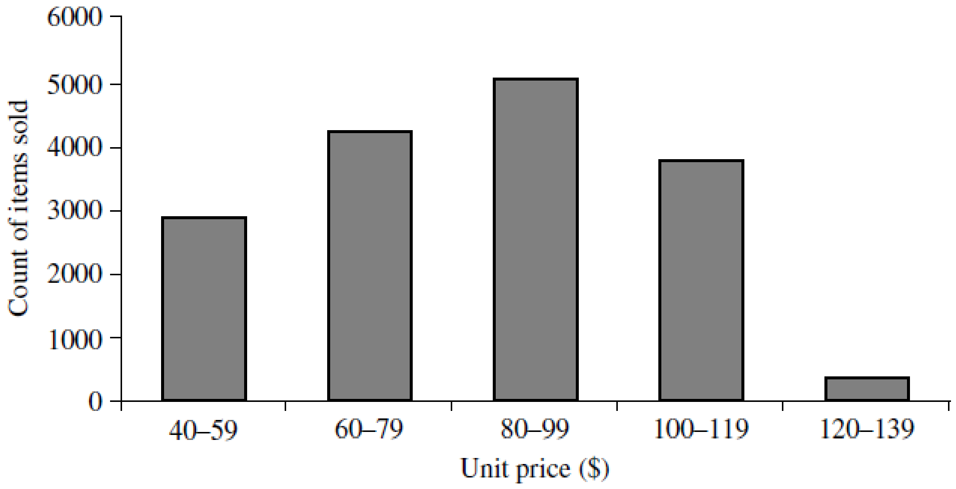
Quantile–Quantile Plot

A**quantile–quantile plot**, or **q-q plot**, graphs the quantiles of one univariate distribution against the corresponding quantiles of another. It is a powerful visualization tool in that it allows the user to view whether there is a shift in going fromone distribution to another.



A q-q plot for unit price data from two *AllElectronics* branches.

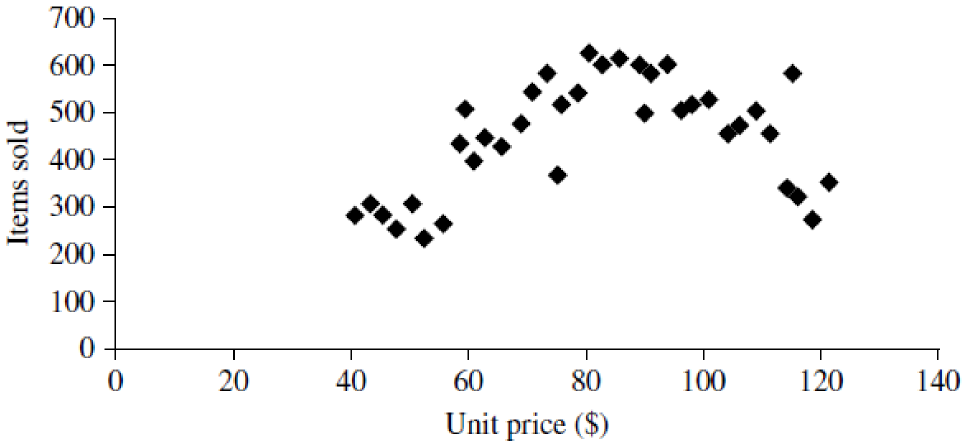
Histograms



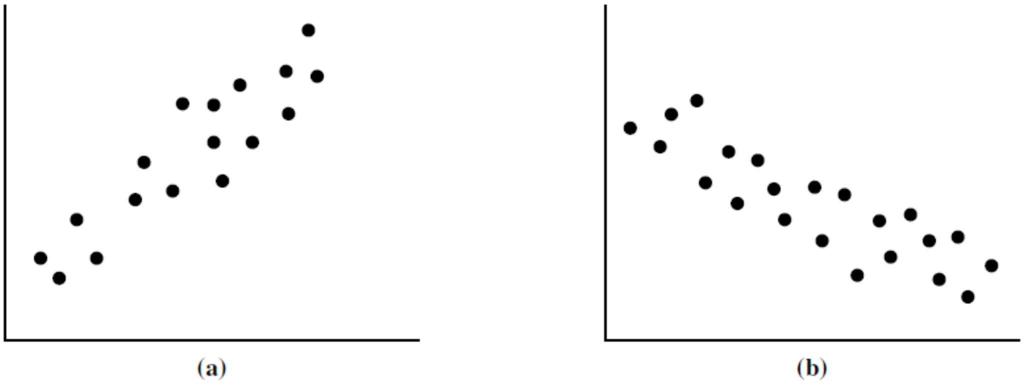
5 A histogram for the Table 2.1 data set.

Scatter Plots and Data Correlation

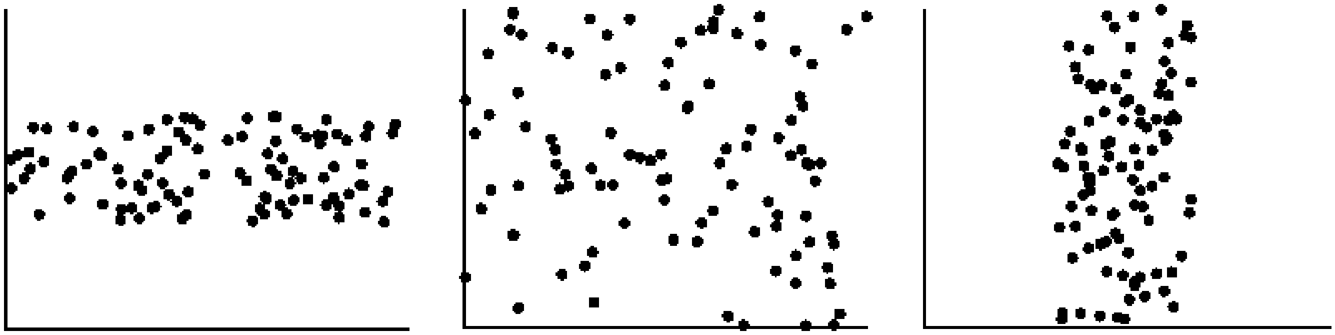
A **scatter plot** is one of the most effective graphical methods for determining if there appears to be a relationship, pattern, or trend between two numeric attributes.



A scatter plot for the Table 2.1 data set.



Scatter plots can be used to find (a) positive or (b) negative correlations between attributes.



Three cases where there is no observed correlation between the two plotted attributes in each of the data sets.

1.2.3 Measuring Data Similarity and Dissimilarity

In data mining applications, such as clustering, outlier analysis, and nearest-neighbor classification, we need ways to assess how alike or unalike objects are in comparison to one another. For example, a store may want to search for clusters of *customer* objects, resulting in groups of customers with similar characteristics (e.g., similar income, area of residence, and age). Such information can then be used for marketing. A **cluster** is a collection of data objects such that the objects within a cluster are *similar* to one another and *dissimilar* to the objects in other clusters. Outlier analysis also employs clustering-based techniques to identify potential outliers as objects that are highly dissimilar to others. Knowledge of object similarities can also be used in nearest-neighbor classification schemes where a given object (e.g., a *patient*) is assigned a class label (relating to, say, a *diagnosis*) based on its similarity toward other objects in the model.

Similarity and dissimilarity are related, which are referred to as measures of *proximity*. A similarity measure for two objects, i and j, will typically return the value 0 if the objects are unalike. 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.

Data Matrix versus Dissimilarity Matrix

Suppose that we have n objects (e.g., persons, items, or courses) described by p attributes (also called *measurements* or *features*, such as age, height, weight, or gender). The objects are $x1 = (x11, x12, \ldots, x1p)$, $x2 =(x21, x22, \ldots, x2p)$, and so on, where xij is the value for object xi of the jth attribute. For brevity, we hereafter refer to object xi as object i. The objects may be tuples in a relational database, and are also referred to as *data samples* or *feature vectors*.

Main memory-based clustering and nearest-neighbor algorithms typically operate on either of the following two data structures:

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 p attributes):

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

(2.8)

Each row corresponds to an object. As part of our notation, we may use *f* to index through the *p* attributes.

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:

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

(2.9)

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

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.

Measures of similarity can often be expressed as a function of measures of dissimilarity. For example, for nominal data

$$sim(i, j) = 1 - d(i, j),$$

(2.10)

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. Many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix. Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms.

Proximity Measures for Nominal Attributes

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*. Notice that such integers are used just for data handling and do not represent any specific ordering.

The dissimilarity between two objects *i* and *j* can be computed based on the ratio of mismatches:

$$d(i, j) = \frac{p - m}{p}, \qquad (2.11)$$

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.

