LECTURE NOTES

ON

DATA WAREHOUSE AND DATA MINING

III B. Tech II semester (JNTUH-R15)

Mr. Ch Suresh Kumar Raju

Assistant Professor

INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)

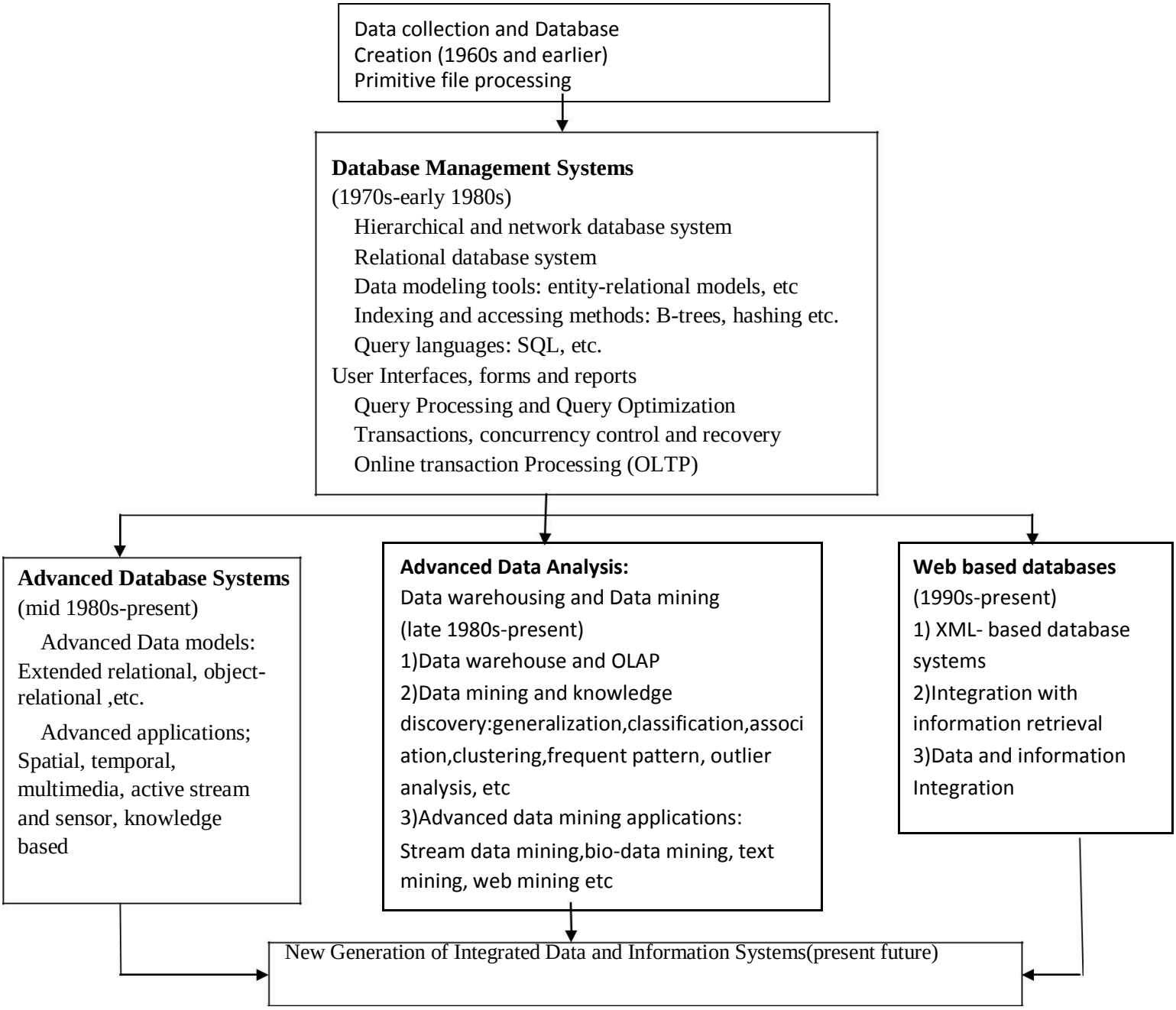
DUNDIGAL, HYDERABAD - 500 043

UNIT I

What motivated data mining? Why is it important?

The major reason that data mining has attracted a great deal of attention in information industry in recent years is due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. The information and knowledge gained can be used for applications ranging from business management, production control, and market analysis, to engineering design and science exploration.

The evolution of database technology



What is data mining?

Data mining refers to extracting or mining" knowledge from large amounts of data. There are many other terms related to data mining, such as knowledge mining, knowledge extraction, data/pattern analysis, data archaeology, and data dredging. Many people treat data mining as a synonym for another popularly used term, Knowledge Discovery in Databases", or KDD

Essential step in the process of knowledge discovery in databases

Knowledge discovery as a process is depicted in following figure and consists of an iterative sequence of the following steps:

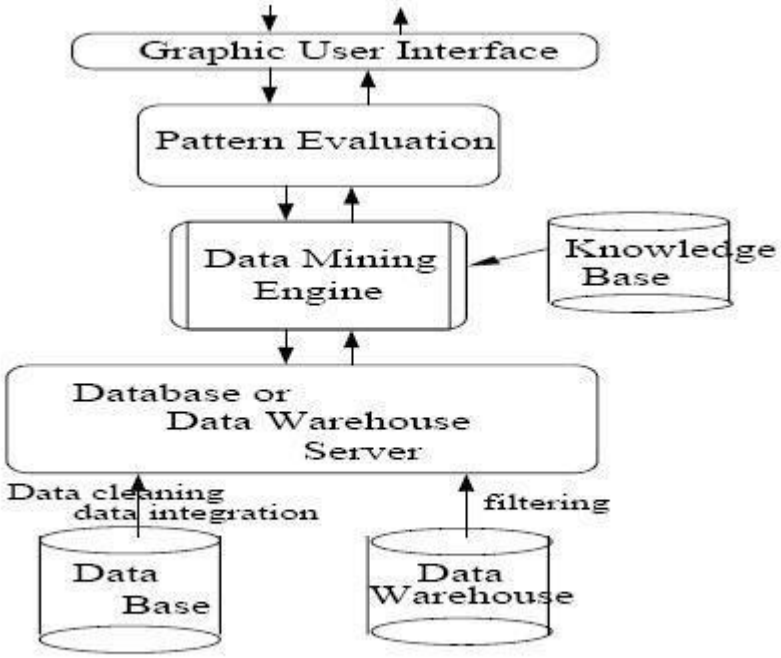
- data cleaning: to remove noise or irrelevant data
- data integration: where multiple data sources may be combined
- data selection: where data relevant to the analysis task are retrieved from the database
- data transformation: where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations
- data mining :an essential process where intelligent methods are applied in order to extract data patterns
- pattern evaluation to identify the truly interesting patterns representing knowledge based on some interestingness measures
- knowledge presentation: where visualization and knowledge representation techniques are used to present the mined knowledge to the user.

Architecture of a typical data mining system/Major Components

Data mining is the process of discovering interesting knowledge from large amounts of data stored either in databases, data warehouses, or other information repositories. Based on this view, the architecture of a typical data mining system may have the following major components:

1. A database, data warehouse, or other information repository, which consists of the set of databases, data warehouses, spreadsheets, or other kinds of information repositories containing the student and course information.
   A database or data warehouse server which fetches the relevant data based on users' data mining requests.
   A knowledge base that contains the domain knowledge used to guide the search or to evaluate the interestingness of resulting patterns. For example, the knowledge base may contain metadata which describes data from multiple heterogeneous sources.
4. A data mining engine, which consists of a set of functional modules for tasks such as classification, association, classification, cluster analysis, and evolution and deviation analysis.
5. A pattern evaluation module that works in tandem with the data mining modules by employing interestingness measures to help focus the search towards interestingness patterns.

A graphical user interface that allows the user an interactive approach to the data mining system.



Architecture of a typical data mining system.

How is a data warehouse different from a database? How are they similar?

Differences between a data warehouse and a database: A data warehouse is a repository of information collected from multiple sources, over a history of time, stored under a unified schema, and used for data analysis and decision support; whereas a database, is a collection of interrelated data that represents the current status of the stored data. There could be multiple heterogeneous databases where the schema of one database may not agree with the schema of another. A database system supports ad-hoc query and on-line transaction processing. **For more details, please refer to the section "Differences between operational database systems and data warehouses."**

Similarities between a data warehouse and a database: Both are repositories of information, storing huge amounts of persistent data.

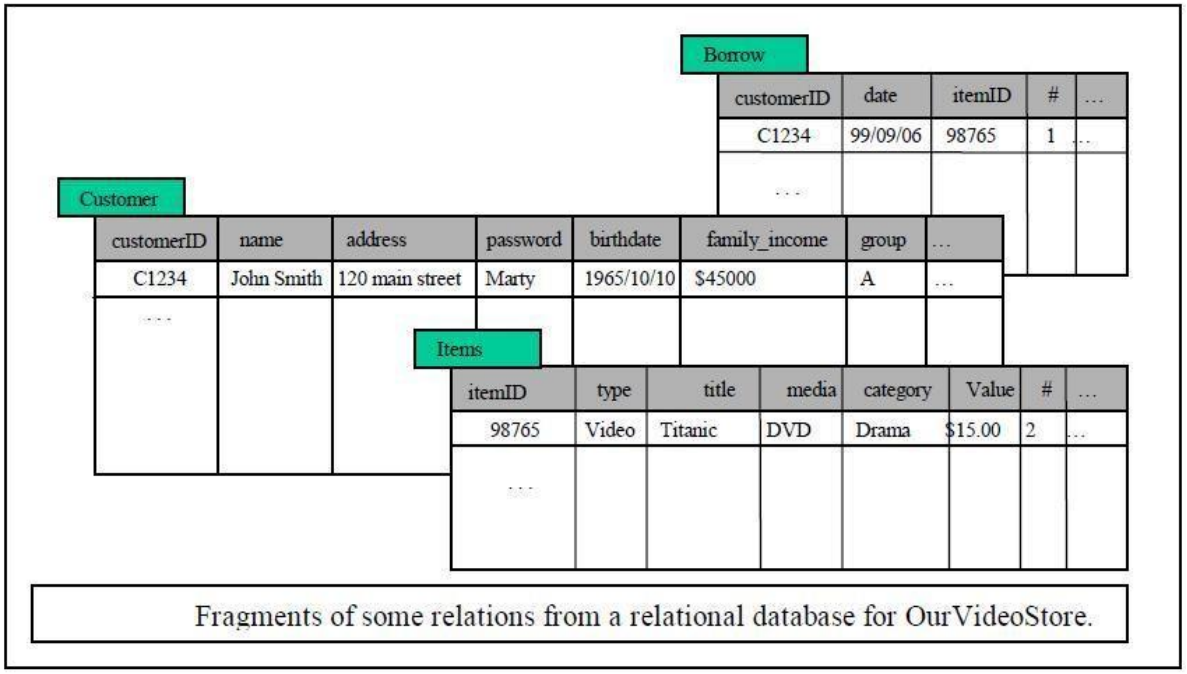
Data mining: on what kind of data? / Describe the following advanced database systems and applications: object-relational databases, spatial databases, text databases, multimedia databases, the World Wide Web.

In principle, data mining should be applicable to any kind of information repository. This includes relational databases, data warehouses, transactional databases, advanced database systems,
flat files, and the World-Wide Web. Advanced database systems include object-oriented and object-relational databases, and special c application-oriented databases, such as

spatial databases, time-series databases, text databases, and multimedia databases.

Flat files: Flat files are actually the most common data source for data mining **algorithms,** especially at the research level. Flat files are simple data files in text or binary format with a structure known by the data mining algorithm to be applied. The data in these files can be transactions, time-series data, scientific measurements, etc.

Relational Databases: a relational database consists of a set of tables containing either values of entity attributes, or values of attributes from entity relationships. Tables have columns and rows, where columns represent attributes and rows represent tuples. A tuple in a relational table corresponds to either an object or a relationship between objects and is identified by a set of attribute values representing a unique key. In following figure it presents some relations Customer, Items, and Borrow representing business activity in a video store. These relations are just a subset of what could be a database for the video store and is given as an example.



Fragments of some relations from a relational database for OurVideoStore.

The most commonly used query language for relational database is SQL, which allows retrieval and manipulation of the data stored in the tables, as well as the calculation of aggregate functions such as average, sum, min, max and count. For instance, an SQL query to select the videos grouped by category would be:

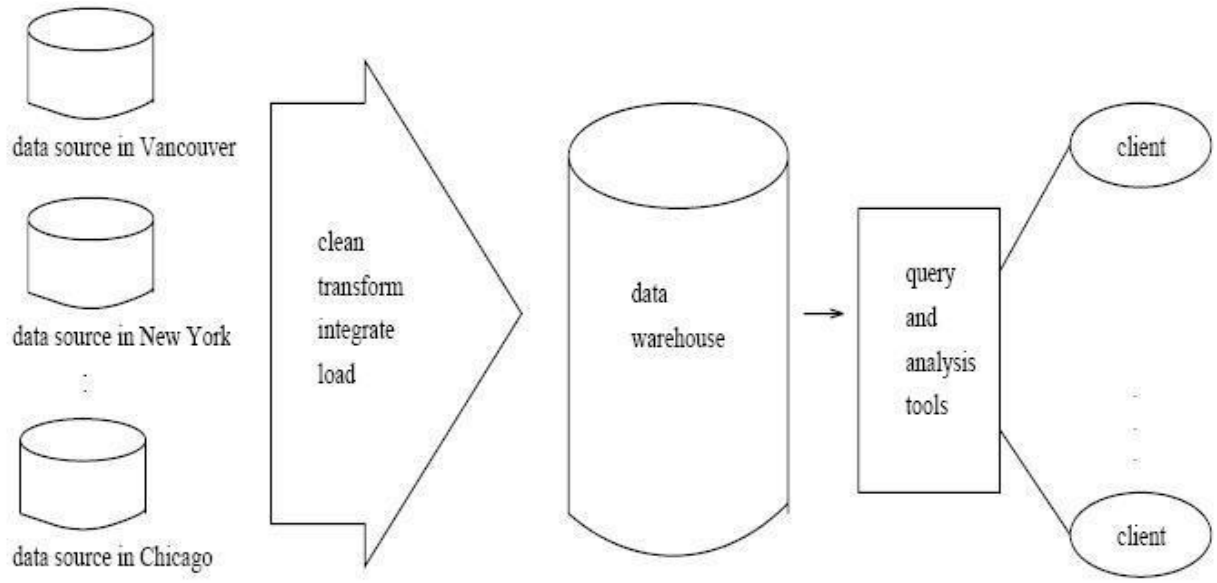
SELECT count(*) FROM Items WHERE type=video GROUP BY category.

Data mining algorithms using relational databases can be more versatile than data mining algorithms specifically written for flat files, since they can take advantage of the structure inherent to relational databases. While data mining can benefit from SQL for data selection, transformation and consolidation, it goes beyond what SQL could provide, such as

predicting, comparing, detecting deviations, etc.

Data warehouses

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and which usually resides at a single site. Data warehouses are constructed via a process of data cleansing, data transformation, data integration, data loading, and periodic data refreshing. The figure shows the basic architecture of a data warehouse.



Architecture of a typical data warehouse.

In order to facilitate decision making, the data in a data warehouse are organized around major subjects, such as customer, item, supplier, and activity. The data are stored to provide information from a historical perspective and are typically summarized.

A data warehouse is usually modeled by a multidimensional database structure, where 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 sales amount. The actual physical structure of a data warehouse may be a relational data store or a multidimensional data cube. It provides a multidimensional view of data and allows the precomputation and fast accessing summarized data.

The data cube structure that stores the primitive or lowest level of information is called a base cuboid. Its corresponding higher level multidimensional (cube) structures are called (non-base) cuboids. A base cuboid together with all of its corresponding higher level cuboids form a data cube. By providing multidimensional data views and the precomputation of summarized data, data warehouse systems are well suited for On-Line

Analytical Processing, or OLAP. OLAP operations make use of background knowledge regarding the domain of the data being studied in order to allow the presentation of data at different levels of abstraction. Such operations accommodate different user viewpoints. Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization, as illustrated in above figure.

Transactional databases

In general, a transactional database consists of a flat file where each record represents a transaction. A transaction typically includes a unique transaction identity number (trans ID), and a list of the items making up the transaction (such as items purchased in a store) as shown below:

SALES

Trans-ID	List of item_ID's
T100	I1,I3,I8
……..	……

Advanced database systems and advanced database applications

An objected-oriented database is designed based on the object-oriented **programming** paradigm where data are a large number of objects organized into classes and class hierarchies. Each entity in the database is considered as an object. The object contains a set of variables that describe the object, a set of messages that the object can use to communicate with other objects or with the rest of the database system and a set of methods where each method holds the code to implement a message.

A spatial database contains spatial-related data, which may be represented in the form of raster or vector data. Raster data consists of n-dimensional bit maps or pixel maps, and vector data are represented by lines, points, polygons or other kinds of processed primitives, Some examples of spatial databases include geographical (map) databases, VLSI chip designs, and medical and satellite images databases.

Time-Series Databases: Time-series databases contain time related data such stock market data or logged activities. These databases usually have a continuous flow of new data coming in, which sometimes causes the need for a challenging real time analysis. Data mining in such databases commonly includes the study of trends and correlations between evolutions of different variables, as well as the prediction of trends and movements of the variables in time.

A text database is a database that contains text documents or other word descriptions in the form of long sentences or paragraphs, such as product specifications, error or bug reports, warning messages, summary reports, notes, or other documents.

A multimedia database stores images, audio, and video data, and is used in applications such as picture content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces.

The World-Wide Web provides rich, world-wide, on-line information **services,** where data objects are linked together to facilitate interactive access. Some examples of distributed information services associated with the World-Wide Web include America Online, Yahoo!, AltaVista, and Prodigy.

Data mining functionalities/Data mining tasks: what kinds of patterns can be mined?

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

Descriptive
predictive

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

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

1 .4.1 **Concept/class description: characterization and discrimination**

Data can be associated with classes or concepts. It describes a given set of data in a concise and summarative manner, presenting interesting general properties of the data. These descriptions can be derived via

data characterization, by summarizing the data of the class under study
(often called the target class)

data discrimination, by comparison of the target class with one or a set of comparative classes
both data characterization and discrimination

Data characterization

It is a summarization of the general characteristics or features of a target class of data.

Example:

A data mining system should be able to produce a description summarizing the characteristics of a student who has obtained more than 75% in every semester; the result could be a general profile of the student.

Data Discrimination is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes.

Example

The general features of students with high GPA's may be compared with the general features of students with low GPA's. The resulting description could be a general comparative profile of the students such as 75% of the students with high GPA's are fourth- year computing science students while 65% of the students with low GPA's are not.

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.

Discrimination descriptions expressed in rule form are referred to as discriminant rules.

Mining Frequent Patterns, Association and Correlations

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

major(X, "computing science"") ⇒ owns(X, "personal computer")

[support = 12%, confidence = 98%]

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

Example:

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

Classification and prediction

Classification:

Classification:

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

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

Example:

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

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

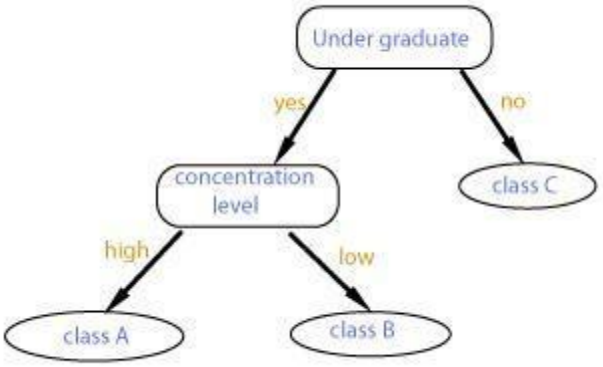
1) IF-THEN rules,

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

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

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

2) Decision Tree



Prediction:

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

Example:

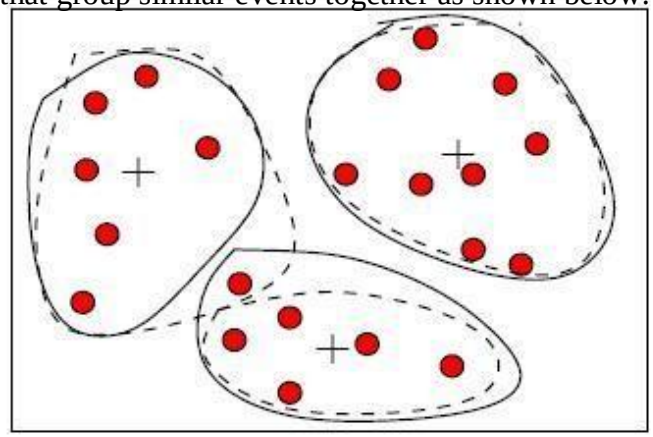
Predicting flooding is difficult problem. One approach is uses monitors placed at various points in the river. These monitors collect data relevant to flood prediction: water level, rain amount, time, humidity etc. These water levels at a potential flooding point in the river can be predicted based on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

Classification vs. Prediction

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

Clustering analysis

Clustering analyzes data objects without consulting a known class label. The objects are clustered or grouped based on the principle of maximizing the intra class similarity and minimizing the interclass similarity. Each cluster that is formed can be viewed as a class of objects.
Clustering can also facilitate taxonomy formation, that is, the organization of observations into a hierarchy of classes that group similar events together as shown below:



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

Example:

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

Classification vs. Clustering

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

Outlier analysis: A database may contain data objects that do not comply with general model of data. These data objects are outliers. In other words, the data objects which do not fall within the cluster will be called as outlier data objects. Noisy data or

exceptional data are also called as outlier data. The analysis of outlier data is referred to as outlier mining.

Example
Outlier analysis may uncover fraudulent usage of credit cards by detecting purchases of extremely 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 location and type of purchase, or the purchase frequency.

Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time.

Example:
The data of result the last several years of a college would give an idea if quality of graduated produced by it

Correlation analysis

Correlation analysis is a technique use to measure the association between two variables. A **correlation coefficient** (**r**) is a statistic used for measuring the strength of **a** supposed linear association between two variables. Correlations range from -1.0 to +1.0 in value.

A correlation coefficient of 1.0 indicates a perfect positive relationship in which high values of one variable are related perfectly to high values in the other variable, and conversely, low values on one variable are perfectly related to low values on the other variable.

A correlation coefficient of 0.0 indicates no relationship between the two variables. That is, one cannot use the scores on one variable to tell anything about the scores on the second variable.

A correlation coefficient of -1.0 indicates a perfect negative relationship in which high values of one variable are related perfectly to low values in the other variables, and conversely, low values in one variable are perfectly related to high values on the other variable.

What is the difference between discrimination and classification? Between characterization and clustering? Between classification and prediction? For each of these pairs of tasks, how are they similar?

Answer:

• Discrimination differs from classification in that the former refers to a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes, while the latter is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts for the purpose of being able to use the model to predict the class of objects whose class label is unknown.

Discrimination and classification are similar in that they both deal with the analysis of class data objects.

Characterization differs from clustering in that the former refers to a summarization of the general characteristics or features of a target class of data while the latter deals with the analysis of data objects without consulting a known class label. This pair of tasks is similar in that they both deal with grouping together objects or data that are related or have high similarity in comparison to one another.

Classification differs from prediction in that the former is the process of finding a set of models (or functions) that describe and distinguish data class or concepts while the latter predicts missing or unavailable, and often numerical, data values. This pair of tasks is similar in that they both are tools for

Prediction: Classification is used for predicting the class label of data objects and prediction is typically used for predicting missing numerical data values.

Are all of the patterns interesting? / What makes a pattern interesting?

A pattern is interesting if,

It is easily understood by humans,

Valid on new or test data with some degree of certainty,

Potentially useful, and

Novel.

A pattern is also interesting if it validates a hypothesis that the user sought to confirm. An interesting pattern represents knowledge.

- **Objective vs. subjective interestingness measures**
  - Objective: based on statistics and structures of patterns, e.g., support, confidence, etc.
  - Subjective: based on user's belief in the data, e.g., unexpectedness, novelty, actionability, etc.

Classification of data mining systems

There are many data mining systems available or being developed. Some are specialized systems dedicated to a given data source or are confined to limited data mining functionalities, other are more versatile and comprehensive. Data mining systems can be categorized according to various criteria among other classification are the following:

Classification according to the type of data source mined: this classification categorizes data mining systems according to the type of data handled such as spatial data, multimedia data, time-series data, text data, World Wide Web, etc.

Classification according to the data model drawn on: this classification categorizes data mining systems based on the data model involved such as relational database, object- oriented database, data warehouse, transactional, etc.

Classification according to the king of knowledge discovered: this classification categorizes data mining systems based on the kind of knowledge discovered or data mining functionalities, such as characterization, discrimination, association, classification, clustering, etc. Some systems tend to be comprehensive systems offering several data mining functionalities together.

Classification according to mining techniques used: Data mining systems **employ** and provide different techniques. This classification categorizes data mining systems according to the data analysis approach used such as machine learning, neural networks, genetic algorithms, statistics, visualization, database oriented or data warehouse-oriented, etc. The classification can also take into account the degree of user interaction involved in the data mining process such as query-driven systems, interactive exploratory systems, or autonomous systems. A comprehensive system would provide a wide variety of data mining techniques to fit different situations and options, and offer different degrees of user interaction.

Five primitives for specifying a data mining task

Task-relevant data: This primitive specifies the data upon which mining **is** to be performed. It involves specifying the database and tables or data warehouse containing the relevant data, conditions for selecting the relevant data, the relevant attributes or dimensions for exploration, and instructions regarding the ordering or grouping of the data retrieved.

Knowledge type to be mined: This primitive specifies the specific data mining function to be performed, such as characterization, discrimination, association, classification, clustering, or evolution analysis. As well, the user can be more specific and provide pattern templates that all discovered patterns must match. These templates or meta patterns (also called meta rules or meta queries), can be used to guide the discovery process.

Background knowledge: This primitive allows users to specify knowledge they have about the domain to be mined. Such knowledge can be used to guide the knowledge discovery process and evaluate the patterns that are found. Of the several kinds of background knowledge, this chapter focuses on concept hierarchies.

Pattern interestingness measure: This primitive allows users to specify functions that are used to separate uninteresting patterns from knowledge and may be used to guide the mining process, as well as to evaluate the discovered patterns. This allows the user to confine the number of uninteresting patterns returned by the process, as a data mining process may generate a large number of patterns. Interestingness measures can be specified for such pattern characteristics as simplicity, certainty, utility and novelty.

Visualization of discovered patterns: This primitive refers to the form in which discovered patterns are to be displayed. In order for data mining to be effective in conveying knowledge to users, data mining systems should be able to display the discovered patterns in multiple forms such as rules, tables, cross tabs (cross-tabulations), pie or bar charts, decision trees, cubes or other visual representations.

Integration of a Data Mining System with a Database or Data Warehouse System

The differences between the following architectures for the integration of a data mining system with a database or data warehouse system are as follows.

• No coupling:

The data mining system uses sources such as flat files to obtain the initial data set to be mined since no database system or data warehouse system functions are implemented as part of the process. Thus, this architecture represents a poor design choice.

• Loose coupling:

The data mining system is not integrated with the database or data warehouse system beyond their use as the source of the initial data set to be mined, and possible use in storage of the results. Thus, this architecture can take advantage of the flexibility, efficiency and features such as indexing that the database and data warehousing systems may provide. However, it is difficult for loose coupling to achieve high scalability and good performance with large data sets as many such systems are memory-based.

• Semitight coupling:

Some of the data mining primitives such as aggregation, sorting or pre computation of statistical functions are efficiently implemented in the database or data warehouse system, for use by the data mining system during mining-query processing. Also, some frequently used inter mediate mining results can be pre computed and stored in the database or data warehouse system, thereby enhancing the performance of the data mining system.

• Tight coupling:

The database or data warehouse system is fully integrated as part of the data mining system and thereby provides optimized data mining query processing. Thus, the data

mining sub system is treated as one functional component of an information system. This is a highly desirable architecture as it facilitates efficient implementations of data mining functions, high system performance, and an integrated information processing environment

From the descriptions of the architectures provided above, it can be seen that tight coupling is the best alternative without respect to technical or implementation issues. However, as much of the technical infrastructure needed in a tightly coupled system is still evolving, implementation of such a system is non-trivial. Therefore, the most popular architecture is currently semi tight coupling as it provides a compromise between loose and tight coupling.

Major issues in data mining

Major issues in data mining is regarding mining methodology, user interaction, performance, and diverse data types

1 Mining methodology and user-interaction issues:

_ Mining different kinds of knowledge in databases: Since different users can be interested in different kinds of knowledge, data mining should cover a wide spectrum of data analysis and knowledge discovery tasks, including data characterization, discrimination, association, classification, clustering, trend and deviation analysis, and similarity analysis. These tasks may use the same database in different ways and require the development of numerous data mining techniques.

_ Interactive mining of knowledge at multiple levels of abstraction: Since **it** is difficult to know exactly what can be discovered within a database, the data mining process should be interactive.

_ Incorporation of background knowledge: Background knowledge, or information regarding the domain under study, may be used to guide the discovery patterns. Domain knowledge related to databases, such as integrity constraints and deduction rules, can help focus and speed up a data mining process, or judge the interestingness of discovered patterns.

_ Data mining query languages and ad-hoc data mining: Knowledge in Relational query languages (such as SQL) required since it allow users to pose ad-hoc queries for data retrieval.

_ Presentation and visualization of data mining results: Discovered knowledge should be expressed in high-level languages, visual representations, so that the knowledge can be easily understood and directly usable by humans

_ Handling outlier or incomplete data: The data stored in a database may reflect outliers: noise, exceptional cases, or incomplete data objects. These objects may confuse the analysis process, causing over fitting of the data to the knowledge model constructed. As a result, the accuracy of the discovered patterns can be poor. Data cleaning methods and data analysis methods which can handle outliers are required.

_ Pattern evaluation: refers to interestingness of pattern: A data mining system can uncover thousands of patterns. Many of the patterns discovered may be uninteresting to the given user, representing common knowledge or lacking novelty. Several challenges remain regarding the development of techniques to assess the interestingness of discovered patterns,

2. Performance issues. These include efficiency, scalability, and parallelization **of** data mining algorithms.

_ Efficiency and scalability of data mining algorithms: To effectively extract **information** from a huge amount of data in databases, data mining algorithms must be efficient and scalable.

_ Parallel, distributed, and incremental updating algorithms: Such **algorithms** divide the data into partitions, which are processed in parallel. The results from the partitions are then merged.

3. Issues relating to the diversity of database types

_ Handling of relational and complex types of data: Since relational databases **and** data warehouses are widely used, the development of efficient and effective data mining systems for such data is important.

_ Mining information from heterogeneous databases and global information systems: Local and wide-area computer networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases. The discovery of knowledge from different sources of structured, semi-structured, or unstructured data with diverse data semantics poses great challenges to data mining.

Data preprocessing

Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user.

Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user

Why Data Preprocessing?

Data in the real world is dirty. It can be in incomplete, noisy and inconsistent from. These data needs to be preprocessed in order to help improve the quality of the data, and quality of the mining results.

If no quality data , then no quality mining results. The quality decision is always based on the quality data.
If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult

Incomplete data: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. e.g., occupation=“ ”.

Noisy data: containing errors or outliers data. e.g., Salary=“-10”

Inconsistent data: containing discrepancies in codes or names. e.g., Age=“42” Birthday=“03/07/1997”

Incomplete data may come from
- “Not applicable” data value when collected
- Different considerations between the time when the data was collected and when it is analyzed.
- Human/hardware/software problems

Noisy data (incorrect values) may come from
- Faulty data collection by instruments
- Human or computer error at data entry
- Errors in data transmission

Inconsistent data may come from
- Different data sources
- Functional dependency violation (e.g., modify some linked data)

Major Tasks in Data Preprocessing

Data cleaning

Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies

Data integration

Integration of multiple databases, data cubes, or files Data

transformation

Normalization and aggregation

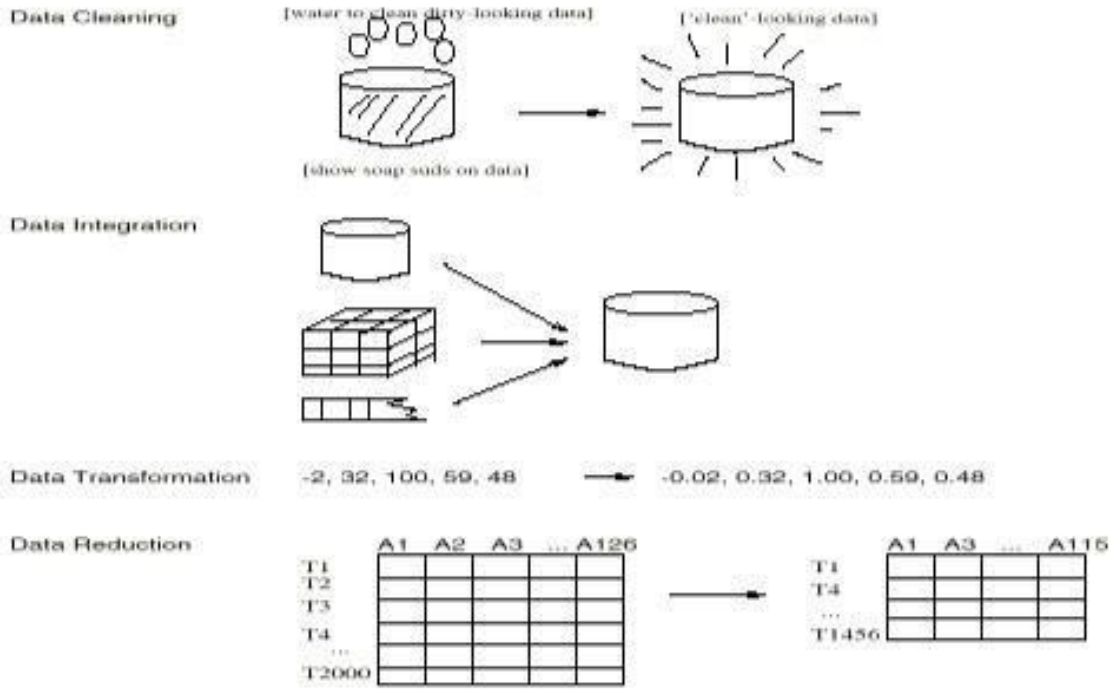
Data reduction

Obtains reduced representation in volume but produces the same or similar analytical results

Data discretization

Part of data reduction but with particular importance, especially for numerical data

Forms of Data Preprocessing



Descriptive Data Summarization

Categorize the measures

- A measure is distributive, if we can partition the dataset into smaller subsets, compute the measure on the individual subsets, and then combine the partial results in order to arrive at the measure's value on the entire (original) dataset
- A measure is algebraic if it can be computed by applying an algebraic function to one or more distributive measures
- A measure is holistic if it must be computed on the entire dataset as a whole

Measure the Central Tendency

A measure of central tendency is a single value that attempts to describe a set of data by identifying the central position within that set of data. As such, measures of central tendency are sometimes called measures of central location.

In other words, in many real-life situations, it is helpful to describe data by a single number that is most representative of the entire collection of numbers. Such a number is called a measure of central tendency. The most commonly used measures are as follows. **Mean, Median, and Mode**

Mean: mean, or average, of numbers is the sum of the numbers divided by n. That is:

$$\bar{x} = \frac{(x_1 + x_2 + \cdots + x_n)}{n} \quad \text{i.e.,} \quad \text{Mean} = \frac{\text{Sum of all data values}}{\text{Number of data values}}$$

shortly,

$$\bar{x} = \frac{\sum x}{n}$$

where $\bar{x}$ (read as 'x bar') is the mean of the set of x values,
$\sum x$ is the sum of all the x values, and
n is the number of x values.

Example 1

The marks of seven students in a mathematics test with a maximum possible mark of 20 are given below:

15 13 18 16 14 17 12

Find the mean of this set of data values.

Solution:

$$\text{Mean} = \frac{\text{Sum of all data values}}{\text{Number of data values}}$$
$$= \frac{15+13+18+16+14+17+12}{7}$$
$$= \frac{105}{7}$$
$$= 15$$

So, the mean mark is 15.

Midrange

The midrange of a data set is the average of the minimum and maximum values.

Median: median of numbers is the middle number when the numbers are written in order. If is even, the median is the average of the two middle numbers.