Example :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

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

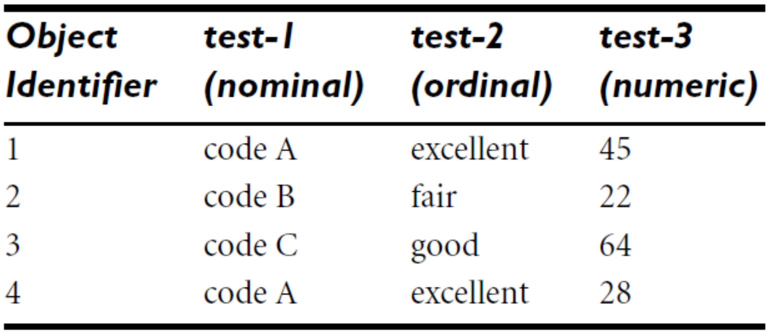
dissimilarity matrix (Eq. 2.9), that is

Since here we have one nominal attribute, *test-1*, we set *p* D 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)

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

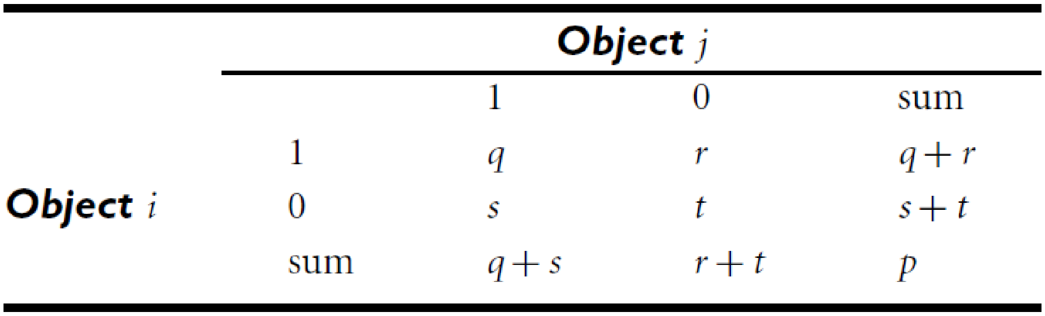
Alternatively, similarity can be computed as

$$sim(i, j) = 1 - d(i, j) = \frac{m}{p}. \tag{2.12}$$

Proximity Measures for 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.
Example: Given the attribute *smoker* describing a patient, for instance, 1 indicates that the patient smokes, while 0 indicates that the patient does not.

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

dissimilarity between *i* and *j* is

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

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**

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

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

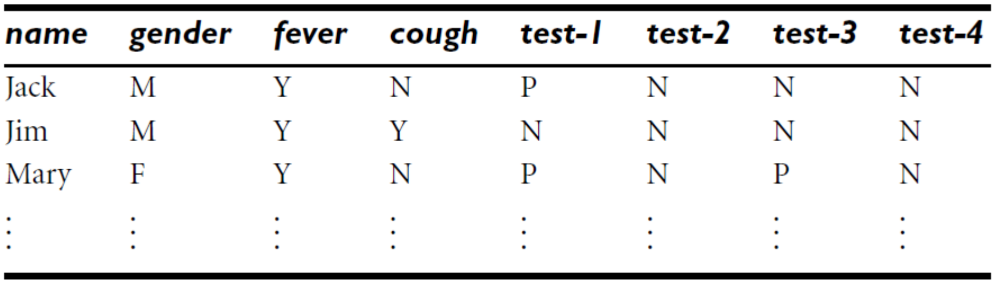
$$sim(i, j) = \frac{q}{q+r+s} = 1 - d(i, j).$$

The coefficient Sim(i, j) is called the **Jaccard coefficient**

Example Dissimilarity between binary attributes. Suppose that a patient record table 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 (patients) is computed based only on the asymmetric attributes

Table 2.4 Relational Table Where Patients Are Described by Binary Attributes



name	*gender*	*fever*	*cough*	*test-1*	*test-2*	*test-3*	*test-4*
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

The distance between each pair of the three patients—Jack, Mary, and Jim—is

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

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

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

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.

4. **Dissimilarity of Numeric Data: Minkowski Distance**

These measures include the ***Euclidean, Manhattan*, and *Minkowski distances***.
In some cases, the data are normalized before applying distance calculations. This involves transforming the data to fall within a smaller or common range, such as [1, 1] or [0.0, 1.0]. Example, *height* attribute, which could be measured in either meters or inches.

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

Another measure is known as Manhattan(or city block) and it is defined as

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

Both the Euclidean and the Manhattan distance satisfy the following mathematical properties:
Non-negativity: *d(i, j)*≥ 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) ≤ 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*.
Minkowski distance is a generalization of the Euclidean and Manhattan distances. It is defined as

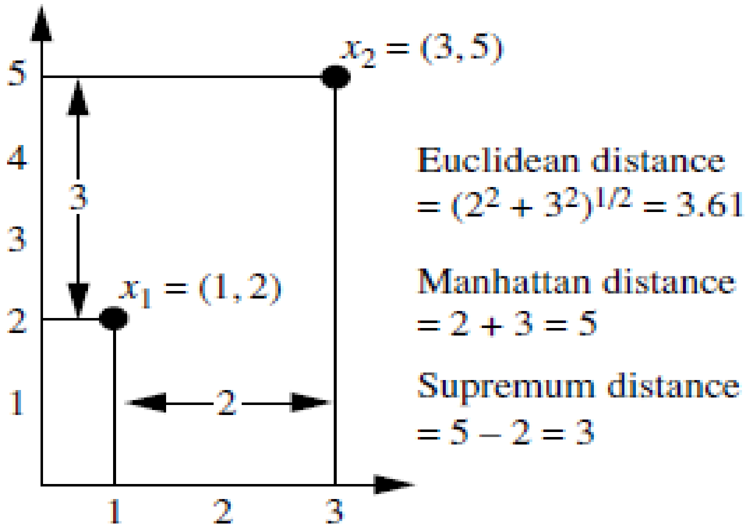
$$d(i,j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}, \qquad (2.18)$$

Supremum distance (also known as the **Chebyshev distance**) is a generalization of the Minkowski distance for *h*→ *infinity*. it is defined as

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

The L^∞ norm is also known as the *uniform norm*.

Euclidean, Manhattan, and supremum distances between two objects



If each attribute is assigned a weight according to its perceived importance, the **weighted Euclidean distance** can be computed as

$$d(i,j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \cdots + w_m|x_{ip} - x_{jp}|^2}. \qquad (2.20)$$

Weighting can also be applied to other distance measures as well.

5. 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. Let *M* represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking 1, …. , *Mf* . The treatment of ordinal attributes is quite similar to that of numeric attributes when computing dissimilarity between objects.

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

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, *Mf* D 3. For step 1, if we replace each value for *test-2* by its rank, the our 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:

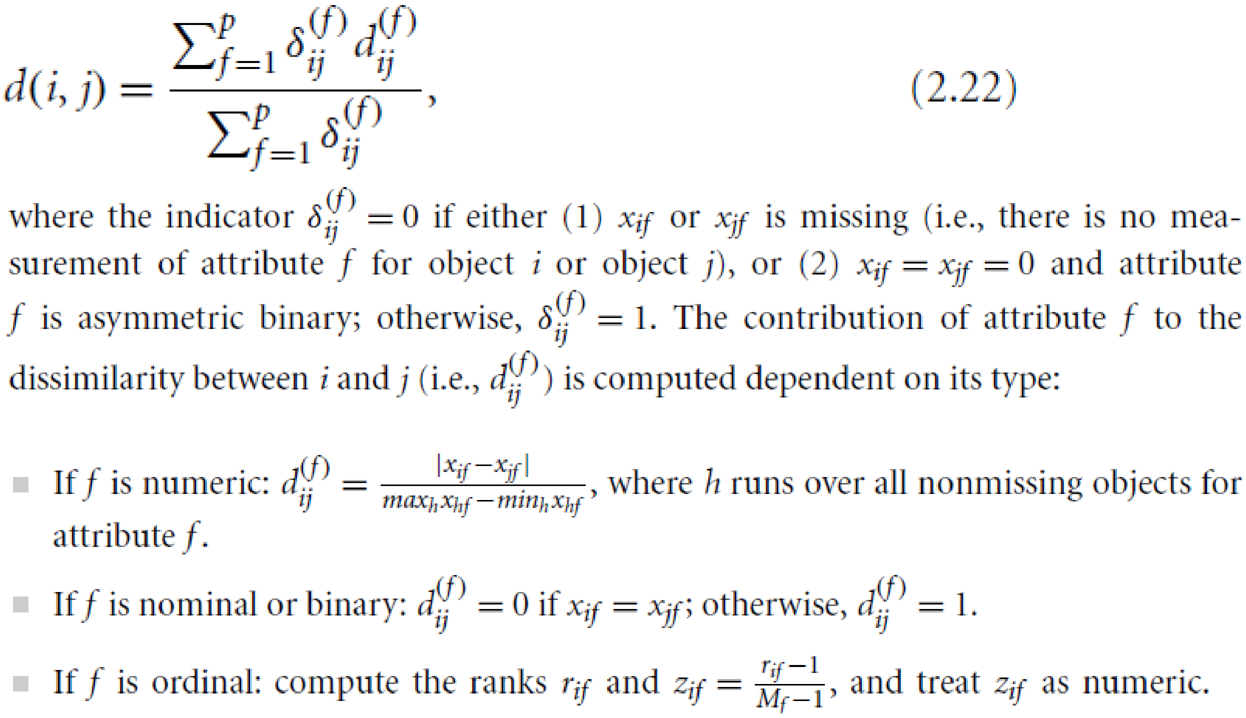
$$\begin{bmatrix} 0 & & & \\ 1.0 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

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 *sim(i, j) =* 1 *–d(i, j)*.