Example 2

The marks of nine students in a geography test that had a maximum possible mark of 50 are given below:

47 35 37 32 38 39 36 34 35

Find the median of this set of data values.

Solution:

Arrange the data values in order from the lowest value to the highest value: 32

34 35 35 36 37 38 39 47

The fifth data value, 36, is the middle value in this arrangement.

$\therefore$ Median = 36

Note:

The number of values, n, in the data set = 9

$$\text{Median} = \frac{1}{2}(9+1)\text{ th value}$$
$$= 5\text{th value}$$
$$= 36$$

In general:

$$\text{Median} = \frac{1}{2}(n+1)\text{ th value, where } n \text{ is the number of data values in the sample}$$

If the number of values in the data set is even, then the **median** is the average of the two middle values.

Example 3

Find the median of the following data set:
12 18 16 21 10 13 17 19

Solution:

Arrange the data values in order from the lowest value to the highest value: 10 12 13 16 17 18 19 21

The number of values in the data set is 8, which is even. So, the median is the average of the two middle values.

$$\therefore \text{Median} = \frac{\text{4th data value} + \text{5th data value}}{2}$$
$$= \frac{16+17}{2}$$
$$= \frac{33}{2}$$
$$= 16.5$$

Trimmed mean

A trimming mean eliminates the extreme observations by removing observations from each end of the ordered sample. It is calculated by discarding a certain percentage of the lowest and the highest scores and then computing the mean of the remaining scores.

Mode of numbers is the number that occurs most frequently. If two numbers tie for most frequent occurrence, the collection has two modes and is called bimodal.

The mode has applications in printing . For example, it is important to print more of the most popular books; because printing different books in equal numbers would cause a shortage of some books and an oversupply of others.

Likewise, the mode has applications in manufacturing. For example, it is important to manufacture more of the most popular shoes; because manufacturing different shoes in equal numbers would cause a shortage of some shoes and an oversupply of others.

Example 4

Find the mode of the following data set:

48 44 48 45 42 49 48

Solution:

The mode is 48 since it occurs most often.

- It is possible for a set of data values to have more than one mode.
- If there are two data values that occur most frequently, we say that the set of data values is **bimodal**.
- If there is three data values that occur most frequently, we say that the set of data values is **trimodal**
- If two or more data values that occur most frequently, we say that the set of data values is **multimodal**
- If there is no data value or data values that occur most frequently, we say that the set of data values has no mode.

The mean, median and mode of a data set are collectively known as measures of **central tendency** as these three measures focus on where the data is centered or clustered. To analyze data using the mean, median and mode, we need to use the most appropriate measure of central tendency. The following points should be remembered:

- The mean is useful for predicting future results when there are no extreme values in the data set. However, the impact of extreme values on the mean may be important and should be considered. E.g. the impact of a stock market crash on average investment returns.
- The median may be more useful than the mean when there are extreme values in the data set as it is not affected by the extreme values.
- The mode is useful when the most common item, characteristic or value of a data set is required.

Measures of Dispersion

Measures of dispersion measure how spread out a set of data is. The two most commonly used measures of dispersion are the variance and the standard deviation. Rather than showing how data are similar, they show how data differs from its variation, spread, or dispersion. Other measures of dispersion that may be encountered include the Quartiles, Inter quartile range (IQR), Five number summary, range and box plots

Variance and Standard Deviation

Very different sets of numbers can have the same mean. You will now study two measures of dispersion, which give you an idea of how much the numbers in a set differ from the mean of the set. These two measures are called the variance of the set and the standard deviation of the set

Consider a set of numbers $\{x_1, x_2, \ldots, x_n\}$ with a mean of $\bar{x}$. The **variance** of the set is

$$v = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_n - \bar{x})^2}{n}$$

and the **standard deviation** of the set is $\sigma = \sqrt{v}$ (σ is the lowercase Greek letter *sigma*).

The standard deviation of a set is a measure of how much a typical number in the set differs from the mean. The greater the standard deviation, the more the numbers in the set *vary* from the mean. For instance, each of the following sets has a mean of 5.

$\{5, 5, 5, 5\}$, $\{4, 4, 6, 6\}$, and $\{3, 3, 7, 7\}$

The standard deviations of the sets are 0, 1, and 2.

$$\sigma_1 = \sqrt{\frac{(5-5)^2 + (5-5)^2 + (5-5)^2 + (5-5)^2}{4}}$$
$$= 0$$
$$\sigma_2 = \sqrt{\frac{(4-5)^2 + (4-5)^2 + (6-5)^2 + (6-5)^2}{4}}$$
$$= 1$$
$$\sigma_3 = \sqrt{\frac{(3-5)^2 + (3-5)^2 + (7-5)^2 + (7-5)^2}{4}}$$
$$= 2$$

Percentile

Percentiles are values that divide a sample of data into one hundred groups containing (as far as possible) equal numbers of observations.

The pth percentile of a distribution is the value such that p percent of the observations fall at or below it.

The most commonly used percentiles other than the median are the 25th percentile and the 75th percentile.

The 25th percentile demarcates the first quartile, the median or 50th percentile demarcates the second quartile, the 75th percentile demarcates the third quartile, and the 100^{th} percentile demarcates the fourth quartile.

Quartiles

Quartiles are numbers that divide an ordered data set into four portions, each containing approximately one-fourth of the data. Twenty-five percent of the data values come before the first quartile (Q1). The median is the second quartile (Q2); 50% of the data values come before the median. Seventy-five percent of the data values come before the third quartile (Q3).

Q1=25^{th} percentile=(n*25/100), where n is total number of data in the given data set

Q2=median=50^{th} percentile=(n*50/100)

Q3=75^{th} percentile=(n*75/100)

Inter quartile range (IQR)

The inter quartile range is the length of the interval between the lower quartile (Q1) and the upper quartile (Q3). This interval indicates the central, or middle, 50% of a data set.

$$IQR = Q_3 - Q_1$$

Range

The range of a set of data is the difference between its largest (maximum) and smallest (minimum) values. In the statistical world, the range is reported as a single number, the difference between maximum and minimum. Sometimes, the range is often reported as "from (the minimum) to (the maximum)," i.e., two numbers.

Example1:

Given data set: 3, 4, 4, 5, 6, 8

The range of data set is 3–8. The range gives only minimal information about the spread of the data, by defining the two extremes. It says nothing about how the data are distributed between those two endpoints.

Example2:

In this example we demonstrate how to find the minimum value, maximum value, and range of the following data: 29, 31, 24, 29, 30, 25

Arrange the data from smallest to largest.

24, 25, 29, 29, 30, 31

Identify the minimum and maximum

values: Minimum = 24, Maximum = 31

3. Calculate the range:

Range = Maximum-Minimum = 31–24 = 7.

Thus the range is 7.

Five-Number Summary

The Five-Number Summary of a data set is a five-item list comprising the minimum value, first quartile, median, third quartile, and maximum value of the set.

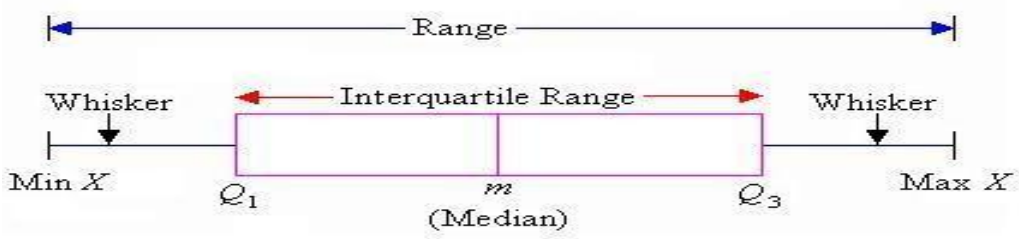
{MIN, Q1, MEDIAN (Q2), Q3, MAX}

Box plots

A box plot is a graph used to represent the range, median, quartiles and inter quartile range of a set of data values.

Constructing a Box plot: To construct a box plot:

Draw a box to represent the middle 50% of the observations of the data set.
Show the median by drawing a vertical line within the box.
Draw the lines (called **whiskers**) from the lower and upper ends of the box to the minimum and maximum values of the data set respectively, as shown in the following diagram.



- *X* is the set of data values.
- Min *X* is the minimum value in the data set.
- Max *X* is the maximum value in the data set.

Example: Draw a boxplot for the following data set of scores:

76 79 76 74 75 71 85 82 82 79 81

Step 1: Arrange the score values in ascending order of magnitude: 71

74 75 76 76 79 79 81 82 82 85

There are 11 values in the data set.

Step 2: Q1=25th percentile value in the given data set

Q1=11*(25/100) th value

=2.75 =>3rd value

=75

Step 3: Q2=median=50th percentile value

=11 * (50/100) th value

=5.5th value => 6th value

=79

Step 4: Q3=75th percentile value

=11*(75/100)th value

=8.25th value=>9th value

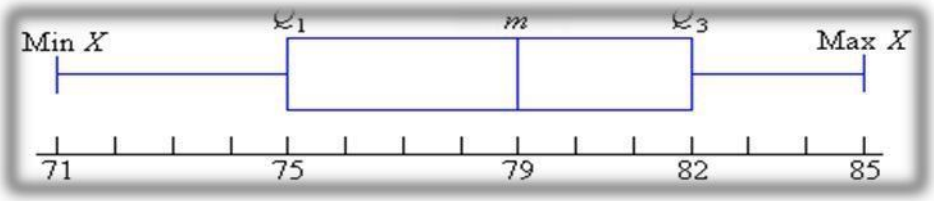
= 82

Step 5: Min X= 71

Step 6: Max X=85

Step 7: Range= 85-71 = 14

Step 5: IQR=height of the box=Q3-Q1=9-3=6th value=79



Since the medians represent the middle points, they split the data into four equal parts. In other words:

- one quarter of the data numbers are less than 75
- one quarter of the data numbers are between 75 and 79
- one quarter of the data numbers are between 79 and 82
- one quarter of the data numbers are greater than 82

Outliers

Outlier data is a data that falls outside the range. Outliers will be any points below $Q_1 - 1.5\times IQR$ or above $Q_3 + 1.5\times IQR$.

Example:

Find the outliers, if any, for the following data set:

10.2, 14.1, 14.4, **14.4**, 14.4, 14.5, 14.5, **14.6,** 14.7, 14.7, 14.7, **14.9,** 15.1, 15.9, 16.4

To find out if there are any outliers, I first have to find the IQR. There are fifteen data points, so the median will be at position (15/2) = 7.5=8th value=14.6. That is, Q_2 = 14.6.

Q_1 is the fourth value in the list and Q_3 is the twelfth: Q_1 = 14.4 and Q_3 = 14.9.

Then IQR = 14.9 – 14.4 = 0.5.

Outliers will be any points below:

$Q_1 – 1.5\times IQR = 14.4 – 0.75 = 13.65$ or above $Q_3 + 1.5\times IQR = 14.9 + 0.75 = 15.65$.

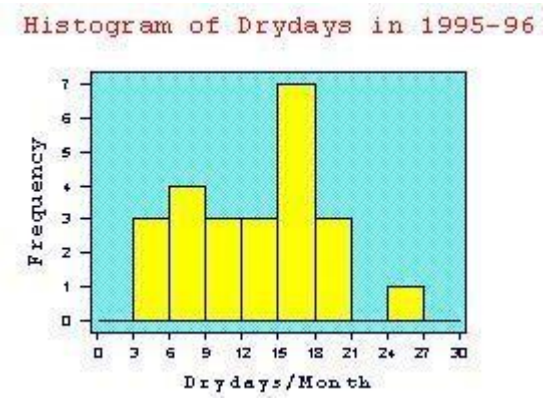
Then the outliers are at 10.2, 15.9, and 16.4.

The values for $Q_1 – 1.5\times IQR$ and $Q_3 + 1.5\times IQR$ are the "fences" that mark off the "reasonable" values from the outlier values. Outliers lie outside the fences.

Graphic Displays of Basic Descriptive Data Summaries 1

Histogram

A histogram is a way of summarizing data that are measured on an interval scale (either discrete or continuous). It is often used in exploratory data analysis to illustrate the major features of the distribution of the data in a convenient form. It divides up the range of possible values in a data set into classes or groups. For each group, a rectangle is constructed with a base length equal to the range of values in that specific group, and an area proportional to the number of observations falling into that group. This means that the rectangles might be drawn of non-uniform height.



The histogram is only appropriate for variables whose values are numerical and measured on an interval scale. It is generally used when dealing with large data sets (>100 observations)

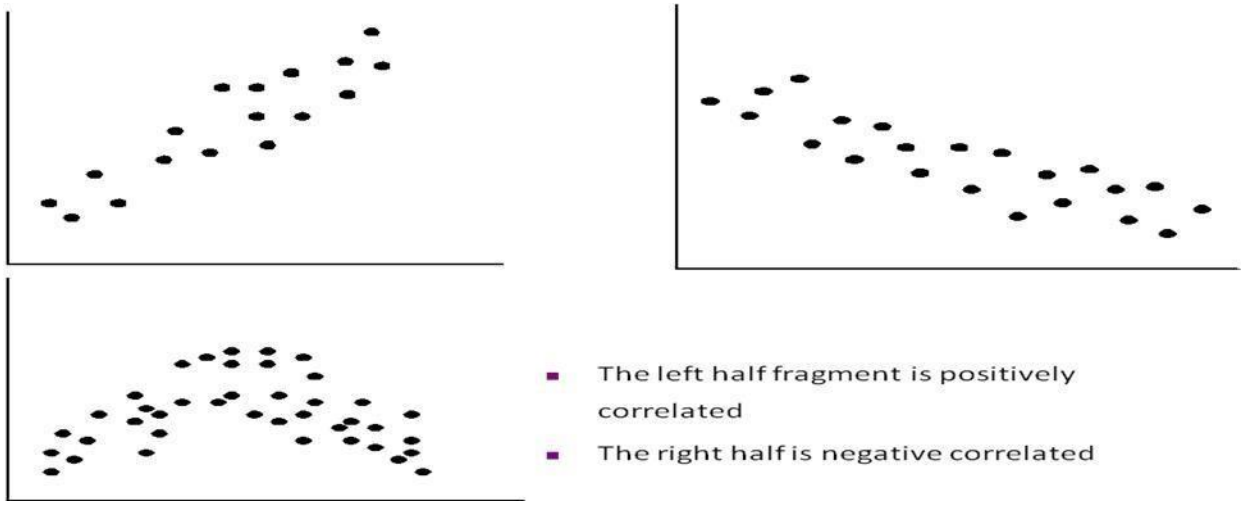
A histogram can also help detect any unusual observations (outliers), or any gaps in the data set.

2 Scatter Plot

A scatter plot is a useful summary of a set of bivariate data (two variables), usually drawn before working out a linear correlation coefficient or fitting a regression line. It gives a good visual picture of the relationship between the two variables, and aids the interpretation of the correlation coefficient or regression model.

Each unit contributes one point to the scatter plot, on which points are plotted but not joined. The resulting pattern indicates the type and strength of the relationship between the two variables.

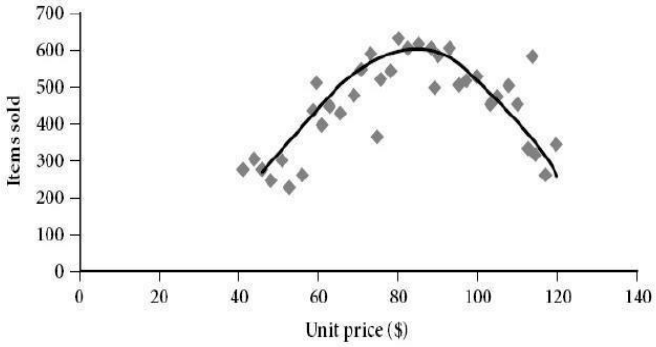
Positively and Negatively Correlated Data



A scatter plot will also show up a non-linear relationship between the two variables and whether or not there exist any outliers in the data.

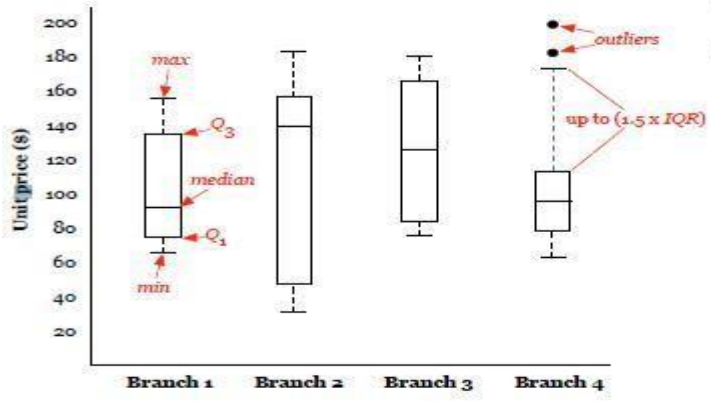
3 Loess curve

It is another important exploratory graphic aid that adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence. The word loess is short for "local regression."



4 Box plot

The picture produced consists of the most extreme values in the data set (maximum and minimum values), the lower and upper quartiles, and the median.

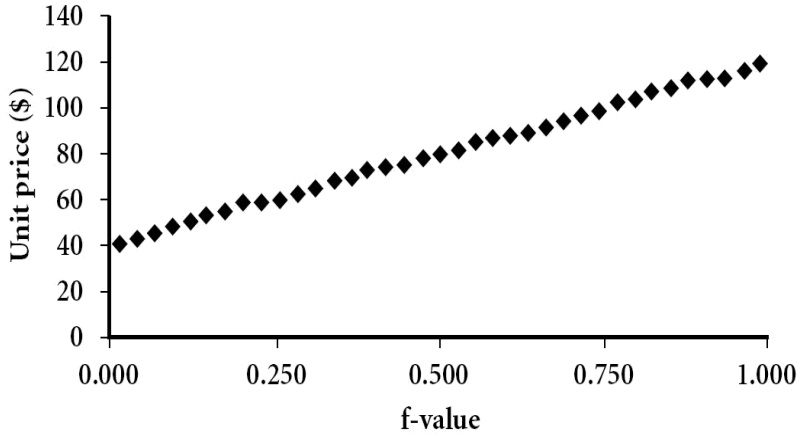


5 Quantile plot

Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)

Plots quantile information

For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i% of the data are below or equal to the value x_i



The *f* quantile is the data value below which approximately a decimal fraction *f* of the data is found. That data value is denoted q(f). Each data point can be assigned an *f*-value. Let a time series *x* of length *n* be sorted from smallest to largest values, such that the sorted values have rank. The *f*-value for each observation is computed as . 1,2,..., n . The *f*-value for each observation is computed as, $f_i = \frac{i - 0.5}{n}$

6 Quantile-Quantile plots (Q-Q plot)

Quantile-quantile plots allow us to compare the quantiles of two sets of numbers.