6. Dissimilarity for Attributes of Mixed 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 dis-similarity 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



$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}, \qquad (2.22)$$

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: 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 $max_h x_h = 64$ and $min_h x = 22$. 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:

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

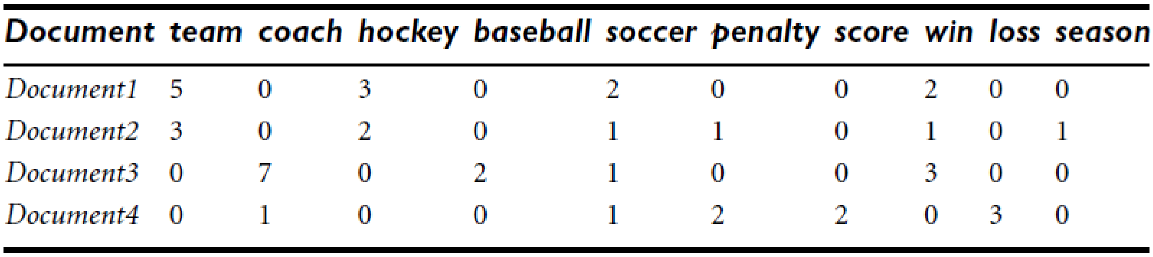
From Table 2.2, we can intuitively guess that objects 1 and 4 are the most similar, based on their values for *test*-1 and *test*-2. This is confirmed by the dissimilarity matrix, where *d(*4, 1) is the lowest value for any pair of different objects. Similarly, the matrix indicates that objects 1 and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

7. 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, a *Document* contains five instances of the word *team*, while *hockey* occurs three times. The word *coach* is absent from the entire document, Such data can be highly asymmetric.

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.

Table 2.5 Document Vector or Term-Frequency Vector



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

similarity function,

$$sim(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}||\,||\boldsymbol{y}||}, \qquad (2.23)$$

where ||***x*** || is the Euclidean norm of vector ***x* =(***x*1, *x*2, … , *xp)*, defined as $\sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$.

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*.

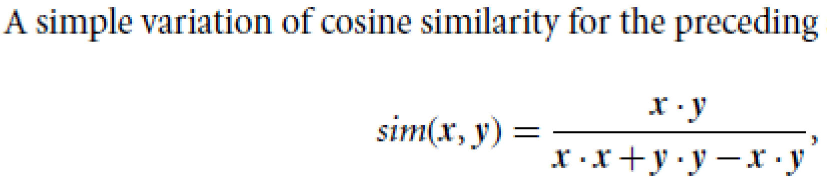
Cosine similarity between two term-frequency vectors. Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are the first two term-frequency vectors in Table 2.5. That is, $\boldsymbol{x} = (5,0,3,0,2,0,0,2,0,0)$ and $\boldsymbol{y} = (3,0,2,0,1,1,0,1,0,1)$. How similar are $\boldsymbol{x}$ and $\boldsymbol{y}$?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

$$\begin{aligned} \boldsymbol{x}^t \cdot \boldsymbol{y} &= 5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 1 + 0 \times 1 + 0 \times 0 + 2 \times 1 \\ &\quad + 0 \times 0 + 0 \times 1 = 25 \\ ||\boldsymbol{x}|| &= \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48 \\ ||\boldsymbol{y}|| &= \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2} = 4.12 \end{aligned}$$

$$sim(\boldsymbol{x}, \boldsymbol{y}) = 0.94$$

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 *xi* = 1. Then ***xt 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, Sim(.***x***, ***y)*** is a measure of relative possession of common attributes.



A simple variation of cosine similarity for the preceding scenario is

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

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.

UNIT –I

1.3 DataPreprocessing

Topics covered:

- Needs Preprocessing.
- Data Cleaning.
- Data Integration.
- Data Reduction.
- DataTransformation.

1.3.1 Data Preprocessing

There are numerous data preprocessing techniques. ***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.

1.3.1.1 Data Quality: Why Preprocess the Data?

Data have quality if they satisfy the requirements of the intended use. There are many factors comprising **data quality**, including

a. accuracy,

b. completeness,

c. consistency,

d. timeliness,

***e. believability*, and**

f. interpretability.

There are many possible reasons for inaccurate data (i.e., having incorrect attribute values).

1. The data collection instruments used may be faulty.
2. There may have been human or computer errors occurring at data entry.
3. Users may purposely submit incorrect data values for mandatory fields when they do not wish to submit personal information (e.g., by choosing the default value “January 1” displayed for birthday). This is known as ***disguised missing data***.
4. Errors in data transmission can also occur. There may be technology limitations such as limited buffer size for coordinating synchronized data transfer and consumption.

Incorrect data may also result from inconsistencies in naming conventions or data codes, or inconsistent formats for input fields (e.g., *date*). Duplicate tuples also require data cleaning.

Incomplete data can occur for a number of reasons.

Other data may not be included simply because they were not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding or because of equipment malfunctions.

Data that were inconsistent with other recorded data may have been deleted.
The recording of the data history or modifications may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.

Timeliness also affects data quality. Suppose that you are overseeing the distribution of monthly sales bonuses to the top sales representatives at *AllElectronics*. Several sales representatives, however, fail to submit their sales records on time at the end of the month. There are also a number of corrections and adjustments that flow in after the month’s end. For a period of time following each month, the data stored in the database are incomplete. However, once all of the data are received, it is correct. The fact that the month-end data are not updated in a timely fashion has a negative impact on the data quality.

Believability reflects how much the data are trusted by users.
Interpretability reflects how easy the data are understood.

Suppose that a database, at one point, had several errors, all of which have since been corrected. The past errors, however, had caused many problems for sales department users, and so they no longer trust the data. The data also use many accounting codes, which the sales department does not know how to interpret. Even though the database is now accurate, complete, consistent, and timely, sales department users may regard it as of low quality due to poor believability and interpretability.

1.3.1.2 Major Tasks in Data Preprocessing

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

suppose that we would like to include data from multiple sources in our analysis. This would involve integrating multiple databases, data cubes, or files (i.e., **data integration**). Some attributes representing a given concept may have different names in different databases, causing inconsistencies and redundancies.

For example, the attribute for customer identification may be referred to as *customer id* in one data
store and *cust id* in another. Naming inconsistencies may also occur for attribute values. For example, the same first name could be registered as "Bill" in one database, "William" in another, and "B." in a third. Furthermore, you suspect that some attributes may be inferred from others (e.g., annual revenue). Having a large amount of redundant data may slow down or confuse the knowledge discovery 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*
Example, 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*).

Discretization and ***concept hierarchy generation*** can also be useful, where raw data values for attributes are replaced by ranges or higher conceptual levels. For example, raw values for *age* may be replaced by higher-level concepts, such as *youth*, *adult*, or *senior*.

Normalization, data discretization, and concept hierarchy generation are forms of **data transformation**.

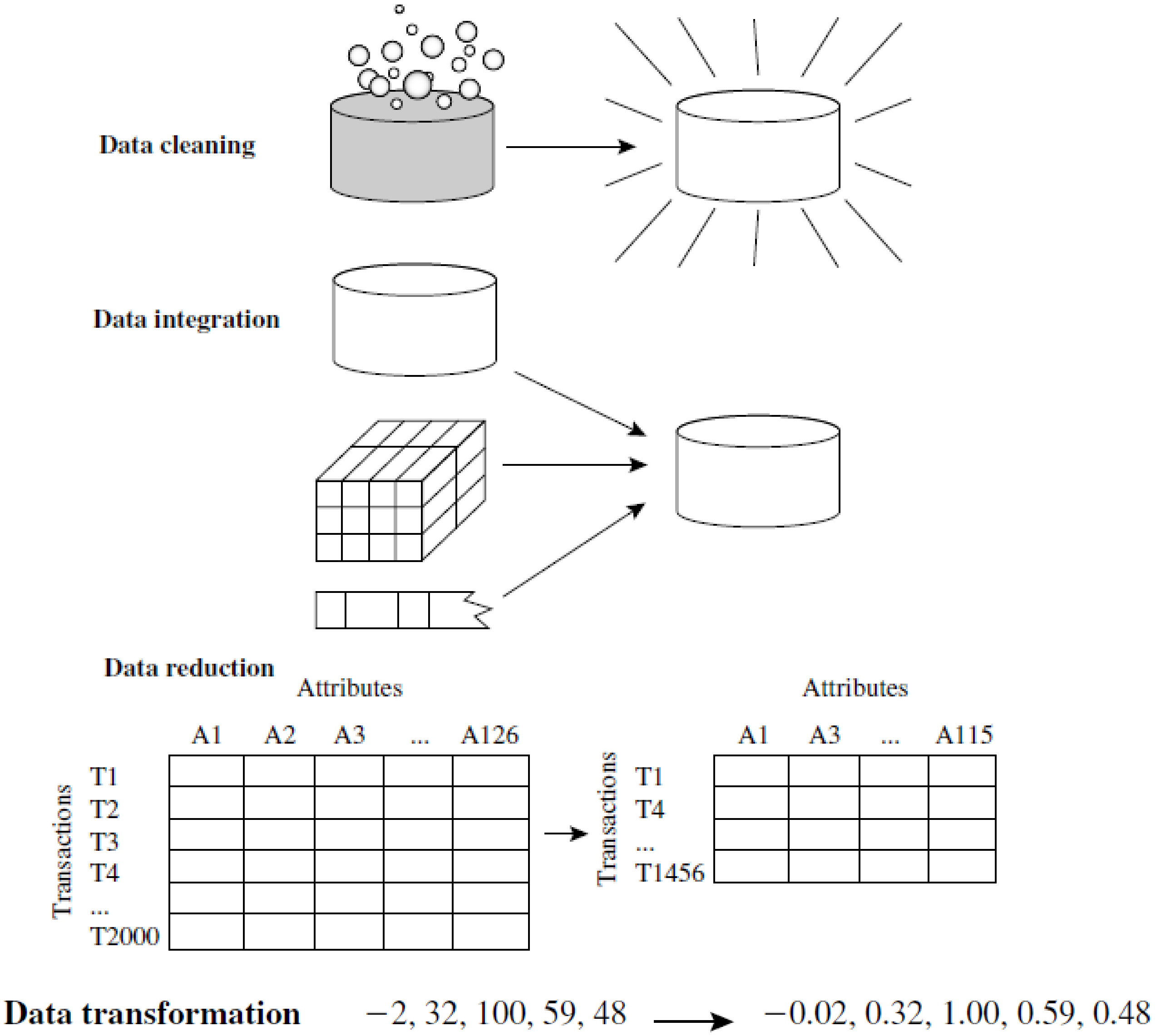


Figure 3.1 Forms of data preprocessing.

1.3.2 Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent. *Data cleaning* (or *data cleansing*) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. In this section, you will study basic methods for data cleaning.

1.3.2.1 Missing Values

following are the methods for filling the Missing values:

1. 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. By ignoring the tuple, we do not make use of the remaining attributes' values in the tuple. Such data could have been useful to the task at hand.

2. 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.

3. 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.

4. Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value: Chapter 2 discussed measures of central tendency, which indicate the "middle" value of a data distribution. For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median (Section 2.2). For example, suppose that the data distribution regarding the income of *AllElectronics* customers is symmetric and that the mean income is $56,000. Use this value to replace the missing value for *income*.

5. 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.

6. 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. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for *income*.

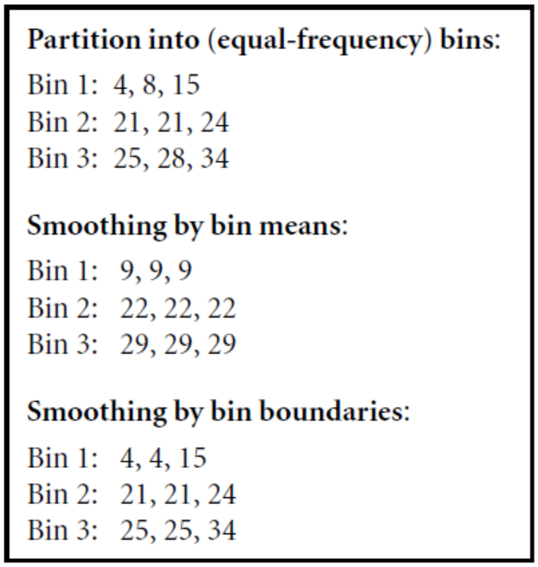
1.3.2.2 Noisy Data

Noise is a random error or variance in a measured variable. Given a numeric attribute such as, say, *price*, how can we "smooth" out the data to remove the noise. These are the following data smoothing techniques.

Binning: Binning methods smooth a sorted data value by consulting its "neighbor-hood," 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. Figure 3.2 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.
Similarly, **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

***Example: S*orted 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

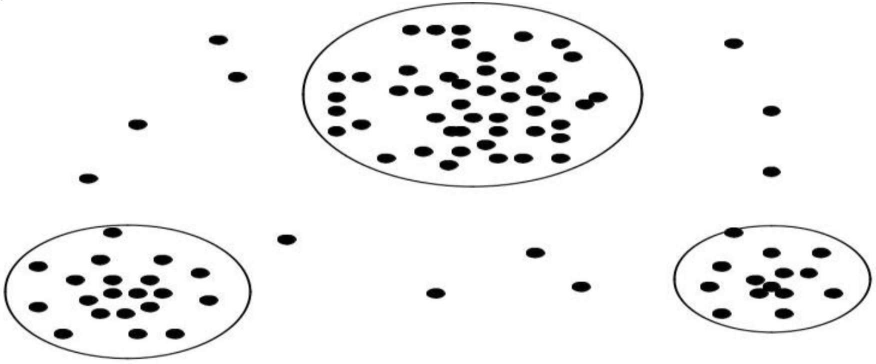
Smoothing by bin boundaries:

Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

Binning methods for data smoothing.

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 regression* 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.



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.

3.2.3 Data Cleaning as a Process

Missing values, noise, and inconsistencies contribute to inaccurate data. *But data cleaning is a big job. What about data cleaning as a process? How exactly does one proceed in tackling this task? Are there any tools out there to help?"*
The first step in data cleaning as a process is *discrepancy detection*. Discrepancies can be caused by several factors, including poorly designed data entry forms that have many optional fields, human error in data entry, deliberate errors (e.g., respondents not wanting to divulge information about themselves), and data decay (e.g., outdated addresses).
"So, how can we proceed with discrepancy detection?" using mete data(domain,data type, values of each attribute or mean median mode etc). **Field overloading** is another error source that typically results when developers squeeze new attribute definitions into unused (bit) portions of already defined attributes. The data should also be examined regarding unique rules, consecutive rules, and null rules. There are a number of different commercial tools that can aid in the discrepancy detection step. **Data scrubbing tools, Data auditing tools, Data migration tools, ETL (extraction/transformation/loading) tools.** The two-step process of discrepancy detection and data transformation (to correct discrepancies) iterates. This process, however, is error-prone and time consuming. New approaches to data cleaning emphasize increased interactivity E.g. Potter's Wheel. Another approach to increased interactivity in data cleaning is the development of declarative languages for the specification of data transformation operators.

1.3.3 Data Integration

The process of merging data from multiple data sources.
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.

1.3.3.1 Entity Identification Problem

It is likely that the data analysis task will involve *data integration*, which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files.
There are a number of issues to consider during data integration. Matching real-world entities from multiple data sources is referred to as the **entity identification problem**.
When matching attributes from one database to another during integration, special attention must be paid to the *structure* of the data. This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system.

For example, in one system, a *discount* may be applied to the order, whereas in another system it is applied to each individual line item within the order. If this is not caught before integration, items in the target system may be improperly discounted.

1.3.3.2 Redundancy and Correlation Analysis

An attribute (such as annual revenue, for instance) may be redundant if it can be “derived” from another attribute or set of attributes. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.
Some redundancies can be detected by **correlation analysis**. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For nominal data, we use the (chi-square) test.
For numeric attributes, we can use the correlation coefficient and covariance.