This kind of comparison is much more detailed than a simple comparison of means or medians.

A normal distribution is often a reasonable model for the data. Without inspecting the data, however, it is risky to assume a normal distribution. There are a number of graphs that can be used to check the deviations of the data from the normal distribution. The most useful tool for assessing normality is a quantile or QQ plot. This is a scatter plot with the quantiles of the scores on the horizontal axis and the expected normal scores on the vertical axis.

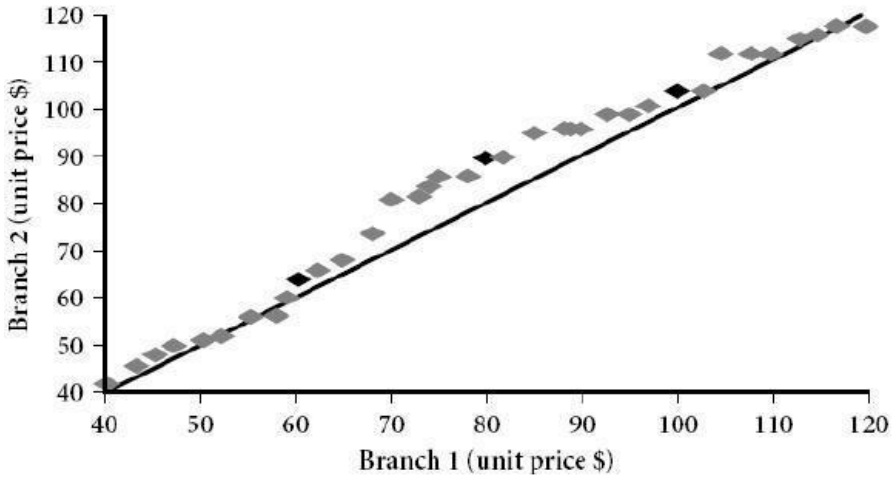
In other words, it is a graph that shows the quantiles of one univariate distribution against the corresponding quantiles of another. It is a powerful visualization tool in that it allows the user to view whether there is a shift in going from one distribution to another.

The steps in constructing a QQ plot are as follows:

First, we sort the data from smallest to largest. A plot of these scores against the expected normal scores should reveal a straight line.

The expected normal scores are calculated by taking the z-scores of (I - ½)/n where I is the rank in increasing order.

Curvature of the points indicates departures of normality. This plot is also useful for detecting outliers. The outliers appear as points that are far away from the overall pattern op points



How is a quantile-quantile plot different from a quantile plot?

A quantile plot is a graphical method used to show the approximate percentage of values below or equal to the independent variable in a univariate distribution. Thus, it displays quantile information for all the data, where the values measured for the independent variable are plotted against their corresponding quantile.

A quantile-quantile plot however, graphs the quantiles of one univariate distribution against the corresponding quantiles of another univariate distribution. Both axes display the range of values measured for their corresponding distribution, and points are plotted that correspond to the quantile values of the two distributions. A line (y = x) can be added to the graph along with points representing where the first, second and third quantiles lie, in order to increase the graph's informational value. Points that lie above such a line indicate a correspondingly higher value for the distribution plotted on the y-axis, than for the distribution plotted on the x-axis at the same quantile. The opposite effect is true for points lying below this line.

Data Cleaning

Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

Various methods for handling this problem:

Missing Values

The various methods for handling the problem of missing values in data tuples include:

Ignoring the tuple: This is usually done when the class label is missing (assuming the mining task involves classification or description). 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.

Manually filling in the missing value: In general, this approach is time-consuming and may not be a reasonable task for large data sets with many missing values, especially when the value to be filled in is not easily determined.

Using 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 $-\infty$. 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 recommended.

Using the attribute mean for quantitative (numeric) values or attribute mode for categorical (nominal) values, for all samples belonging to the same class as the given tuple: For example, if classifying customers according to credit risk, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple.

Using the most probable value to fill in the missing value: This may be **determined** with regression, inference-based tools using 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.

Noisy data:

Noise is a random error or variance in a measured variable. Data smoothing tech is used for removing such noisy data.

Several Data smoothing techniques:

1 Binning methods: Binning methods smooth a sorted data value by consulting the neighborhood", or values around it. The sorted values are distributed into a number of 'buckets', or bins. Because binning methods consult the neighborhood of values, they perform local smoothing.

In this technique,

The data for first sorted
Then the sorted list partitioned into equi-depth of bins.
Then one can smooth by bin means, smooth by bin median, smooth by bin

boundaries, etc.

Smoothing by bin means: Each value in the bin is replaced by the mean value of the bin.

Smoothing by bin medians: Each value in the bin is replaced by the bin median.

Smoothing by boundaries: The min and max values of a bin are identified as the bin boundaries. Each bin value is replaced by the closest boundary value.

- Example: Binning Methods for Data Smoothing

  - Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
  - Partition into (equi-depth) bins(equi depth of 3 since each bin contains three values):
    - **Bin 1**: 4, 8, 9, 15

      Bin 2:21, 21, 24, 25

      Bin 3:26, 28, 29, 34
  - Smoothing by bin means:
    - **Bin 1**: 9, 9, 9, 9

      Bin 2:23, 23, 23, 23

      Bin 3:29, 29, 29, 29
  - Smoothing by bin boundaries:
    - **Bin 1**: 4, 4, 4, 15

      Bin 2:21, 21, 25, 25

      Bin 3:26, 26, 26, 34

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. Each bin value is then replaced by the closest boundary value.

Suppose that the data for analysis include the attribute age. The age values for the data tuples are (in
increasing order): 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.
(a) Use smoothing by bin means to smooth the above data, using a bin depth of 3. Illustrate your steps.
Comment on the effect of this technique for the given data.

The following steps are required to smooth the above data using smoothing by bin means with a bin
depth of 3.

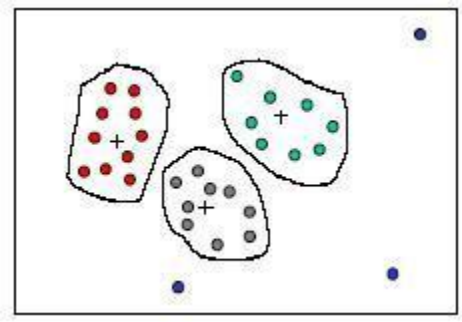
Step 1: Sort the data. (This step is not required here as the data are already sorted.) Step

2: Partition the data into equi-depth bins of depth
3. Bin 1: 13, 15, 16 Bin 2: 16, 19, 20 Bin 3: 20, 21, 22
Bin 4: 22, 25, 25 Bin 5: 25, 25, 30 Bin 6: 33, 33, 35
Bin 7: 35, 35, 35 Bin 8: 36, 40, 45 Bin 9: 46, 52, 70

Step 3: Calculate the arithmetic mean of each bin.

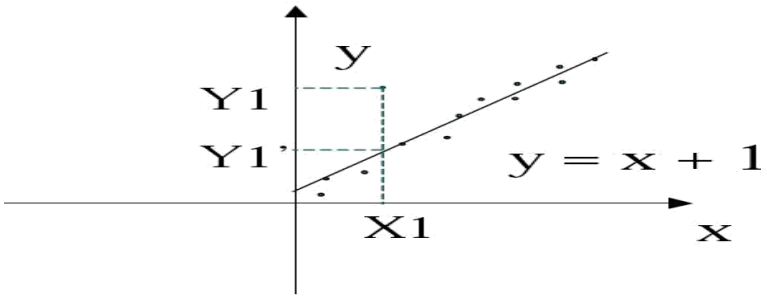
Step 4: Replace each of the values in each bin by the arithmetic mean calculated for the bin.
Bin 1: 14, 14, 14 Bin 2: 18, 18, 18 Bin 3: 21, 21, 21
Bin 4: 24, 24, 24 Bin 5: 26, 26, 26 Bin 6: 33, 33, 33
Bin 7: 35, 35, 35 Bin 8: 40, 40, 40 Bin 9: 56, 56, 56

2 Clustering: Outliers in the data may be detected by clustering, where similar **values** are organized into groups, or 'clusters'. Values that fall outside of the set of clusters may be considered outliers.



3 Regression : smooth by fitting the data into regression functions.

- Linear regression involves finding the best of line to fit two variables, so that one variable can be used to predict the other.



- Multiple linear regression is an extension of linear regression, where more than two variables are involved and the data are fit to a multidimensional surface.

Using regression to find a mathematical equation to fit the data helps smooth out the noise.

Field overloading: is a kind of source of errors that typically occurs when **developers** compress new attribute definitions into unused portions of already defined attributes.

Unique rule is a rule says that each value of the given attribute must be **different** from all other values of that attribute

Consecutive rule is a rule says that there can be no missing values between the **lowest** and highest values of the attribute and that all values must also be unique.

Null rule specifies the use of blanks, question marks, special characters or **other** strings that may indicate the null condition and how such values should be handled.

Data Integration and Transformation

Data Integration

It combines data from multiple sources into a coherent store. There are number of issues to consider during data integration.

Issues:

- **Schema integration:** refers integration of metadata from different sources.
- **Entity identification problem:** Identifying entity in one data source similar to entity in another table. For example, customer_id in one db and customer_no in another db refer to the same entity
- **Detecting and resolving data value conflicts:** Attribute values from different sources can be different due to different representations, different scales. E.g. metric vs. British units
- **Redundancy:** is another issue while performing data integration. Redundancy can occur due to the following reasons:

- Object identification: The same attribute may have different names in different db
- Derived Data: one attribute may be derived from another attribute.

Handling redundant data in data integration

1. **Correlation analysis**

For numeric data

Some redundancy can be identified by correlation analysis. The correlation between two variables A and B can be measured by

$$r_{A,B} = \frac{\Sigma(A-\overline{A})(B-\overline{B})}{(n-1)\sigma_A\sigma_B}$$

$\overline{A}$, $\overline{B}$ are respective mean values of A and B
σ_A, σ_B are respective standard deviation of A and B
n is the number of tubles

- The result of the equation is > 0, then A and B are positively correlated, which means the value of A increases as the values of B increases. The higher value may indicate redundancy that may be removed.
- The result of the equation is = 0, then A and B are independent and there is no correlation between them.
- If the resulting value is < 0, then A and B are negatively correlated where the values of one attribute increase as the value of one attribute decrease which means each attribute may discourages each other.

-also called Pearson's product moment coefficient

For categorical data

- X^2 (chi-square) test

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- The larger the X^2 value, the more likely the variables are related
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

Example:

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- X^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

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

- It shows that like_science_fiction and play_chess are correlated in the group

Data Transformation

Data transformation can involve the following:

- **Smoothing:** which works to remove noise from the data
- **Aggregation**: where summary or aggregation operations are applied to **the** data. For example, the daily sales data may be aggregated so as to compute weekly and annual total scores.
- **Generalization of the data:** where low-level or "primitive" (raw) data **are** replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher-level concepts, like city or country.
- **Normalization:** where the attribute data are scaled so as to fall within a **small** specified range, such as −1.0 to 1.0, or 0.0 to 1.0.
- **Attribute construction (feature construction)**: this is where new **attributes** are constructed and added from the given set of attributes to help the mining process.

Normalization

In which data are scaled to fall within a small, specified range, useful for classification algorithms involving neural networks, distance measurements such as nearest neighbor classification and clustering. There are 3 methods for data normalization. They are:

min-max normalization
z-score normalization
normalization by decimal scaling

Min-max normalization: performs linear transformation on the original data values. It can be defined as,

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

v is the value to be normalized
min_A, max_A are minimum and maximum values of an attribute A
new_max_A, new_min_A are the normalization range.

Z-score normalization / zero-mean normalization: In which values of an attribute A are normalized based on the mean and standard deviation of A. It can be defined as,

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

This method is useful when min and max value of attribute A are unknown or when outliers that are dominate min-max normalization.

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

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that $Max(|v'|)<1$

Data Reduction techniques

These 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. Data reduction

- Data cube aggregation
- Dimension reduction
- Data compression
- Discretization
- Numerosity reduction
  - Regression
  - Histograms
  - Clustering
  - Sampling

Data cube aggregation, where aggregation operations are applied to the data in the construction of a data cube.

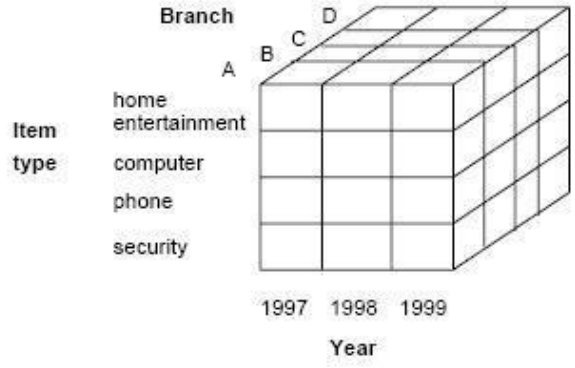
Attribute subset selection, where irrelevant, weakly relevant or redundant attributes or dimensions may be detected and removed.

Dimensionality reduction, where encoding mechanisms are used to reduce the data set size. Examples: Wavelet Transforms Principal Components Analysis **Numerosity reduction**, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need store only the model parameters instead of the actual data) or nonparametric methods such as clustering, sampling, and the use of histograms.

Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Data Discretization is a

form of numerosity reduction that is very useful for the automatic generation of concept hierarchies.

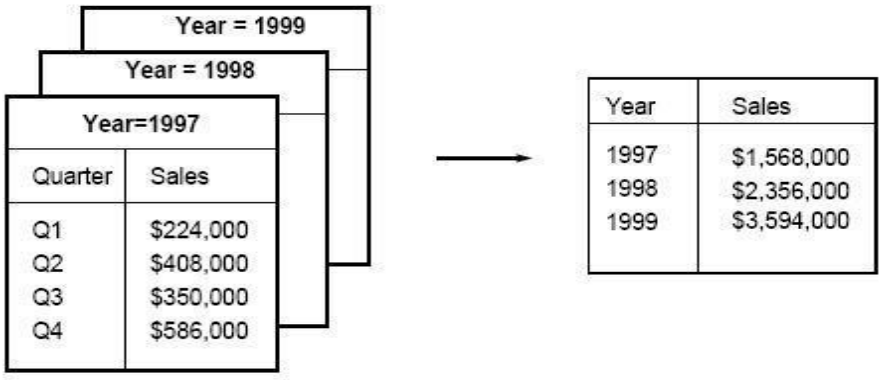
Data cube aggregation: Reduce the data to the concept level needed in the analysis. Queries regarding aggregated information should be answered using data cube when possible. Data cubes store multidimensional aggregated information. The following figure shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each branch.



Each cells holds an aggregate data value, corresponding to the data point in multidimensional space.
Data cubes provide fast access to pre computed, summarized data, thereby benefiting on- line analytical processing as well as data mining.

The cube created at the lowest level of abstraction is referred to as the base cuboid. A cube for the highest level of abstraction is the apex cuboid. The lowest level of a data cube (base cuboid). Data cubes created for varying levels of abstraction are sometimes referred to as cuboids, so that a "data cube" may instead refer to a lattice of cuboids. Each higher level of abstraction further reduces the resulting data size.

The following database consists of sales per quarter for the years 1997-1999.



Suppose, the analyzer interested in the annual sales rather than sales per quarter, the above data can be aggregated so that the resulting data summarizes the total sales per year instead of per quarter. The resulting data in smaller in volume, without loss of information necessary for the analysis task.

Dimensionality Reduction

It reduces the data set size by removing irrelevant attributes. This is a method of attribute subset selection are applied. A heuristic method of attribute of sub set selection is explained here:

Attribute sub selection / Feature selection

Feature selection is a must for any data mining product. That is because, when you build a data mining model, the dataset frequently contains more information than is needed to build the model. For example, a dataset may contain 500 columns that describe characteristics of customers, but perhaps only 50 of those columns are used to build a particular model. If you keep the unneeded columns while building the model, more CPU and memory are required during the training process, and more storage space is required for the completed model.

In which select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features

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

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

Step-wise backward elimination: The procedure starts with the full set of **attributes.** At each step, it removes the worst attribute remaining in the set.

Combination forward selection and backward elimination: The step-wise **forward** selection and backward elimination methods can be combined, where at each step one selects the best attribute and removes the worst from among the remaining attributes.

Decision tree induction: Decision tree induction constructs a flow-chart-like structure where each internal (non-leaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the “best" attribute to partition the data into individual classes. When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

Forward Selection

Initial attribute set:
{A1, A2, A3, A4, A5, A6}

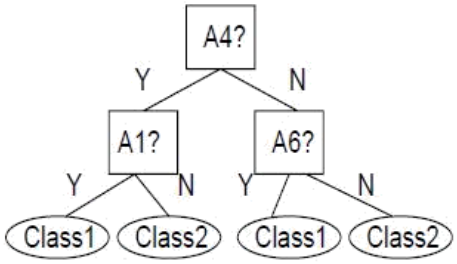
Initial reduced set:
{}
-> {A1}
--> {A1, A4}
---> Reduced attribute set:
{A1, A4, A6}

Backward Elimination

Initial attribute set:
{A1, A2, A3, A4, A5, A6}
-> {A1, A3, A4, A5, A6}
--> {A1, A4, A5, A6}
---> Reduced attribute set:
{A1, A4, A6}

Decision Tree Induction

Initial attribute set:
{A1, A2, A3, A4, A5, A6}



---> Reduced attribute set:
{A1, A4, A6}

Greedy (heuristic) methods for attribute subset selection.

Wrapper approach/Filter approach:

The mining algorithm itself is used to determine the attribute sub set, then it is called wrapper approach or filter approach. Wrapper approach leads to greater accuracy since it optimizes the evaluation measure of the algorithm while removing attributes.

Data compression

In data compression, data encoding or 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 loss of information, the data compression technique used is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data compression technique is called lossy. Effective methods of lossy data compression:

- **Wavelet transforms**
- **Principal components analysis**.

Wavelet compression is a form of data compression well suited for image compression. The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector D, transforms it to a numerically different vector, D0, of wavelet coefficients.

The general algorithm for a discrete wavelet transform is as follows.

The length, L, of the input data vector must be an integer power of two. This condition can be met by padding the data vector with zeros, as necessary.

Each transform involves applying two functions:

- data smoothing
- calculating weighted difference

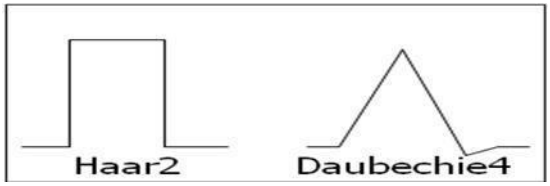
The two functions are applied to pairs of the input data, resulting in two sets of data of length L/2.

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

A selection of values from the data sets obtained in the above iterations are designated the wavelet coefficients of the transformed data.

If wavelet coefficients are larger than some user-specified threshold then it can be retained. The remaining coefficients are set to 0.

Haar2 and Daubechie4 are two popular wavelet transforms.

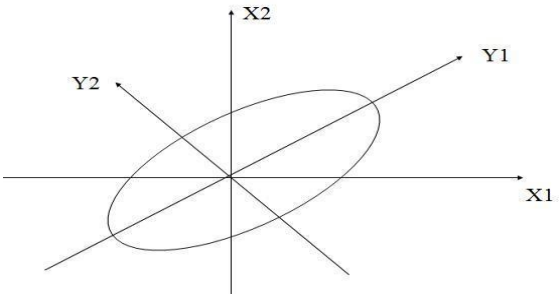


Principal Component Analysis (PCA)
-also called as Karhunen-Loeve (K-L)
method Procedure

- Given *N* data vectors from *k*-dimensions, find *c* <= *k* orthogonal vectors that can be best used to represent data
  - – The original data set is reduced (projected) to one consisting of N data vectors on c principal components (reduced dimensions)

  Each data vector is a linear combination of the *c* principal component vectors
  Works for ordered and unordered attributes
  Used when the number of dimensions is large

The principal components (new set of axes) give important information about variance. Using the strongest components one can reconstruct a good approximation of the original signal.



Numerosity Reduction

Data volume can be reduced by choosing alternative smaller forms of data. This tech. can be

- Parametric method
- Non parametric method

Parametric: Assume the data fits some model, then estimate model parameters, and store only the parameters, instead of actual data.

Non parametric: In which histogram, clustering and sampling is used to store reduced form of data.

Numerosity reduction techniques: 1

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 using

  Y = α + β X, where α, β are coefficients

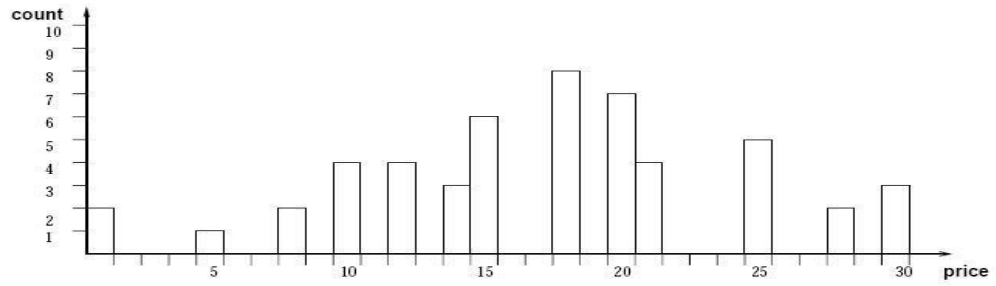
  Multiple regression: *Y = b0 + b1 X1 + b2 X2.*
  - Many nonlinear functions can be transformed into the above.

  Log-linear model: The multi-way table of joint probabilities is approximated by a product of lower-order tables.

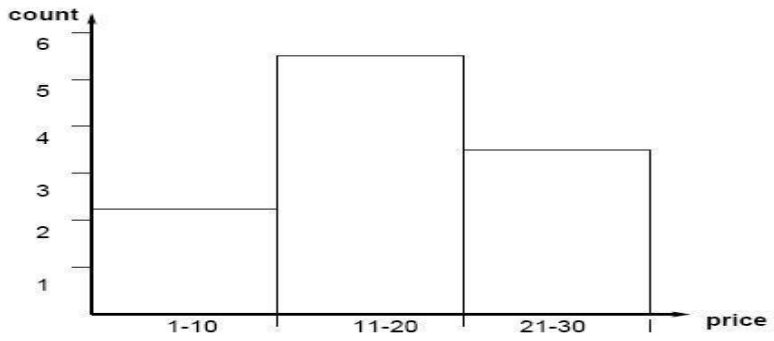
  Probability: *p(a, b, c, d) = □ab □ac□ad □bcd*

2 Histogram

- Divide data into buckets and store average (sum) for each bucket
- A bucket represents an attribute-value/frequency pair
- It can be constructed optimally in one dimension using dynamic programming
- It divides up the range of possible values in a data set into classes or groups. For each group, a rectangle (bucket) is constructed with a base length equal to the range of values in that specific group, and an area proportional to the number of observations falling into that group.
- The buckets are displayed in a horizontal axis while height of a bucket represents the average frequency of the values.



Example:
The following data are a list of prices of commonly sold items. 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. Draw histogram plot for price where each bucket should have equi-width of 10

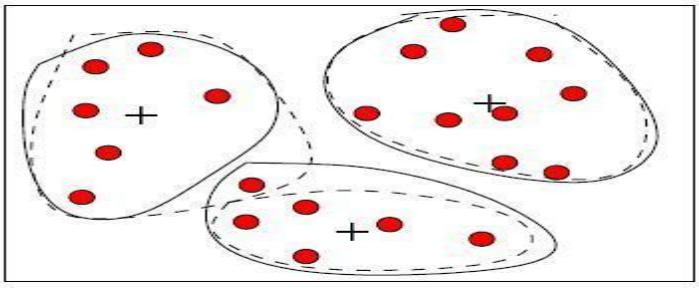


The buckets can be determined based on the following partitioning rules, including the following.

Equi-width: histogram with bars having the same width
Equi-depth: histogram with bars having the same height
3. V-Optimal: histogram with least variance ($count_b * value_b$)
4. MaxDiff: bucket boundaries defined by user specified threshold

V-Optimal and MaxDiff histograms tend to be the most accurate and practical. Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed, and uniform data.

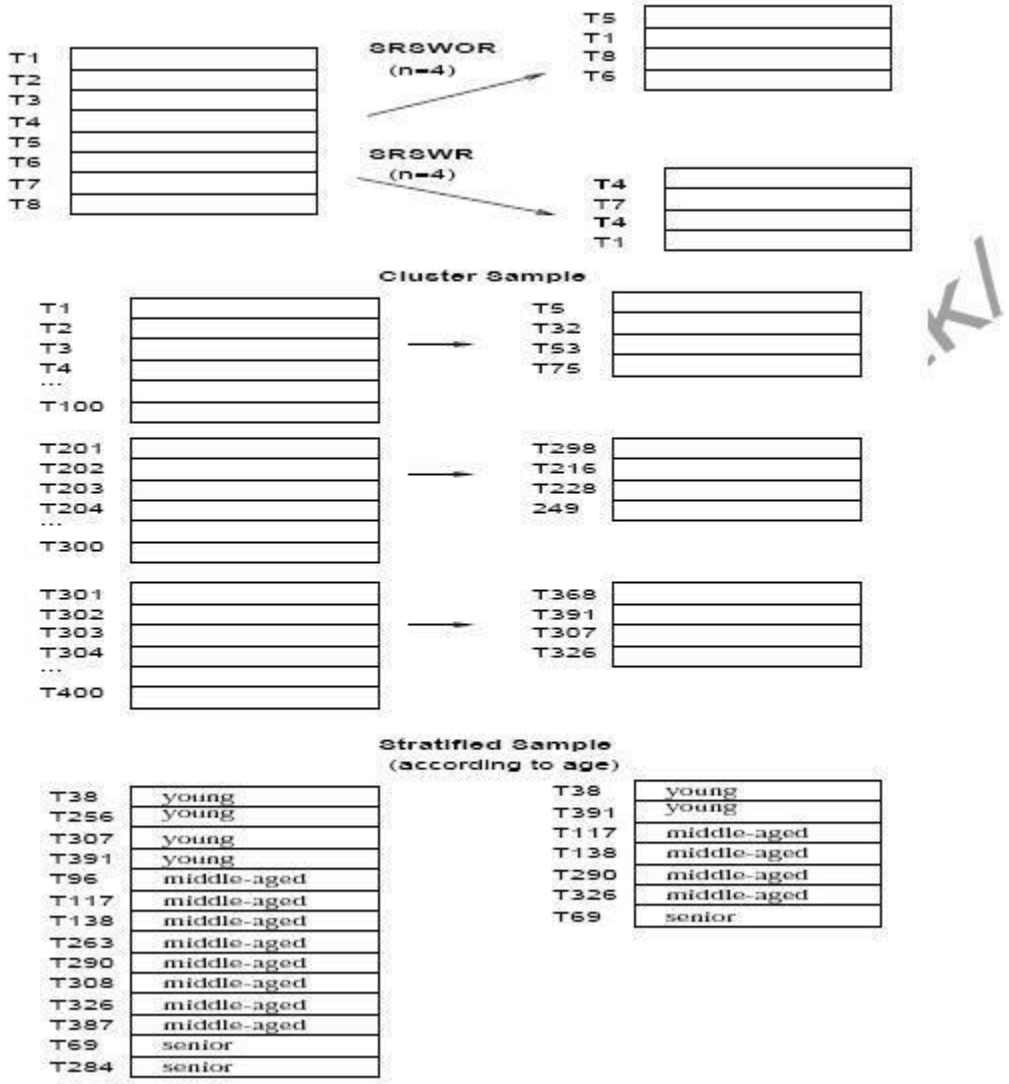
Clustering techniques consider data tuples as objects. They partition the **objects** into groups or clusters, so that objects within a cluster are “similar" to one another and “dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how “close" the objects are in space, based on a distance function.



Quality of clusters measured by their diameter (max distance between any two objects in the cluster) or centroid distance (avg. distance of each cluster object from its centroid)

Sampling

Sampling can be used as a data reduction technique since it allows a large data set to be represented by a much smaller random sample (or subset) of the data. Suppose that a large data set, D, contains N tuples. Let's have a look at some possible samples for D.



Simple random sample without replacement (SRSWOR) of size n: This is created by drawing n of the
N tuples from D (n < N), where the probably of drawing any tuple in D is 1=N, i.e., all tuples are equally likely.

Simple random sample with replacement (SRSWR) of size n: 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 a SRS of m clusters can be obtained, where m < 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.

Stratified sample: If D is divided into mutually disjoint parts called “strata", a stratified sample of D is generated by obtaining a SRS at each stratum. This helps to ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

Advantages of sampling

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

When applied to data reduction, sampling is most commonly used to estimate the answer to an aggregate query.

Discretization and concept hierarchies

Discretization:

Discretization techniques can be used to reduce the number of values for a given continuous attribute, by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values.

Concept Hierarchy

A concept hierarchy for a given numeric attribute defines a Discretization of the attribute. Concept hierarchies can be used to reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior).

Discretization and Concept hierarchy for numerical data:

Three types of attributes:
Nominal — values from an unordered set, e.g., color, profession
Ordinal — values from an ordered set, e.g., military or academic rank Continuous — real numbers, e.g., integer or real numbers

There are five methods for numeric concept hierarchy generation. These include:

binning,
histogram analysis,
clustering analysis,
entropy-based Discretization, and
data segmentation by "natural partitioning".

An information-based measure called "entropy" can be used to recursively partition the values of a numeric attribute A, resulting in a hierarchical Discretization.
Procedure:

- Given a set of samples S, if S is partitioned into two intervals S_1 and S_2 using boundary T, the information gain after partitioning is

$$I(S,T) = \frac{|S_1|}{|S|} Entropy(S_1) + \frac{|S_2|}{|S|} Entropy(S_2)$$

- Entropy is calculated based on class distribution of the samples in the set. Given m classes, the entropy of S_1 is

$$Entropy(S_1) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

where p_i is the probability of class i in S_1

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization
- The process is recursively applied to partitions obtained until some stopping criterion is met
- Such a boundary may reduce data size and improve classification accuracy

5 Segmentation by Natural Partitioning

- A simply 3-4-5 rule can be used to segment numeric data into relatively uniform, "natural" intervals.
  - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
  - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
  - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals

Example:
Suppose that profits at different branches of a company for the year 1997 cover a wide range, from -$351,976.00 to $4,700,896.50. A user wishes to have a concept hierarchy for profit automatically generated

Suppose that the data within the 5%-tile and 95%-tile are between -$159,876 and $1,838,761. The results of applying the 3-4-5 rule are shown in following figure

Step 1: Based on the above information, the minimum and maximum values are: MIN = -$351, 976.00, and MAX = $4, 700, 896.50. The low (5%-tile) and high (95%-tile) values to be considered for the top or first level of segmentation are: LOW = -$159, 876, and HIGH = $1, 838,761.

Step 2: Given LOW and HIGH, the most significant digit is at the million dollar digit position (i.e., msd = 1,000,000). Rounding LOW down to the million dollar digit, we get LOW' = -$1; 000; 000; and rounding HIGH up to the million dollar digit, we get HIGH' = +$2; 000; 000.

Step 3: Since this interval ranges over 3 distinct values at the most significant digit, i.e., (2; 000; 000-(-1, 000; 000))/1, 000, 000 = 3, the segment is partitioned into 3 equi-width sub segments according to the 3-4-5 rule: (-$1,000,000 - $0], ($0 - $1,000,000], and ($1,000,000 - $2,000,000]. This represents the top tier of the hierarchy.

Step 4: We now examine the MIN and MAX values to see how they "fit" into the first level partitions. Since the first interval, (-$1, 000, 000 - $0] covers the MIN value, i.e., LOW' < MIN, we can adjust the left boundary of this interval to make the interval smaller. The most significant digit of MIN is the hundred thousand digit position. Rounding MIN down to this position, we get MIN0' = -$400, 000.

Therefore, the first interval is redefined as (-$400,000 - 0]. Since the last interval, ($1,000,000-$2,000,000] does not cover the MAX value, i.e., MAX > HIGH', we need to create a new interval to cover it. Rounding up MAX at its most significant digit position, the new interval is ($2,000,000 - $5,000,000]. Hence, the top most level of the hierarchy contains four partitions, (-$400,000 - $0], ($0 - $1,000,000], ($1,000,000 - $2,000,000], and ($2,000,000 - $5,000,000].

Step 5: Recursively, each interval can be further partitioned according to the 3-4-5 rule to form the next lower level of the hierarchy:

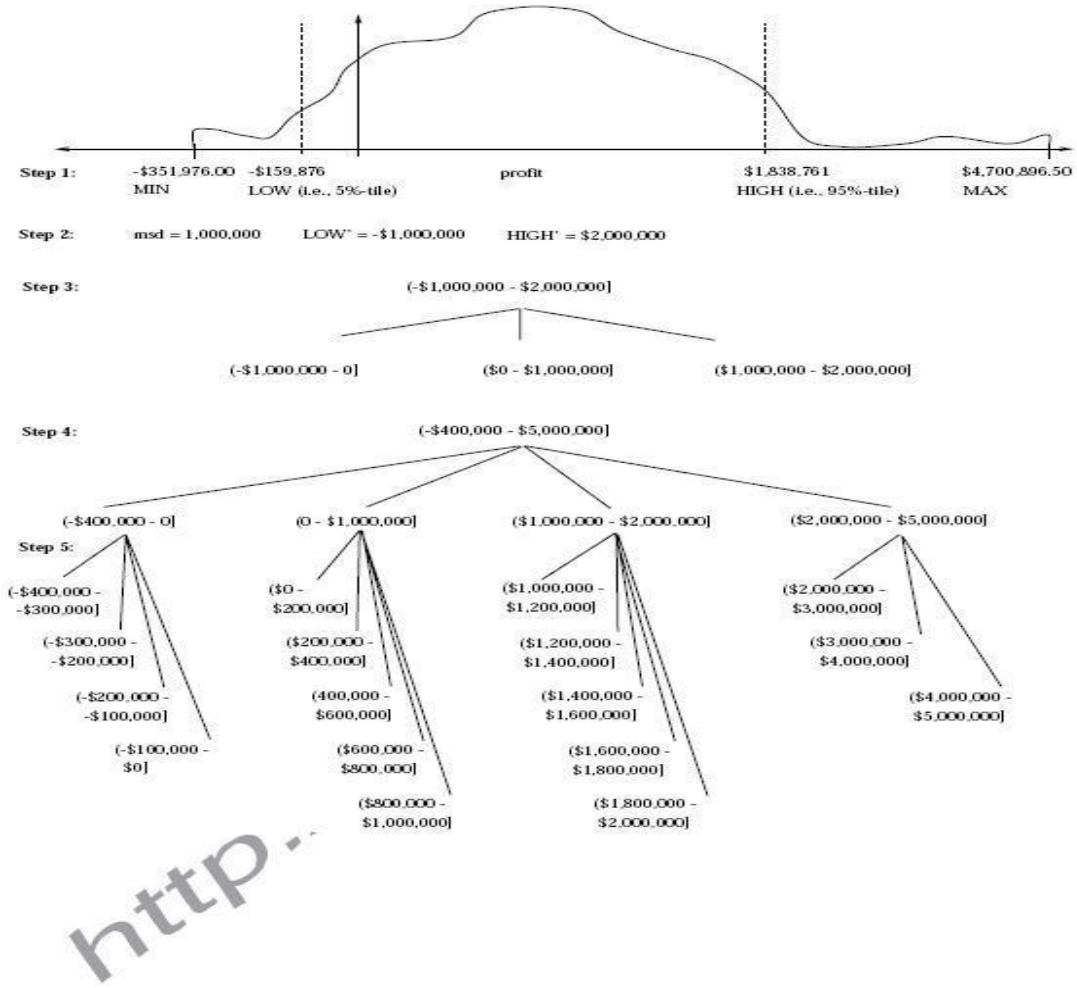
The first interval (-$400,000 - $0] is partitioned into 4 sub-intervals: (-$400,000 - -$300,000], (-$300,000 - -$200,000], (-$200,000 - -$100,000], and (-$100,000 - $0].

The second interval, ($0- $1,000,000], is partitioned into 5 sub-intervals: ($0 - $200,000], ($200,000 - $400,000], ($400,000 - $600,000], ($600,000 - $800,000], and ($800,000 -$1,000,000].