χ2 Correlation Test for Nominal Data

For nominal data, a correlation relationship between two attributes, *A* and *B*, can be discovered by a χ2 (**chi-square**) test. Suppose *A* has *c* distinct values, namely *a*1,*a*2, .. , *ac* . *B* has *r* distinct values, namely *b*1,*b*2, ..,*br* . The data tuples described by *A* and *B* can be shown as a **contingency table**, with the *c* values of *A* making up the columns and the *r* values of *B* making up the rows. Let (*Ai* ,*Bj)* denote the joint event that attribute *A* takes on value *ai* and attribute *B* takes on value *bj* , that is, where *A=ai* ,*B=bj*. Each and
every possible (*Ai* ,*Bj)* joint event has its own cell (or slot) in the table. The χ2 value (also known as the *Pearson χ*2 *statistic*) is computed as

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \qquad (3.1)$$

where *oij* is the *observed frequency* (i.e., actual count) of the joint event (*Ai* ,*Bj)* and *eij* is the *expected frequency* of (*Ai* ,*Bj)*, which can be computed as

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{n}, \qquad (3.2)$$

where *n* is the number of data tuples. The test is based on a significance level, with (*r*-1)*(*c*-1) degrees of freedom.

Example :Correlation analysis of nominal attributes using χ2.

Example 2.1's 2 × 2 Contingency Table Data

	male	*female*	*Total*
fiction	250 (90)	200 (360)	450
non_fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

Note: Are *gender* and *preferred_reading* correlated?

$$e_{11} = \frac{count(male) \times count(fiction)}{n} = \frac{300 \times 450}{1500} = 90,$$

Using Eq. (3.1) for χ^2 computation, we get

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840}$$
$$= 284.44 + 121.90 + 71.11 + 30.48 = 507.93.$$

For this 2 X 2 table, the degrees of freedom are (2-1)(2-1) = 1. For 1 degree of freedom, the χ2 value needed to reject the hypothesis at the 0.001 significance level is 10.828 (from standard table). Since our computed value is above this, we can reject the hypothesis that *gender* and *preferred reading* are independent and conclude that the two attributes are (strongly) correlated for the given group of people.

Correlation Coefficient for Numeric Data

For numeric attributes, we can evaluate the correlation between two attributes, *A* and *B*, by computing the **correlation coefficient** (also known as **Pearson's product moment coefficient**, named after its inventer, Karl Pearson). This is

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B}, \qquad (3.3)$$

Note that : $-1 \le r_{A,B} \le +1$. If $r_{A,B}$ is greater than 0, then *A* and *B* are *positively correlated*, meaning that the values of *A* increase as the values of *B* increase. If it is equal to 0, then *A* and *B* are *independent* and there is no correlation between them. If it is less than 0, then *A* and *B* are *negatively correlated*, where the values of one attribute increase as the values of the other attribute decrease.
Eg: # of hospitals Vs # of car theft.

Covariance of Numeric Data

In probability theory and statistics, correlation and covariance are two similar measures for assessing howmuch two attributes change together. Consider two numeric attributes *A* and *B*, and a set of *n* observations { (*a*1,*b*1), … , (*an*,*bn) }*. The mean values of *A* and *B*, respectively, are also known as the **expected values** on *A* and *B*, that is,

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n}$$

and

$$E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}.$$

The **covariance** between A and B is defined as

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}. \qquad (3.4)$$

If we compare Eq. (3.3) for $r_{A,B}$ (correlation coefficient) with Eq. (3.4) for covariance, we see that

$$r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B}, \qquad (3.5)$$

where σ_A and σ_B are the standard deviations of A and B, respectively. It can also be shown that

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}. \qquad (3.6)$$

if *A* is larger than $\bar{A}$, then *B* is likely to be larger than $\bar{B}$ the covariance between *A* and *B* is *positive*. If less then negative . If *A* and *B* are *independent* (i.e., they do not have correlation), then *E(A.B) = E(A).E(B)*. Therefore, the covariance is *Cov(A,B) = E(A.B) -* $\bar{A}\bar{B}$ = *E(A).E(B) -*$\bar{A}\bar{B}$ = 0.

1.3.3.3 Tuple Duplication

In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level (e.g., where there are two or more identical tuples for a given unique data entry case). The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy. Inconsistencies often arise between various duplicates, due to inaccurate data entry or updating some but not all data occurrences. For example, if a purchase order database contains attributes for the purchaser’s name and address instead of a key to this information in a purchaser database, discrepancies can occur, such as the same purchaser’s name appearing with different addresses within the purchase order database.

1.3.3.4 Data Value Conflict Detection and Resolution

Data integration also involves the ***detection and resolution of data value conflicts***.

For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding. For instance, a ***weight*** attribute may be stored in metric units in one system and British imperial units in another. For a hotel chain, the *price* of rooms in different cities may involve not only different currencies but also different services (e.g., free breakfast) and taxes.

When exchanging information between schools, for example, each school may have its own curriculum and grading scheme. One university may adopt a quarter system, offer three courses on database systems, and assign grades from A to F, whereas another may adopt a semester system, offer two courses on databases, and assign grades from 1 to 10. It is difficult to work out precise course-to-grade transformation rules between the two universities, making information exchange difficult.

1.3.4 Data Reduction

Consider that we have selected data from the *AllElectronics* data warehouse for analysis. The data set will likely be huge. Complex data analysis and mining on huge amounts of data can take a long time, making such analysis impractical or infeasible.
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.

1.3.4.1 Overview of Data Reduction Strategies

Data reduction strategies include

1. Dimensionality reduction,
***2. Numerosity reduction*, and**
3. Data compression.

Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. Dimensionality reduction methods include ***wavelet transforms*** and ***principal components analysis*** which transform or project the original data onto a smaller space. ***Attribute subset selection*** is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed

Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or non-parametric. For ***parametric methods***, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Regression and log-linear models) are examples. ***Nonparametric methods*** for storing reduced representations of the data include *histograms*, *clustering* , *sampling*, and *data cube aggregation.*

In **data compression**, transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be *reconstructed* from the compressed data without any information loss, the data reduction is called **lossless**. we can reconstruct only an approximation of the original data, then the data reduction is called **lossy**

1.3.4.2 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. When applying this technique to data reduction, we consider each tuple as an *n*-dimensional data vector, that is, ***X* =(***x*1, *x*2,.., *xn*), depicting *n* measurements made on the tuple from *n* database attributes.

Given a set of coefficients, an approximation of the original data can be constructed by applying the *inverse* of the DWT used.

The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines. The DWT achieves better lossy compression i.e, if the same number of coefficients is retained for a DWT and a DFT of a given data vector, the DWT version will provide a more accurate approximation of the original data.

The general procedure for applying a discrete wavelet transform uses a **hierarchical *pyramid algorithm*** that halves the data at each iteration, resulting in fast computational speed. The method is as follows:

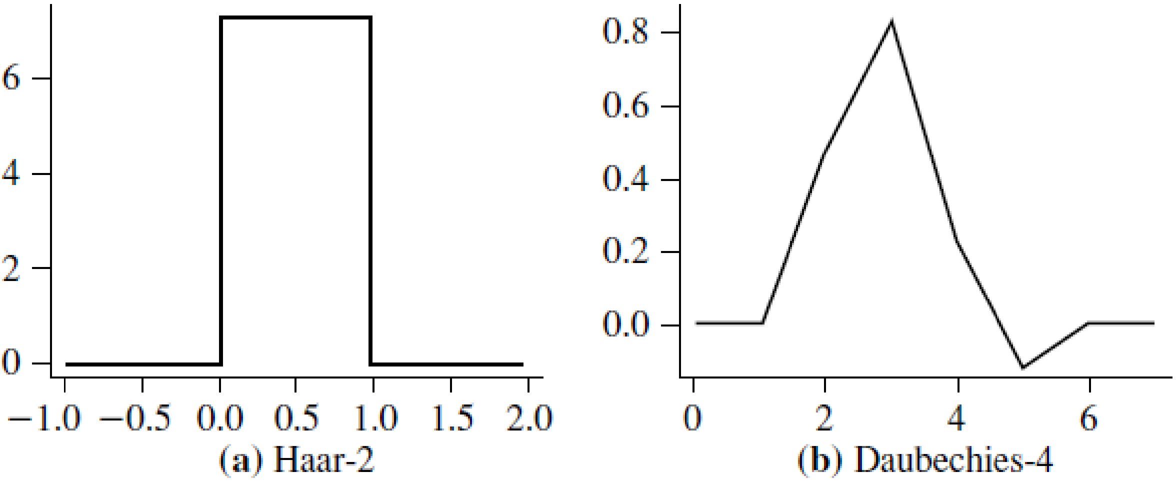
1. The length, *L*, of the input data vector must be an integer power of 2.

2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.

3. The two functions are applied to pairs of data points in ***X***, that is, to all pairs of measurements (x_{2i}, x_{2i+1}). This results in two data sets of length *L*/2. In general, these represent a smoothed or low-frequency version of the input data and the high-frequency content of it, respectively.

4. The two functions are recursively applied to the data sets obtained in the previous loop, until the resulting data sets obtained are of length 2.

5. Selected values from the data sets obtained in the previous iterations are designated the wavelet coefficients of the transformed data.



Examples of wavelet families.

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.

The computational complexity involved is linear with respect to the number of cells in the cube. Wavelet transforms give good results on sparse or skewed data and on data with ordered attributes.

Lossy compression by wavelets is reportedly better than JPEG compression, the current commercial standard. Wavelet transforms have many real-world applications, including the compression of fingerprint images, computer vision, analysis of time-series data, and data cleaning.

3.4.3 **Principal Components Analysis**

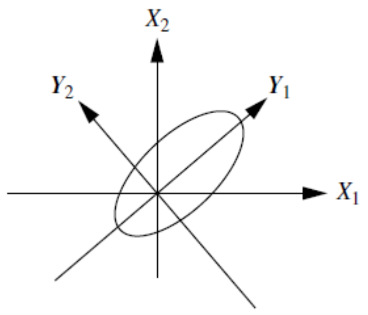
Suppose that the data to be reduced consist of tuples or data vectors described by *n* attributes or dimensions. **Principal components analysis** (**PCA**; also called the Karhunen-Loeve, or K-L, method) searches for *k* *n*-dimensional orthogonal vectors that can best be used to represent the data,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

The basic procedure is as follows:

1. The input data are normalized, so that each attribute falls within the same range.

2. PCA computes *k* orthonormal 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 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.



Principal components analysis. Y_1 and Y_2 are the first two principal components for the given 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. Principal components may be used as inputs to multiple regression and cluster analysis. In comparison with wavelet transforms, PCA tends to be better at handling sparse data, whereas wavelet transforms are more suitable for data of high dimensionality.

1.3.4.4 Attribute Subset Selection

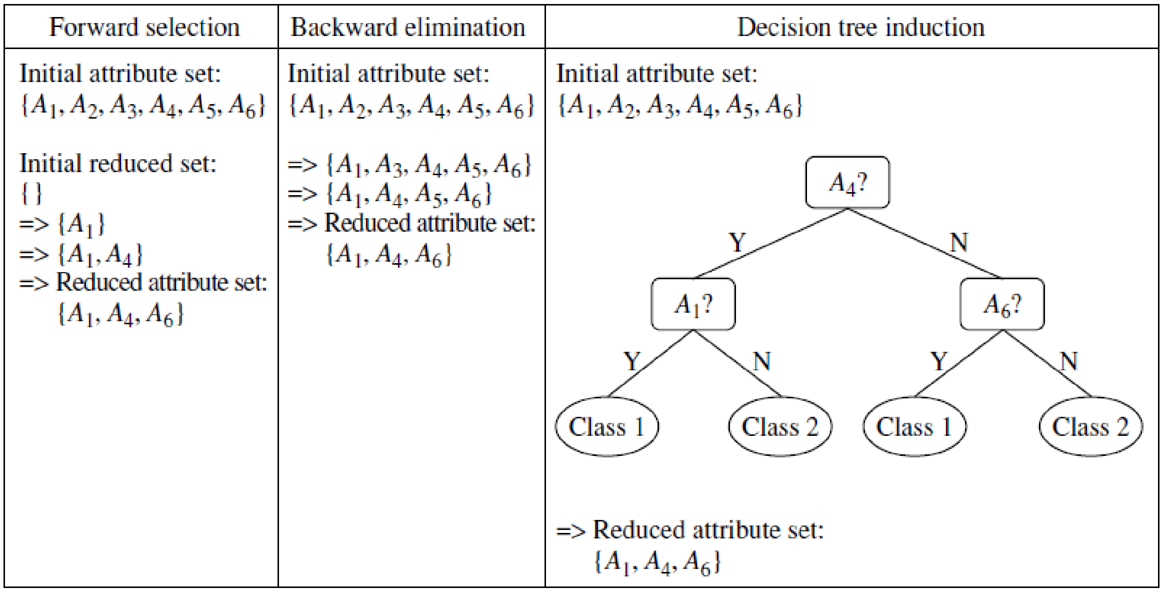
Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. For example, if the task is to classify customers based on whether or not they are likely to purchase a popular new CD at *AllElectronics* when notified of a sale, attributes such as the customer's telephone number are likely to be irrelevant, unlike attributes such as *age* or *music taste*.

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.

"How can we find a 'good' subset of the original attributes?" For *n* attributes, there are 2^n possible subsets. These methods are typically **greedy** in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.

The "best" (and "worst") attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used such as the *information gain* measure used in building decision trees for classification.5

Basic heuristic methods of attribute subset selection include the techniques that follow, some of which are illustrated below:



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: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$

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 (nonleaf) 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.

3.4.5 Regression and Log-Linear Models: Parametric Data Reduction

Regression and log-linear models can be used to approximate the given data. In **linear regression**, the data are modeled to fit a straight line. x-predictor variable, y- response variable.

$$y = wx + b, \qquad (3.7)$$

The coefficients, *w (slope)* and *b(y-intercept) are* called *regression coefficients*,

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.

Regression and log-linear models can both be used on sparse data, although their application may be limited. While both methods can handle skewed data, regression does exceptionally well. Regression can be computationally intensive when applied to high-dimensional data, whereas log-linear models show good scalability for up to 10 or so dimensions.

1.3.4.6 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.

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. More studies are needed regarding the effectiveness of multidimensional histograms for high dimensionalities.

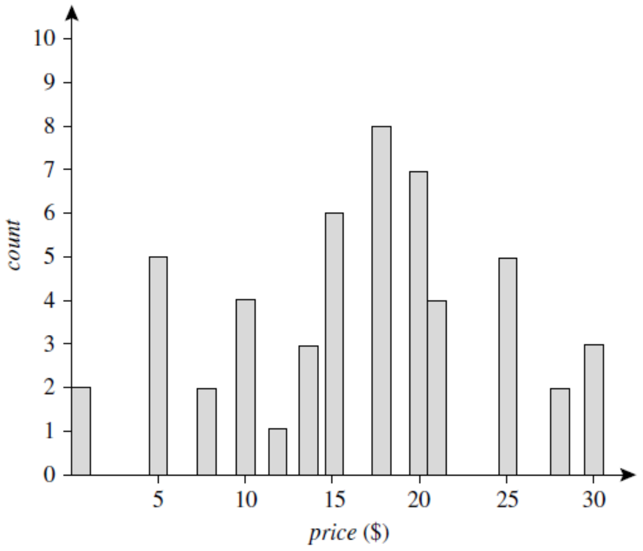
Singleton buckets are useful for storing high-frequency outliers

"How are the buckets determined and the attribute values partitioned?" There are several partitioning rules, including the following:

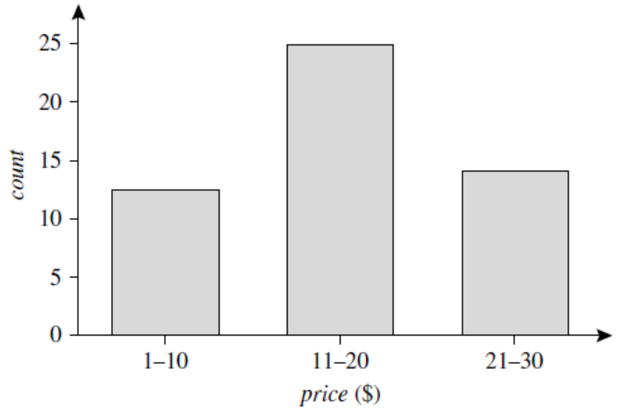
Equal-width: In an equal-width histogram, the width of each bucket range is uniform (e.g., the width of $10 for the buckets in Figure 3.8).

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).

Example Histograms. The following data are a list of *AllElectronics* 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, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.



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



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

1.3.4.7 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.