The third interval, ($1,000,000 - $2,000,000], is partitioned into 5 sub-intervals: ($1,000,000 - $1,200,000], ($1,200,000 - $1,400,000], ($1,400,000 - $1,600,000], ($1,600,000 - $1,800,000], and ($1,800,000 - $2,000,000].

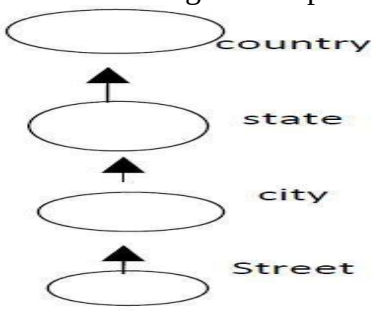
The last interval, ($2,000,000 - $5,000,000], is partitioned into 3 sub-intervals:

($2,000,000 - $3,000,000], ($3,000,000 - $4,000,000], and ($4,000,000 - $5,000,000].



Concept hierarchy generation for category data

A concept hierarchy defines a sequence of mappings from set of low-level concepts to higher-level, more general concepts.
It organizes the values of attributes or dimension into gradual levels of abstraction. They are useful in mining at multiple levels of abstraction



UNIT II

What is Data Warehouse?

Data Warehouse Introduction

A data warehouse is a collection of data marts representing historical data from different operations in the company. This data is stored in a structure optimized for querying and data analysis as a data warehouse. Table design, dimensions and organization should be consistent throughout a data warehouse so that reports or queries across the data warehouse are consistent. A data warehouse can also be viewed as a database for historical data from different functions within a company.

The term Data Warehouse was coined by Bill Inmon in 1990, which he defined in the following way: "A warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process". He defined the terms in the sentence as follows:

Subject Oriented: Data that gives information about a particular subject instead of about a company's ongoing operations.

Integrated: Data that is gathered into the data warehouse from a variety of sources and merged into a coherent whole.

Time-variant: All data in the data warehouse is identified with a particular time period.

Non-volatile: Data is stable in a data warehouse. More data is added but data is never removed.

This enables management to gain a consistent picture of the business. It is a single, complete and consistent store of data obtained from a variety of different sources made available to end users in what they can understand and

use in a business context. It can be

- Used for decision Support
- Used to manage and control business
- Used by managers and end-users to understand the business and make judgments

Data Warehousing is an architectural construct of information systems that provides users with current and historical decision support information that is hard to access or present in traditional operational data stores.

Other important terminology

Enterprise Data warehouse: It collects all information about subjects (*customers, products, sales, assets, personnel*) that span the entire organization

Data Mart: Departmental subsets that focus on selected subjects. A data mart is a segment of a data warehouse that can provide data for reporting and analysis on a section, unit, department or operation in the company, e.g. sales, payroll, production. Data marts are sometimes complete individual data warehouses which are usually smaller than the corporate data warehouse.

*Decision Support System (DSS):*Information technology to help the knowledge worker (executive, manager, and analyst) makes faster & better decisions

Drill-down: Traversing the summarization levels from highly summarized data to the underlying current or old detail

Metadata: Data about data. Containing location and description of warehouse system components: names, definition, structure…

<u>Benefits of data warehousing</u>

- Data warehouses are designed to perform well with aggregate queries running on large amounts of data.
- The structure of data warehouses is easier for end users to navigate, understand and query against unlike the relational databases primarily designed to handle lots of transactions.
- Data warehouses enable queries that cut across different segments of a company's operation. E.g. production data could be compared against inventory data even if they were originally stored in different databases with different structures.
- Queries that would be complex in very normalized databases could be easier to build and maintain in data warehouses, decreasing the workload on transaction systems.
- Data warehousing is an efficient way to manage and report on data that is from a variety of sources, non uniform and scattered throughout a company.
- Data warehousing is an efficient way to manage demand for lots of information from lots of users.
- Data warehousing provides the capability to analyze large amounts of historical data for nuggets of wisdom that can provide an organization with competitive advantage.

Operational and informational Data

• Operational Data:
  - Focusing on transactional function such as bank card withdrawals and deposits
  - Detailed
  - Updateable
  - Reflects current data

• Informational Data:
  - Focusing on providing answers to problems posed by decision makers
  - Summarized
  - Non updateable

These differences between the informational and operational databases are
These differences between the informational and operational databases are
summarized in the following table.

	Operational data	Informational data
Data content	Current values	Summarized, archived, derived
Data organization	By application	By subject
Data stability	Dynamic	Static until refreshed
Data structure	Optimized for transactions	Optimized for complex queries
Access frequency	High	Medium to low
Access type	Read/update/delete Field-by-field	Read/aggregate Added to
Usage	Predictable Repetitive	Ad hoc, unstructured Heuristic
Response time	Subsecond (<1 s) to 2–3 s	Several seconds to minutes

Data Warehouse Characteristics

• A data warehouse can be viewed as an information system with the following attributes: – It is a database designed for analytical tasks
– It‘s content is periodically updated
– It contains current and historical data to provide a historical perspective of information

Operational data store (ODS)

ODS is an architecture concept to support day-to-day operational decision support and contains current value data propagated from operational applications
ODS is subject-oriented, similar to a classic definition of a Data warehouse • ODS is integrated

However:

ODS	DATA WAREHOUSE

Volatile	Non volatile
Very current data	Current and historical data
Detailed data	Pre calculated summaries

3.1.1 Differences between Operational Database Systems and Data Warehouses

Features of OLTP and OLAP

The major distinguishing features between OLTP and OLAP are summarized as follows.

1. Users and system orientation: An OLTP system is customer-oriented and is used for transaction and query processing by clerks, clients, and information technology professionals. An OLAP system is market-oriented and is used for data analysis by knowledge workers, including managers, executives, and analysts.

Data contents: An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. An OLAP system manages large amounts of historical data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity. These features make the data easier for use in informed decision making.

Database design: An OLTP system usually adopts an entity-relationship (ER) data model and an application oriented database design. An OLAP system typically adopts either a star or snowflake model and a subject-oriented database design.

View: An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historical data or data in different organizations. In contrast, an OLAP system often spans multiple versions of a database schema. OLAP systems also deal with information that originates from different organizations, integrating information from many data stores. Because of their huge volume, OLAP data are stored on multiple storage media.

Access patterns: The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery

mechanisms. However, accesses to OLAP systems are mostly read-only operations although many could be complex queries.

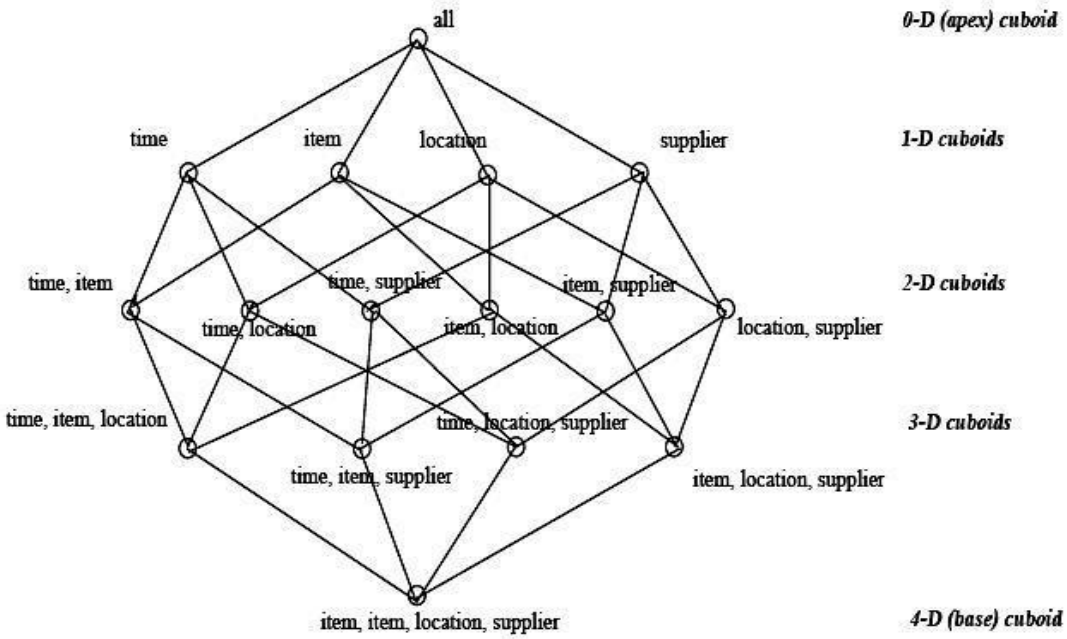
Comparison between OLTP and OLAP systems.

Feature	OLTP	OLAP
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, anal
Function	day-to-day operations	long term informational requirements, decision support
DB design	E-R based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
# of records accessed	tens	millions
# of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

A Multidimensional Data Model.

The most popular data model for data warehouses is a multidimensional model. This model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema. Let's have a look at each of these schema types.

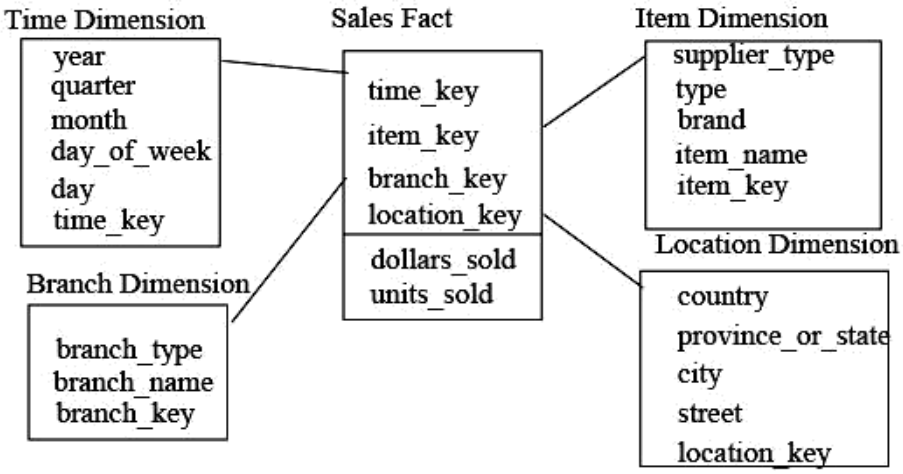
From Tables and Spreadsheets to Data Cubes



Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Databases

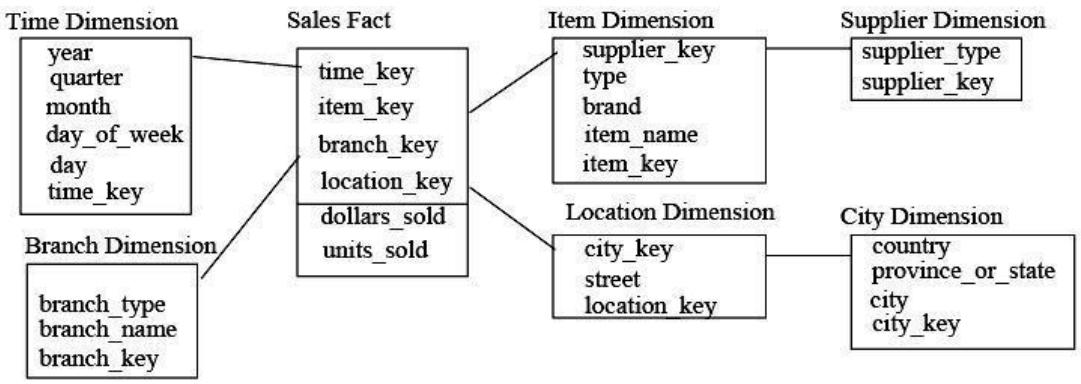
Star schema: The star schema is a modeling paradigm in which the data warehouse contains (1) a large central table (fact table), and (2) a set of smaller attendant tables (dimension tables), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.

Figure Star schema of a data warehouse for sales.



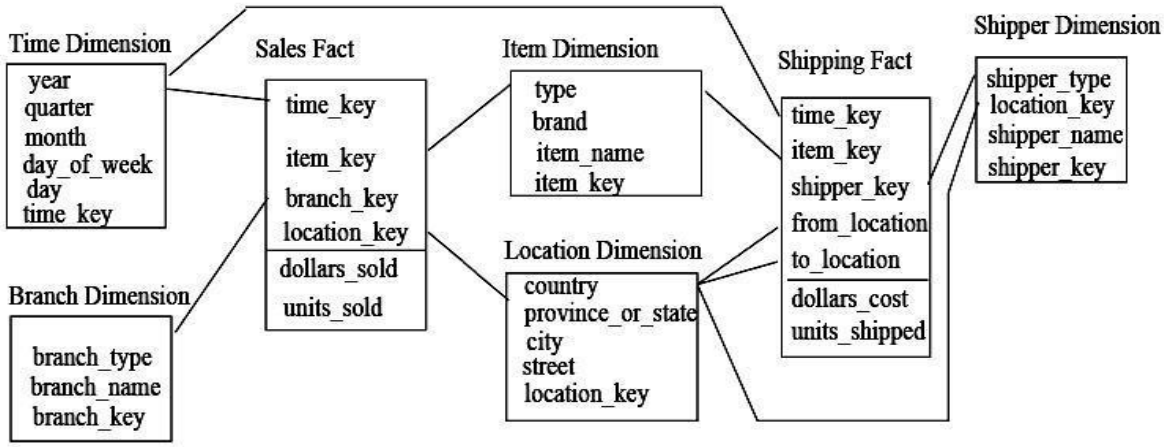
Snowflake schema: The snowflake schema is a variant of the star schema model, where some dimension tables are normalized, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake. The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form. Such a table is easy to maintain and also saves storage space because a large dimension table can be extremely large when the dimensional structure is included as columns.

Figure Snowflake schema of a data warehouse for sales.



Fact constellation: Sophisticated applications may require multiple fact tables to share dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

Figure Fact constellation schema of a data warehouse for sales and shipping.



Example for Defining Star, Snowflake, and Fact Constellation Schemas

A Data Mining Query Language, DMQL: Language Primitives

Cube Definition (Fact Table)

define cube <cube _name> [<dimension_ list>]: <measure_ list>

Dimension Definition (Dimension Table)

define dimension <dimension_name> as (<attribute_or_subdimension_list>)

Special Case (Shared Dimension Tables)

First time as "cube definition"

define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining a Star Schema in DMQL

define cube sales_star [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state, country)

Defining a Snowflake Schema in DMQL
define cube sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city(city_key, province_or_state, country))

Defining a Fact Constellation in DMQL define
cube sales [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars), units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state,
country) define cube shipping [time, item, shipper, from_location, to_location]:
dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales

Measures: Three Categories

Measure: a function evaluated on aggregated data corresponding to given dimension-value pairs.

Measures can be:

distributive: if the measure can be calculated in a distributive manner.
E.g., count(), sum(), min(), max().

algebraic: if it can be computed from arguments obtained by applying distributive aggregate functions.
E.g., avg()=sum()/count(), min_N(), standard_deviation().

holistic: if it is not algebraic.
E.g., median(), mode(), rank().

A Concept Hierarchy

A Concept hierarchy defines a sequence of mappings from a set of low level Concepts to higher level, more general Concepts. Concept hierarchies allow data to be handled at varying levels of abstraction

3.2.6 OLAP operations on multidimensional data.

Roll-up: The roll-up operation performs aggregation on a data cube, either by climbing-up a concept hierarchy for a dimension or by dimension reduction. Figure shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for location. This hierarchy was defined as the total order street < city < province or state <country.

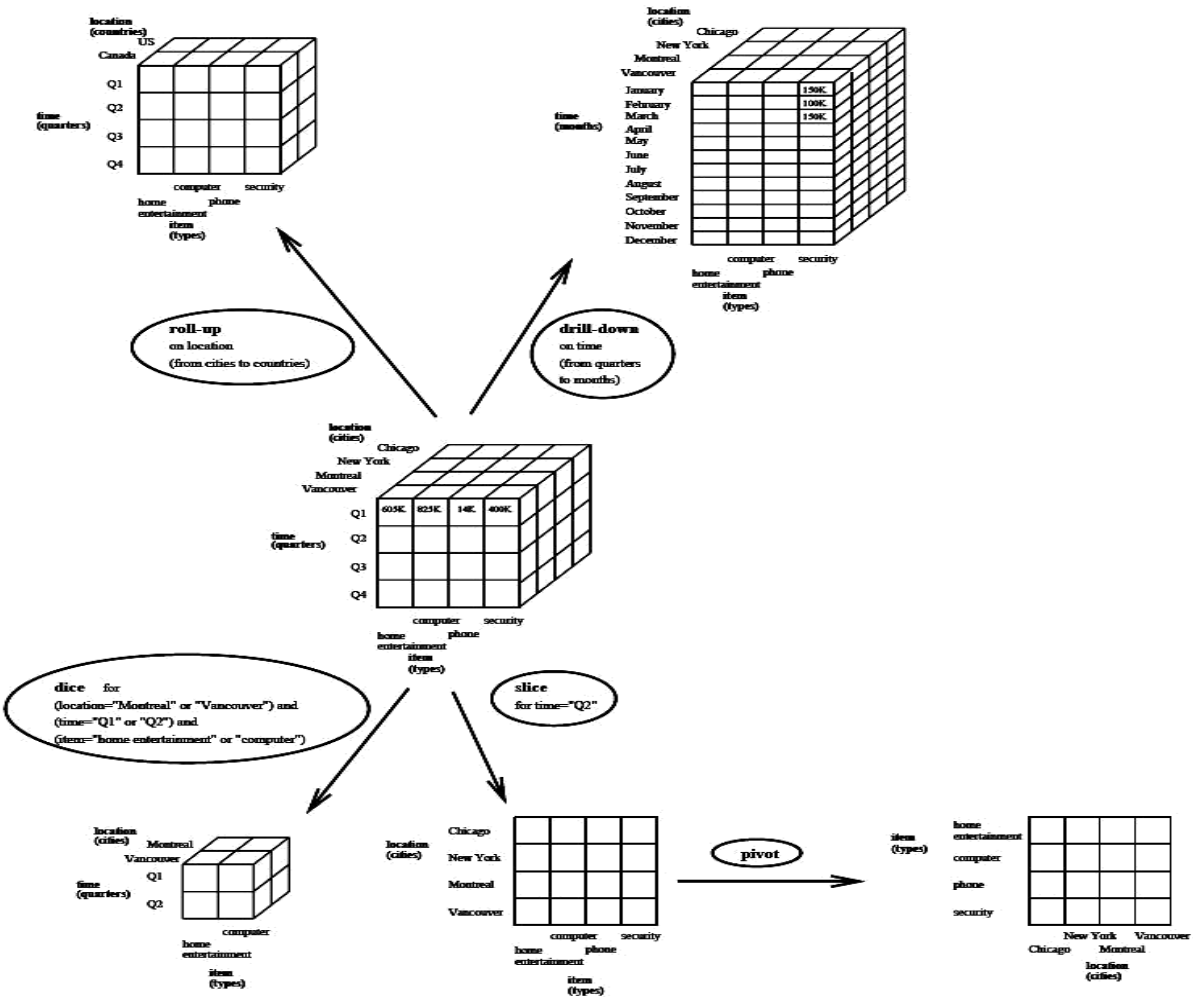
Drill-down: Drill-down is the reverse of roll-up. It navigates from less detailed data to more detailed data. Drill-down can be realized by either stepping-down a concept hierarchy for a dimension or introducing additional dimensions. Figure shows the result of a drill-down operation performed on the central cube by stepping down a concept hierarchy for time defined as day < month < quarter < year. Drill-down occurs by descending the time hierarchy from the level of quarter to the more detailed level of month.

Slice and dice: The slice operation performs a selection on one dimension of the given cube, resulting in a sub cube. Figure shows a slice operation where the sales data are selected from the central cube for the dimension time using the criteria

time=”Q2". The dice operation defines a sub cube by performing a selection on two or more dimensions.

4. **Pivot (rotate):** Pivot is a visualization operation which rotates the data axes in view in order to provide an alternative presentation of the data. Figure shows a pivot operation where the item and location axes in a 2-D slice are rotated.

Figure : Examples of typical OLAP operations on multidimensional data.



Data warehouse architecture

Steps for the Design and Construction of Data Warehouse

This subsection presents a business analysis framework for data warehouse design. The basic steps involved in the design process are also described.

The Design of a Data Warehouse: A Business Analysis Framework

Four different views regarding the design of a data warehouse must be considered: the top-down view, the data source view, the data warehouse view, the business query view.

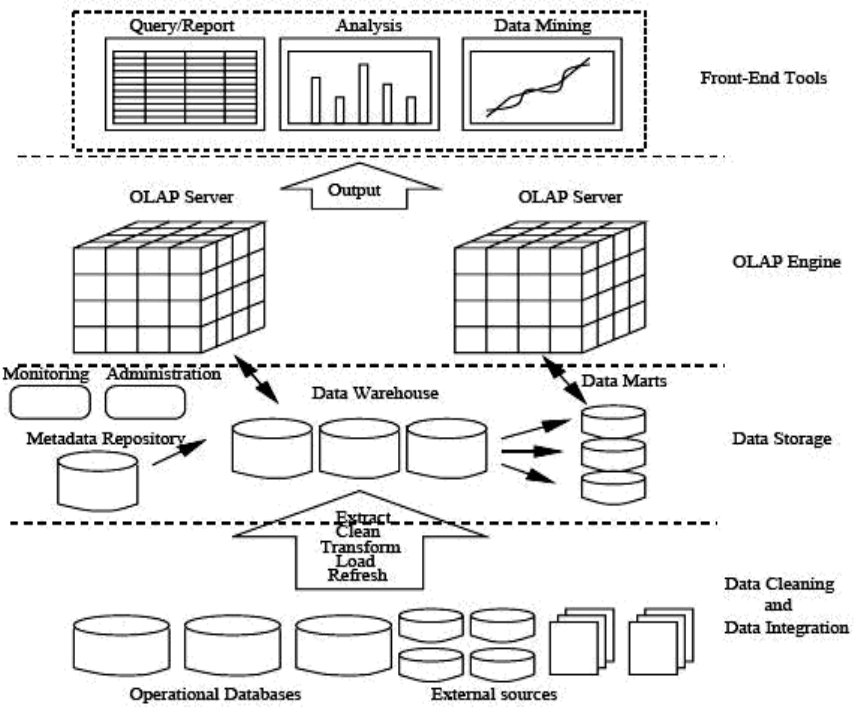
The top-down view allows the selection of relevant information necessary for the data warehouse.

The data source view exposes the information being captured, stored and managed by operational systems.

The data warehouse view includes fact tables and dimension tables

Finally the business query view is the Perspective of data in the data warehouse from the viewpoint of the end user.

Three-tier Data warehouse architecture



The bottom tier is ware-house database server which is almost always a relational database system. The middle tier is an OLAP server which is typically implemented using either (1) a Relational OLAP (ROLAP) model, (2) a Multidimensional OLAP (MOLAP) model. The top tier is a client, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

From the architecture point of view, there are three data warehouse models: the enterprise warehouse, the data mart, and the virtual warehouse.

Enterprise warehouse: An enterprise warehouse collects all of the information about subjects spanning the entire organization. It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope. It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.

Data mart: A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is connected to specific, selected subjects. For example, a marketing data mart may connect its subjects to customer, item, and sales. The data contained in data marts tend to be summarized. Depending on the source of data, data marts can be categorized into the following two classes:

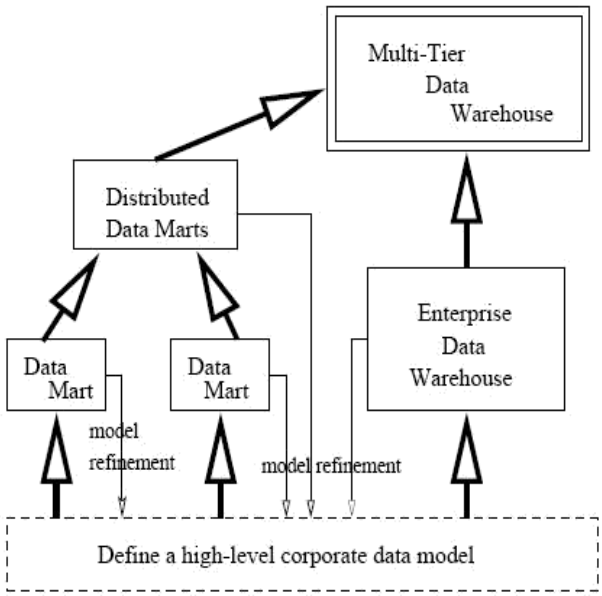
(i).Independent data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area.

(ii).Dependent data marts are sourced directly from enterprise data warehouses.

Virtual warehouse: A virtual warehouse is a set of views over operational databases. For efficient query processing, only some of the possible summary

views may be materialized. A virtual warehouse is easy to build but requires excess capacity on operational database servers.

Figure: A recommended approach for data warehouse development.



Data warehouse Back-End Tools and Utilities

The ETL (Extract Transformation Load) process

In this section we will discussed about the 4 major process of the data warehouse. They are extract (data from the operational systems and bring it to the data warehouse), transform (the data into internal format and structure of the data warehouse), cleanse (to make sure it is of sufficient quality to be used for decision making) and load (cleanse data is put into the data warehouse).

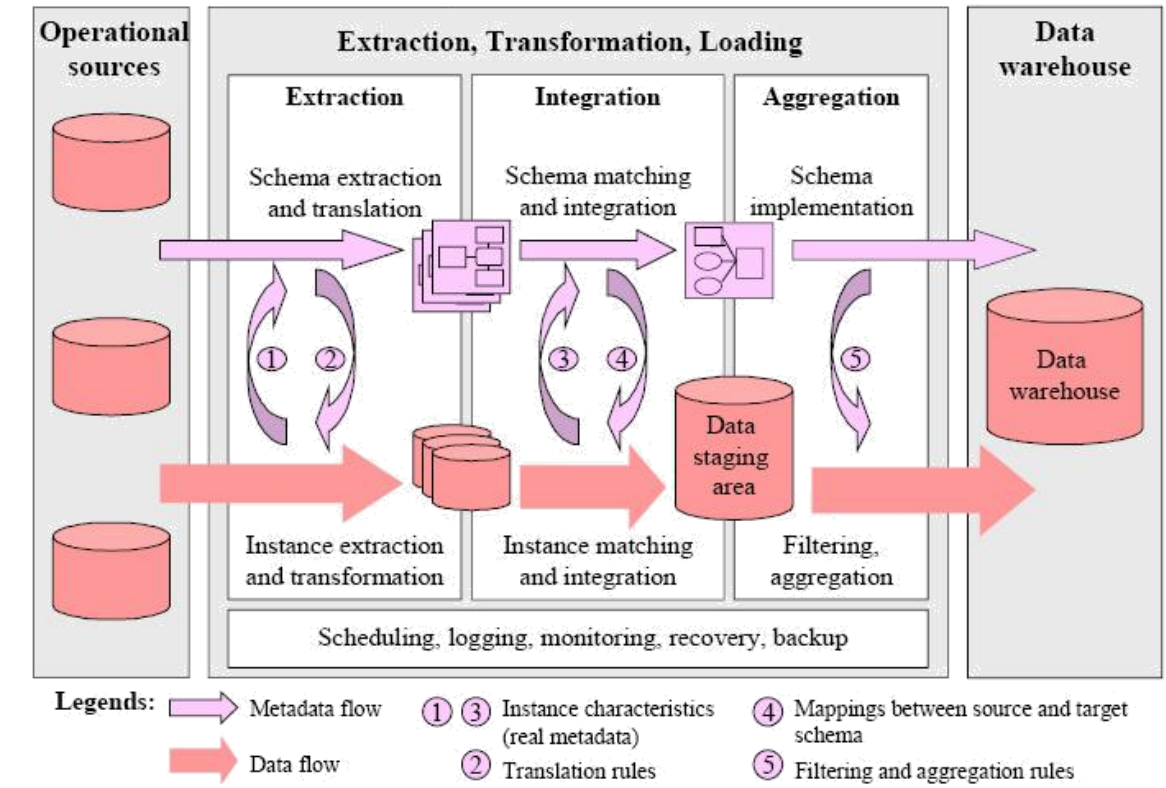


Figure 1. Steps of building a data warehouse: the ETL process

The four processes from extraction through loading often referred collectively as **Data Staging**.

EXTRACT

Some of the data elements in the operational database can be reasonably be expected to be useful in the decision making, but others are of less value for that purpose. For this reason, it is necessary to extract the relevant data from the operational database before bringing into the data warehouse. Many commercial tools are available to help with the extraction process. **Data Junction** is one of the commercial products. The user of one of these tools typically has an easy-to-use windowed interface by which to specify the following:

Which files and tables are to be accessed in the source database?
Which fields are to be extracted from them? This is often done internally by SQL Select statement.
What are those to be called in the resulting database?

What is the target machine and database format of the output?
On what schedule should the extraction process be repeated?

TRANSFORM

The operational databases developed can be based on any set of priorities, which keeps changing with the requirements. Therefore those who develop data warehouse based on these databases are typically faced with inconsistency among their data sources. Transformation process deals with rectifying any inconsistency (if any).

One of the most common transformation issues is 'Attribute Naming Inconsistency'. It is common for the given data element to be referred to by different data names in different databases. Employee Name may be EMP_NAME in one database, ENAME in the other. Thus one set of Data Names are picked and used consistently in the data warehouse. Once all the data elements have right names, they must be converted to common formats. The conversion may encompass the following:

Characters must be converted ASCII to EBCDIC or vise versa.
Mixed Text may be converted to all uppercase for consistency.
Numerical data must be converted in to a common format.
Data Format has to be standardized.
Measurement may have to convert. (Rs/ $)
Coded data (Male/ Female, M/F) must be converted into a common format.

All these transformation activities are automated and many commercial products are available to perform the tasks. **DataMAPPER** from Applied Database Technologies is one such comprehensive tool.

CLEANSING

Information quality is the key consideration in determining the value of the information. The developer of the data warehouse is not usually in a position to change the quality of its underlying historic data, though a data warehousing project can put spotlight on the data quality issues and lead to improvements for the future. It is, therefore, usually necessary to go through the data entered into the data warehouse and make it as error free as possible. This process is known as **Data Cleansing**.

Data Cleansing must deal with many types of possible errors. These include missing data and incorrect data at one source; inconsistent data and conflicting data when two

or more source are involved. There are several algorithms followed to clean the data, which will be discussed in the coming lecture notes.

LOADING

Loading often implies physical movement of the data from the computer(s) storing the source database(s) to that which will store the data warehouse database, assuming it is different. This takes place immediately after the extraction phase. The most common channel for data movement is a high-speed communication link. Ex: Oracle Warehouse Builder is the API from Oracle, which provides the features to perform the ETL task on Oracle Data Warehouse.

Data cleaning problems

This section classifies the major data quality problems to be solved by data cleaning and data transformation. As we will see, these problems are closely related and should thus be treated in a uniform way. Data transformations [26] are needed to support any changes in the structure, representation or content of data. These transformations become necessary in many situations, e.g., to deal with schema evolution, migrating a legacy system to a new information system, or when multiple data sources are to be integrated. As shown in Fig. 2 we roughly distinguish between single-source and multi-source problems and between schema- and instance-related problems. Schema-level problems of course are also reflected in the instances; they can be addressed at the schema level by an improved schema design (schema evolution), schema translation and schema integration. Instance-level problems, on the other hand, refer to errors and inconsistencies in the actual data contents which are not visible at the schema level. They are the primary focus of data cleaning. Fig. 2 also indicates some typical problems for the various cases. While not shown in Fig. 2, the single-source problems occur (with increased likelihood) in the multi-source case, too, besides specific multi-source problems.

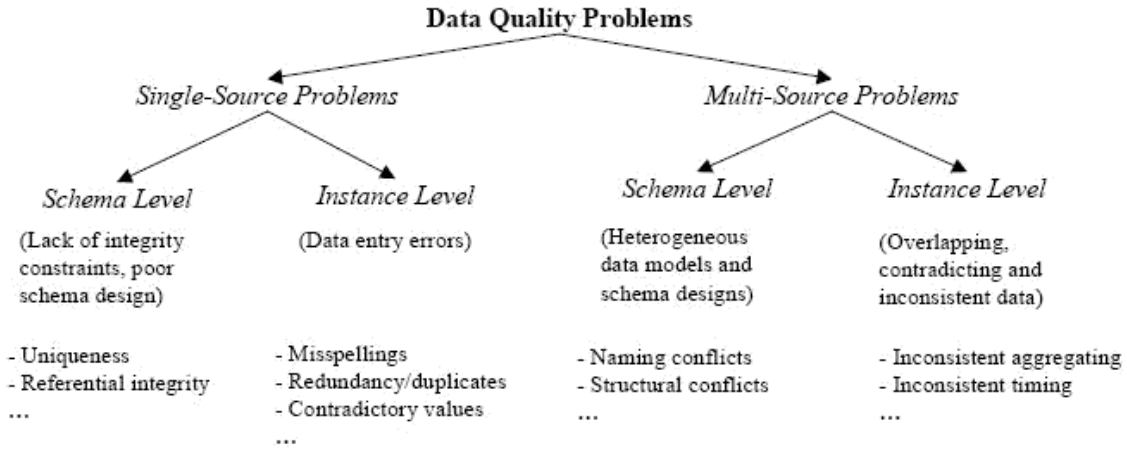


Figure 2. Classification of data quality problems in data sources

Single-source problems

The data quality of a source largely depends on the degree to which it is governed by schema and integrity constraints controlling permissible data values. For sources without schema, such as files, there are few restrictions on what data can be entered and stored, giving rise to a high probability of errors and inconsistencies. Database systems, on the other hand, enforce restrictions of a specific data model (e.g., the relational approach requires simple attribute values, referential integrity, etc.) as well as application-specific integrity constraints. Schema-related data quality problems thus occur because of the lack of appropriate model-specific or application-specific integrity constraints, e.g., due to data model limitations or poor schema design, or because only a few integrity constraints were defined to limit the overhead for integrity control. Instance-specific problems relate to errors and inconsistencies that cannot be prevented at the schema level (e.g., misspellings).

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Illegal values	bdate=30.13.70	values outside of domain range
Record	Violated attribute dependencies	age=22, bdate=12.02.70	age = (current date – birth date) should hold
Record type	Uniqueness violation	emp_1=(name="John Smith", SSN="123456") emp_2=(name="Peter Miller", SSN="123456")	uniqueness for SSN (social security number) violated
Source	Referential integrity violation	emp=(name="John Smith", deptno=127)	referenced department (127) not defined

Table 1. Examples for single-source problems at schema level (violated integrity constraints)

For both schema- and instance-level problems we can differentiate different problem scopes: attribute (field), record, record type and source; examples for the various cases are shown in Tables 1 and 2. Note that uniqueness constraints specified at the schema level do not prevent duplicated instances, e.g., if information on the same real world entity is entered twice with different attribute values (see example in Table 2).

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city="Liipzig"	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience="B"; occupation="DB Prog."	
	Embedded values	name="J. Smith 12.02.70 New York"	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city="Germany"	
Record	Violated attribute dependencies	city="Redmond", zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= "J. Smith", $name_2$="Miller P."	usually in a free-form field
	Duplicated records	emp_1=(name="John Smith",...); emp_2=(name="J. Smith",...)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name="John Smith", bdate=12.02.70); emp_2=(name="John Smith", bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name="John Smith", deptno=17)	referenced department (17) is defined but wrong

Table 2. Examples for single-source problems at instance level

Multi-source problems

The problems present in single sources are aggravated when multiple sources need to be integrated. Each source may contain dirty data and the data in the sources may be represented differently, overlap or contradict. This is because the sources are typically developed, deployed and maintained independently to serve specific needs. This results in a large degree of heterogeneity w.r.t. data management systems, data models, schema designs and the actual data.

At the schema level, data model and schema design differences are to be addressed by the steps of schema translation and schema integration, respectively. The main problems w.r.t. schema design are naming and structural conflicts. Naming conflicts arise when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms). Structural conflicts occur in many variations and refer to different representations of the same object in different sources, e.g., attribute vs. table representation, different component structure, different data types, different integrity constraints, etc. In addition to schema-level

conflicts, many conflicts appear only at the instance level (data conflicts). All problems from the single-source case can occur with different representations in different sources (e.g., duplicated records, contradicting records,…). Furthermore, even when there are the same attribute names and data types, there may be different value representations (e.g., for marital status) or different interpretation of the values (e.g., measurement units Dollar vs. Euro) across sources. Moreover, information in the sources may be provided at different aggregation levels (e.g., sales per product vs. sales per product group) or refer to different points in time (e.g. current sales as of yesterday for source 1 vs. as of last week for source 2).