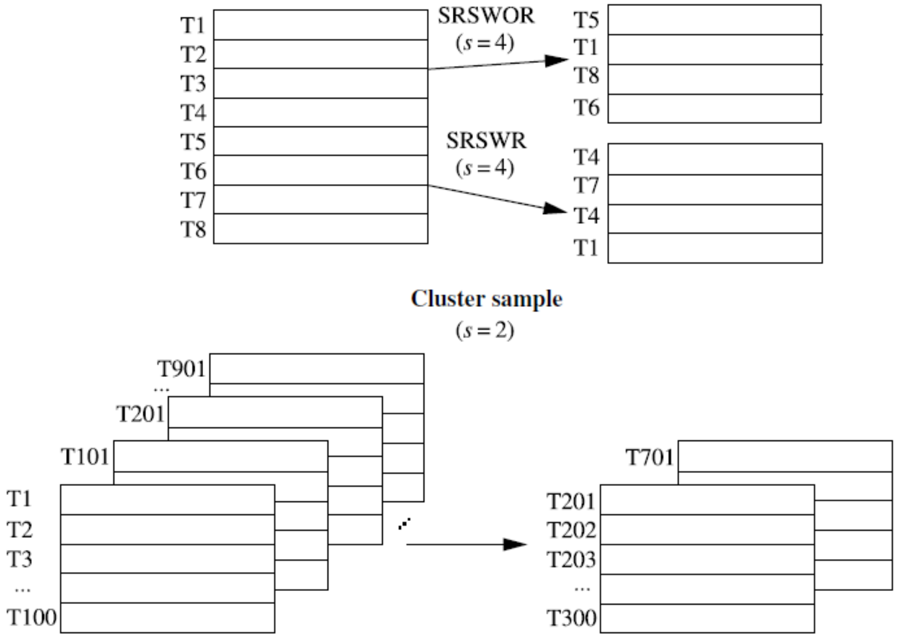
Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "average object," or average point in space for the cluster).

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. There are many measures for defining clusters and cluster quality.

1.3.4.8 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.

- **Simple random sample without replacement (SRSWOR) of size** *s*: 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) of size** *s*: 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 a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples. Other clustering criteria conveying rich semantics can also be explored. For example, in a spatial database, we may choose to define clusters geographically based on how closely different areas are located.
- **Stratified sample**: If *D* is divided into mutually disjoint parts called *strata,* a stratified sample of *D* is generated by obtaining an SRS at each stratum.



Startified sample
(according to *age*)

T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

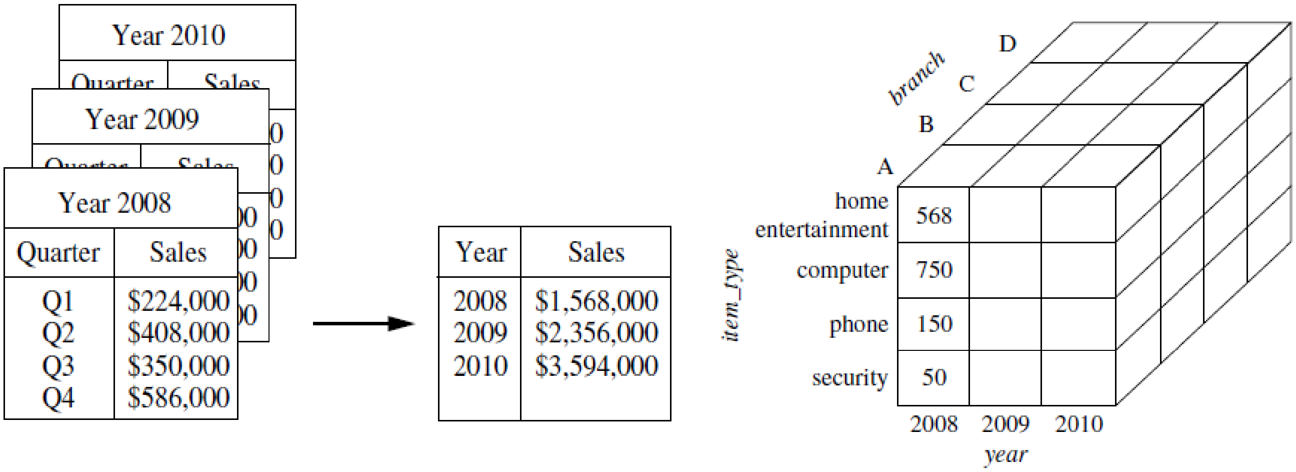
T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior

An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, s, as opposed to N , the data set size. Hence, sampling complexity is potentially sublinear to the size of the data.

Other data reduction techniques can require at least one complete pass through D. For a fixed sample size, sampling complexity increases only linearly as the number of data dimensions, n, increases, whereas techniques using histograms, for example, increase exponentially in n.

1.3.4.9 Data Cube Aggregation

Data cube stores multidimensional aggregated information. Data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch.



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

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.

Each cell holds an aggregate data value, corresponding to the data point in multidimensional space. (For readability, only some cell values are shown.) *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. 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.

cube at the highest level of abstraction is the **apex cuboid**. For the sales data, 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***.

1.3.5 Data Transformation and Data Discretization

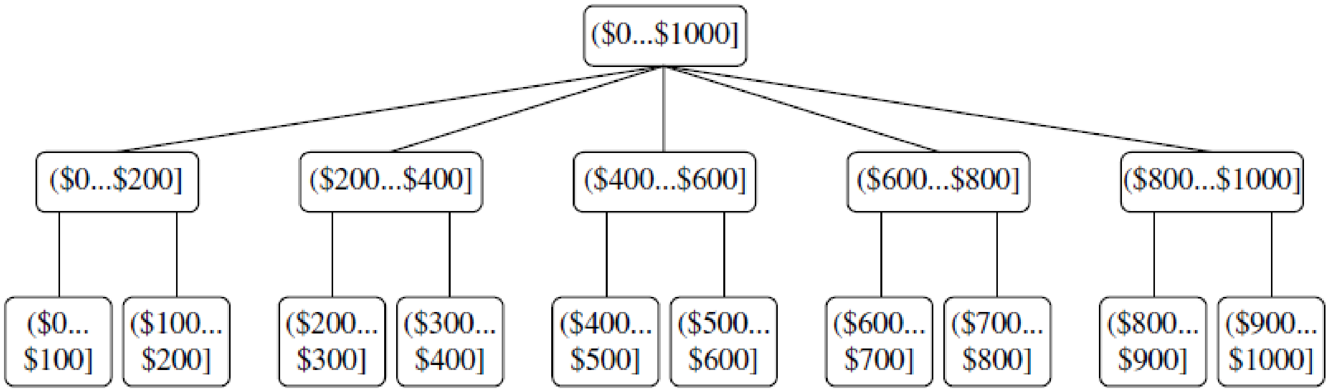
In this preprocessing step, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand.

3.5.1 Data Transformation Strategies Overview

In *data transformation*, the data are transformed or consolidated into forms appropriate for mining. Strategies for data transformation include the following:

- **Smoothing**, which works to remove noise from the data. Techniques include binning, regression, and clustering.
- **Attribute construction** (or *feature construction*), where new attributes are con-structed and added from the given set of attributes to help the mining process.
- **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. This step is typically used in constructing a data cube for data analysis at multiple abstraction levels.
- **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.
- **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*). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute
- **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.

Recall that there is much overlap between the major data preprocessing tasks. The first three of these strategies were discussed earlier in this chapter. Smoothing is a form of



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

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.

Data discretization and concept hierarchy generation are also forms of data reduction. The raw data are replaced by a smaller number of interval or concept labels. This simplifies the original data and makes the mining more efficient. The resulting patterns mined are typically easier to understand. Concept hierarchies are also useful for mining at multiple abstraction levels.

1.3.5.2 Data Transformation by Normalization

The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or from kilograms to pounds for *weight*, may lead to very different results.

In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or "weight." To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*. This involves transforming the data to fall within a smaller or common range such as [1, 1] or [0.0, 1.0]. (The terms *standardize* and *normalize* are used interchangeably in data preprocessing)

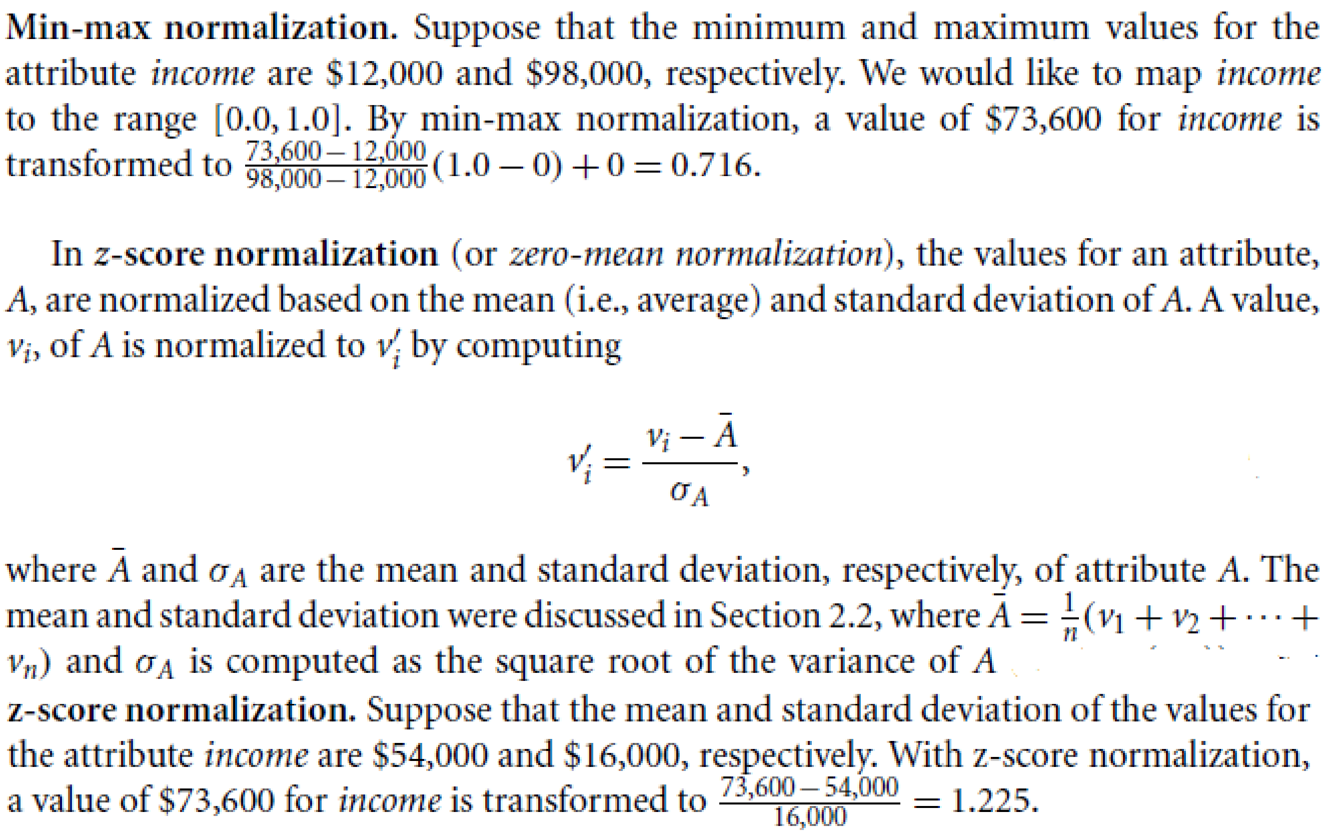
Normalizing the data attempts to give ll attributes an equal weight. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest neighbor classification and clustering. If using the neural network backpropagation algorithm for classification mining, normalizing the input values for each attribute measured in the training tuples will help speed up the learning phase.

There are many methods for data normalization. ***min-max normalization, z-score normalization,*** **and** ***normalization by decimal scaling****.*
let *A* be a numeric attribute with *n* observed values, *v*1, *v*2, …, *vn*.

Min-max normalization performs a linear transformation on the original data. Suppose that *minA* and *maxA* are the minimum and maximum values of an attribute, *A*. Min-max normalization maps a value, *vi*, of *A* to *vi***0** in the range [*new minA*, *new maxA*] by computing

$$v'_i = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

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*.



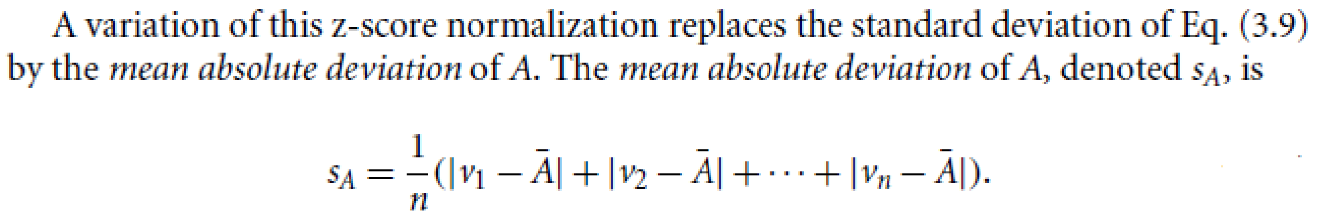
Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of \$73,600 for *income* is transformed to $\frac{73{,}600-12{,}000}{98{,}000-12{,}000}(1.0-0)+0=0.716$.

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 $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. The mean and standard deviation were discussed in Section 2.2, where $\bar{A} = \frac{1}{n}(v_1 + v_2 + \cdots + v_n)$ and σ_A is computed as the square root of the variance of A

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 $\frac{73{,}600-54{,}000}{16{,}000} = 1.225$.



A variation of this z-score normalization replaces the standard deviation of Eq. (3.9) by the *mean absolute deviation* of A. The *mean absolute deviation* of A, denoted s_A, is

$$s_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + \cdots + |v_n - \bar{A}|).$$

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, *vi* , of *A* is normalized to v'_i by computing

$$v'_i = \frac{v_i}{10^j}, \tag{3.12}$$

where *j* is the smallest integer such that max($|v'_i|$)< 1.

Example Decimal scaling. 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 1000 (i.e., *j*=3) so that -986 normalizes to -0.986 and 917 normalizes to 0.917.

1.3.5.3 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 applying 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.

1.3.5.4 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 prespecified 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.

1.3.5.5 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.

Techniques to generate decision trees for classification can be applied to discretization. Such techniques employ a top-down splitting approach. Decision tree approaches to discretization are supervised, that is, they make use of class label information.

For example, we may have a data set of patient symptoms (the attributes) where each patient has an associated *diagnosis* class label. Class distribution information is used in the calculation and determination of split-points (data values for partitioning an attribute range). Intuitively, the main idea is to select split-points so that a given resulting partition contains as many tuples of the same class as possible. *Entropy* is the most commonly used measure for this purpose. To discretize a numeric attribute, *A*, the method selects the value of *A* that has the minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Such discretization forms a concept hierarchy for *A*.

1.3.5.6 Concept Hierarchy Generation for Nominal Data

Data transformation for nominal data. Nominal 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*.
four methods for the generation of concept hierarchies for nominal data, as follows.

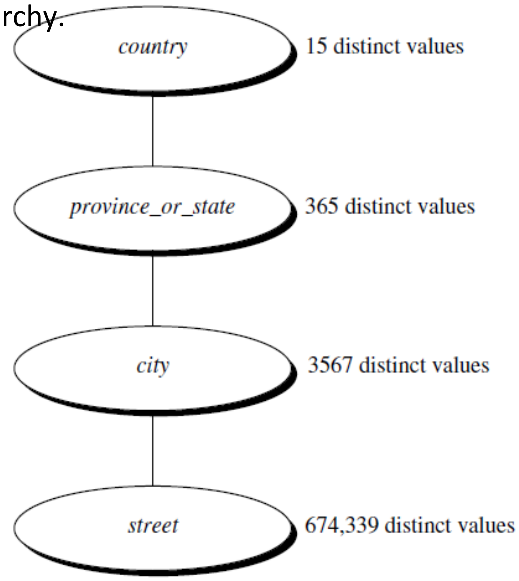
1. Specification of a partial ordering of attributes explicitly at the schema level by users or experts: Concept hierarchies for nominal attributes or dimensions typically involve a group of attributes.

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:. 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.

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.



Automatic generation of a schema concept hierarchy based on the number of distinct attribute values. First, sort the attributes in ascending order based on the number of distinct values in each attribute. Results in: *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 bottomlevel.

This heuristic rule is not foolproof. For example, a time dimension in a database may contain 20 distinct years, 12 distinct months, and 7 distinct days of the week. However, this does not suggest that the time hierarchy should be "*year* <*month* < *days of the week*," with *days of the week* at the top of the hierarchy.

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.

Example: Concept hierarchy generation using prespecified semantic connections. Suppose that a data mining expert (serving as an administrator) has pinned together the five attributes *number, street, city, province or state*, and *country*, because they are closely linked semantically regarding the notion of *location*. If a user were to specify only the attribute*city* for a hierarchy defining *location*, the systemcan automatically drag in all five semantically related attributes to form a hierarchy. The user may choose to drop any of these attributes (e.g., *number* and *street*) from the hierarchy, keeping *city* as the lowest conceptual level.