A main problem for cleaning data from multiple sources is to identify overlapping data, in particular matching records referring to the same real-world entity (e.g., customer). This problem is also referred to as the object identity problem, duplicate elimination or the merge/purge problem. Frequently, the information is only partially redundant and the sources may complement each other by providing additional information about an entity. Thus duplicate information should be purged out and complementing information should be consolidated and merged in order to achieve a consistent view of real world entities.

Customer (source 1)

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Client (source 2)

Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Customers (integrated target with cleaned data)

No	*LName*	*FName*	*Gender*	*Street*	*City*	*State*	*ZIP*	*Phone*	*Fax*	*CID*	*Cno*
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

Figure 3. Examples of multi-source problems at schema and instance level

The two sources in the example of Fig. 3 are both in relational format but exhibit schema and data conflicts. At the schema level, there are name conflicts (synonyms *Customer*/*Client*, *Cid*/*Cno*, *Sex*/*Gender*) and structural conflicts (different representations for names and addresses). At the instance level, we note that there are different gender representations (“0”/”1” vs. “F”/”M”) and presumably a duplicate

record (Kristen Smith). The latter observation also reveals that while *Cid*/*Cno* are both source-specific identifiers, their contents are not comparable between the sources; different numbers (11/493) may refer to the same person while different persons can have the same number (24). Solving these problems requires both schema integration and data cleaning; the third table shows a possible solution. Note that the schema conflicts should be resolved first to allow data cleaning, in particular detection of duplicates based on a uniform representation of names and addresses, and matching of the *Gender*/*Sex* values.

Data cleaning approaches

In general, data cleaning involves several phases

Data analysis: In order to detect which kinds of errors and inconsistencies are to be removed, a detailed

data analysis is required. In addition to a manual inspection of the data or data samples, analysis programs should be used to gain metadata about the data properties and detect data quality problems.

Definition of transformation workflow and mapping rules: Depending on the number of data sources ,their degree of heterogeneity and the "dirtyness" of the data, a large number of data transformation and cleaning steps may have to be executed. Sometime, a schema translation is used to map sources to a common data model; for data warehouses, typically a relational representation is used. Early data cleaning steps can correct single-source instance problems and prepare the data for integration. Later steps deal with schema/data integration and cleaning multi-source instance problems, e.g., duplicates.

For data warehousing, the control and data flow for these transformation and cleaning steps should be specified within a workflow that defines the ETL process (Fig. 1).

The schema-related data transformations as well as the cleaning steps should be specified by a declarative query and mapping language as far as possible, to enable automatic generation of the transformation code. In addition, it should be possible to invoke user-written cleaning code and special purpose tools during a data transformation workflow. The transformation steps may

request user feedback on data instances for which they have no built-in cleaning logic.

Verification: The correctness and effectiveness of a transformation workflow and the transformation definitions should be tested and evaluated, e.g., on a sample or copy of the source data, to improve the definitions if necessary.
Multiple iterations of the analysis, design and verification steps may be needed, e.g., since some errors only become apparent after applying some transformations.

Transformation: Execution of the transformation steps either by running the ETL workflow for loading and refreshing a data warehouse or during answering queries on multiple sources.

Backflow of cleaned data: After (single-source) errors are removed, the cleaned data should also replace the dirty data in the original sources in order to give legacy applications the improved data too and to avoid redoing the cleaning work for future data extractions. For data warehousing, the cleaned data is available from the data staging area (Fig. 1).

Data analysis

Metadata reflected in schemas is typically insufficient to assess the data quality of a source, especially if only a few integrity constraints are enforced. It is thus important to analyse the actual instances to obtain real (reengineered) metadata on data characteristics or unusual value patterns. This metadata helps finding data quality problems. Moreover, it can effectively contribute to identify attribute correspondences between source schemas (schema matching), based on which automatic data transformations can be derived.

There are two related approaches for data analysis, data profiling and data mining. *Data profiling* focuses on the instance analysis of individual attributes. It derives information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, typical string pattern (e.g., for phone numbers), etc., providing an exact view of various quality aspects of the attribute.

Table 3 shows examples of how this metadata can help detecting data quality problems.

Problems	Metadata	Examples/Heuristics
Illegal values	cardinality	e.g., cardinality (gender) > 2 indicates problem
	max, min	max, min should not be outside of permissible range
	variance, deviation	variance, deviation of statistical values should not be higher than threshold
Misspellings	attribute values	sorting on values often brings misspelled values next to correct values
Missing values	null values	percentage/number of null values
	attribute values + default values	presence of default value may indicate real value is missing
Varying value representations	attribute values	comparing attribute value set of a column of one table against that of a column of another table
Duplicates	cardinality + uniqueness	attribute cardinality = # rows should hold
	attribute values	sorting values by number of occurrences; more than 1 occurrence indicates duplicates

Table 3. Examples for the use of reengineered metadata to address data quality problems

Metadata repository

Metadata are data about data. When used in a data warehouse, metadata are the data that define warehouse objects. Metadata are created for the data names and definitions of the given warehouse. Additional metadata are created and captured for time stamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes. A metadata repository should contain:

A description of the structure of the data warehouse. This includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents;

Operational metadata, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails);

the algorithms used for summarization, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports;

The mapping from the operational environment to the data warehouse, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).

Data related to system performance, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles; and

Business metadata, which include business terms and definitions, data ownership information, and charging policies.

Types of OLAP Servers: ROLAP versus MOLAP versus HOLAP

Relational OLAP (ROLAP)

Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware to support missing pieces

Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services

greater scalability

Multidimensional OLAP (MOLAP)

Array-based multidimensional storage engine (sparse matrix techniques)

fast indexing to pre-computed summarized data

Hybrid OLAP (HOLAP)

User flexibility, e.g., low level: relational, high-level: array

Specialized SQL servers

specialized support for SQL queries over star/snowflake schemas

Data Warehouse Implementation

Efficient Computation of Data Cubes

Data cube can be viewed as a lattice of cuboids

The bottom-most cuboid is the base cuboid
The top-most cuboid (apex) contains only one cell
How many cuboids in an n-dimensional cube with L levels?

Materialization of data cube

Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
Selection of which cuboids to materialize
Based on size, sharing, access frequency, etc.

Cube Operation

Cube definition and computation in DMQL

define cube sales[item, city, year]: sum(sales_in_dollars)

compute cube sales

Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

SELECT item, city, year, SUM (amount)

FROM SALES

CUBE BY item, city, year

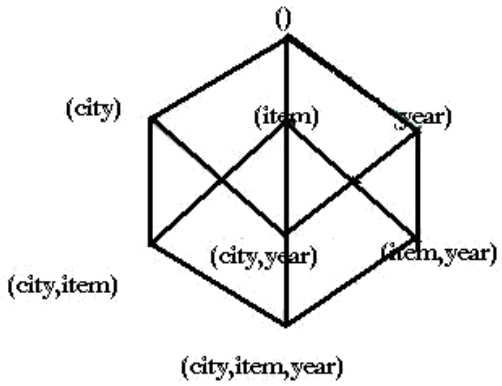
Need compute the following Group-Bys

(date, product, customer),

(date,product),(date, customer), (product, customer),

(date), (product), (customer)

()



Cube Computation: ROLAP-Based Method

Efficient cube computation methods

- ROLAP-based cubing algorithms (Agarwal et al'96)
- Array-based cubing algorithm (Zhao et al'97)
- Bottom-up computation method (Bayer & Ramarkrishnan'99)

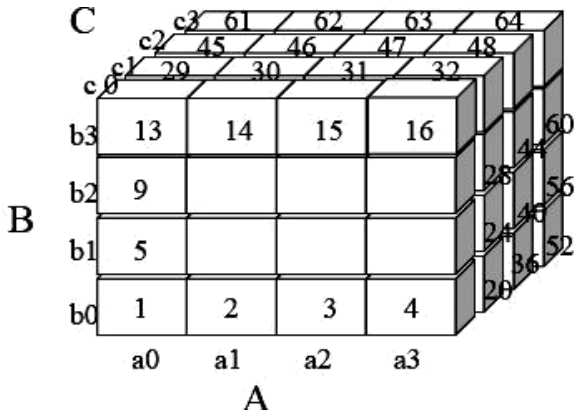
ROLAP-based cubing algorithms

- Sorting, hashing, and grouping operations are applied to the dimension attributes in order to reorder and cluster related tuples
- Grouping is performed on some sub aggregates as a "partial grouping step"
- Aggregates may be computed from previously computed aggregates, rather than from the base fact table

Multi-way Array Aggregation for Cube

Computation

- Partition arrays into chunks (a small sub cube which fits in memory).
- Compressed sparse array addressing: (chunk_id, offset)
- Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.



Indexing OLAP data

The bitmap indexing method is popular in OLAP products because it allows quick searching in data cubes.

The **bitmap index** is an alternative representation of the record ID (RID) list. In the bitmap index for a given attribute, there is a distinct bit vector, By, for each value v in the domain of the attribute. If the domain of a given attribute consists of n values, then n bits are needed for each entry in the bitmap index

The **join indexing** method gained popularity from its use in relational database query processing. Traditional indexing maps the value in a given column to a list of rows having that value. In contrast, join indexing registers the joinable rows of two relations from a relational database. For example, if two relations R(RID;A) and S(B; SID) join on the attributes A and B, then the join index record contains the pair (RID; SID), where RID and SID are record identifiers from the R and S relations, respectively.

Efficient processing of OLAP queries

1. Determine which operations should be performed on the available cuboids. This involves transforming any selection, projection, roll-up (group-by) and drill-down operations specified in the query into corresponding SQL and/or OLAP operations. For example, slicing and dicing of a data cube may correspond to selection and/or projection operations on a materialized cuboid.

2. Determine to which materialized cuboid(s) the relevant operations should be applied. This involves identifying all of the materialized cuboids that may potentially be used to answer the query.

From Data Warehousing to Data mining

3.5.1 Data Warehouse Usage:

Three kinds of data warehouse applications

Information processing

supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs

Analytical processing

multidimensional analysis of data warehouse data

supports basic OLAP operations, slice-dice, drilling, pivoting

Data mining

knowledge discovery from hidden patterns

supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.
Differences among the three tasks

Note:

From On-Line Analytical Processing to On Line Analytical Mining (OLAM) called from data warehousing to data mining

3.5.2 From on-line analytical processing to on-line analytical mining.

On-Line Analytical Mining (OLAM) (also called OLAP mining), which integrates on-line analytical processing (OLAP) with data mining and mining knowledge in multidimensional databases, is particularly important for the following reasons.

1. High quality of data in data warehouses.

Most data mining tools need to work on integrated, consistent, and cleaned data, which requires costly data cleaning, data transformation and data integration as preprocessing steps. A data warehouse constructed by such preprocessing serves as a valuable source of high quality data for OLAP as well as for data mining.

2. Available information processing infrastructure surrounding data warehouses.

Comprehensive information processing and data analysis infrastructures have been or will be systematically constructed surrounding data warehouses, which include accessing, integration, consolidation, and transformation of multiple, heterogeneous databases, ODBC/OLEDB connections, Web-accessing and service facilities, reporting and OLAP analysis tools.

3. OLAP-based exploratory data analysis.

Effective data mining needs exploratory data analysis. A user will often want to traverse through a database, select portions of relevant data, analyze them at different granularities, and present knowledge/results in different forms. On-line analytical mining provides facilities for data mining on different subsets of data and at different levels of abstraction, by drilling, pivoting, filtering, dicing and slicing on a data cube and on some intermediate data mining results.

4. On-line selection of data mining functions.

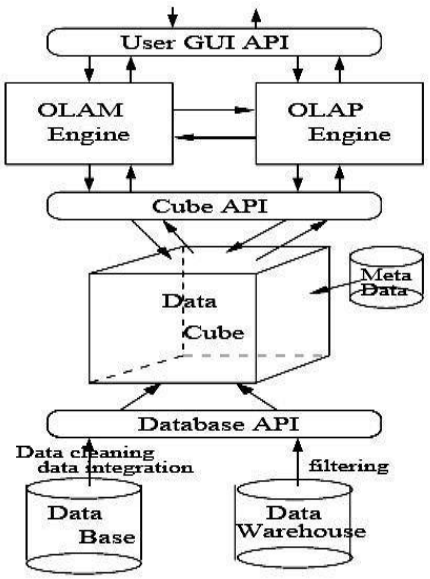
By integrating OLAP with multiple data mining functions, on-line analytical mining provides users with the exibility to select desired data mining functions and swap data mining tasks dynamically.

Architecture for on-line analytical mining

An OLAM engine performs analytical mining in data cubes in a similar manner as an OLAP engine performs on-line analytical processing. An integrated OLAM and OLAP architecture is shown in Figure, where the OLAM and OLAP engines both accept users' on-line queries via a User GUI API and work with the data cube in the data analysis via a Cube API.

A metadata directory is used to guide the access of the data cube. The data cube can be constructed by accessing and/or integrating multiple databases and/or by filtering a data warehouse via a Database API which may support OLEDB or ODBC connections. Since an OLAM engine may perform multiple data mining tasks, such as concept description, association, classification, prediction, clustering, time-series analysis ,etc., it usually consists of multiple, integrated data mining modules and is more sophisticated than an OLAP engine.

Figure: An integrated OLAM and OLAP architecture.



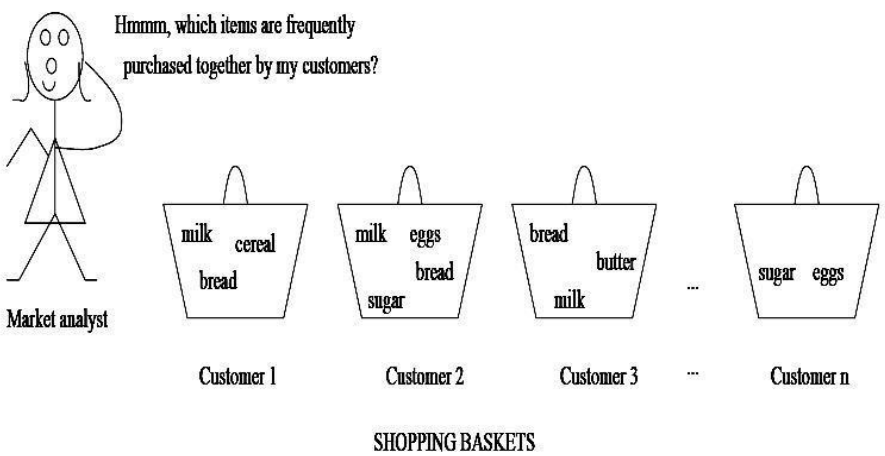
UNIT III

Basic Concepts and a Road Map

Market – Basket analysis

A market basket is a collection of items purchased by a customer in a single transaction, which is a well-defined business activity. For example, a customer's visits to a grocery store or an online purchase from a virtual store on the Web are typical customer transactions. Retailers accumulate huge collections of transactions by recording business activities over time. One common analysis run against a transactions database is to find sets of items, or *itemsets*, that appear together in many transactions. A business can use knowledge of these patterns to improve the Placement of these items in the store or the layout of mail- order catalog page and Web pages. An itemset containing *i* items is called an *i-itemset*. The percentage of transactions that contain an itemset is called the itemsets *support*. For an itemset to be interesting, its support must be higher than a user-specified minimum. Such itemsets are said to be frequent.

Figure : Market basket analysis.



$$computer \Rightarrow financial\_management\_software \quad [support = 2\%, confidence = 60\%]$$

Rule support and confidence are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% for association Rule means that 2% of all the transactions under analysis show that computer and financial management software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.

Frequent Itemsets, Closed Itemsets, and Association Rules

Association rule mining:

- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

Applications:

- Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

Examples.

- Rule form: "Body ® Head [support, confidence]".
- buys(x, "diapers") ® buys(x, "beers") [0.5%, 60%]
- major(x, "CS") ^ takes(x, "DB") ® grade(x, "A") [1%, 75%]

Association Rule: Basic Concepts

Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)

Find: all rules that correlate the presence of one set of items with that of another set of items

- E.g., *98% of people who purchase tires and auto accessories also get automotive services done*

Applications

- * □ *Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
- *Home Electronics* □ * (What other products should the store stocks up?)

- Attached mailing in direct marketing
- Detecting “ping-pong”ing of patients, faulty “collisions”

Rule Measures: Support and Confidence

Find all the rules *X & Y* □ *Z* with minimum confidence and support
- support, *s*, probability that a transaction contains {X □ Y □Z}
- confidence, *c*, conditional probability that a transaction having {X □ Y} also contains *Z*

Let minimum support 50%, and minimum confidence 50%, we have

- *A* □ *C* (50%, 66.6%)
- *C* □ *A* (50%, 100%)

Association Rule Mining: A Map

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Boolean vs. quantitative associations (Based on the types of values handled)
- buys(x, “SQLServer”) ^ buys(x, “DMBook”) ® buys(x, “DBMiner”) [0.2%, 60%]
- age(x, “30..39”) ^ income(x, “42..48K”) ® buys(x, “PC”) [1%, 75%]

Single dimension vs. multiple dimensional associations (see ex. Above)
Single level vs. multiple-level analysis
- What brands of beers are associated with what brands of diapers?

Various extensions
- Correlation, causality analysis
  Association does not necessarily imply correlation or causality
- Maxpatterns and closed itemsets
- Constraints enforced
  E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

Efficient and Scalable Frequent Itemset Mining Methods

Mining Frequent Patterns

The method that mines the complete set of frequent itemsets with candidate generation.
Apriori property & The Apriori Algorithm.

Apriori property

All nonempty subsets of a frequent item set most also be frequent.

- An item set I does not satisfy the minimum support threshold, min-sup, then I is not frequent, i.e., support(I) < min-sup
- If an item A is added to the item set I then the resulting item set (I U A) can not occur more frequently than I.

Monotonic functions are functions that move in only one direction.
This property is called anti-monotonic.
If a set cannot pass a test, all its supersets will fail the same test as well.
This property is monotonic in failing the test.

The Apriori Algorithm

Join Step: C_k is generated by joining $L_{k-1}$with itself

Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Input: Database, D, of transactions; minimum support threshold, *min_sup*.

Output: L, frequent itemsets in D.

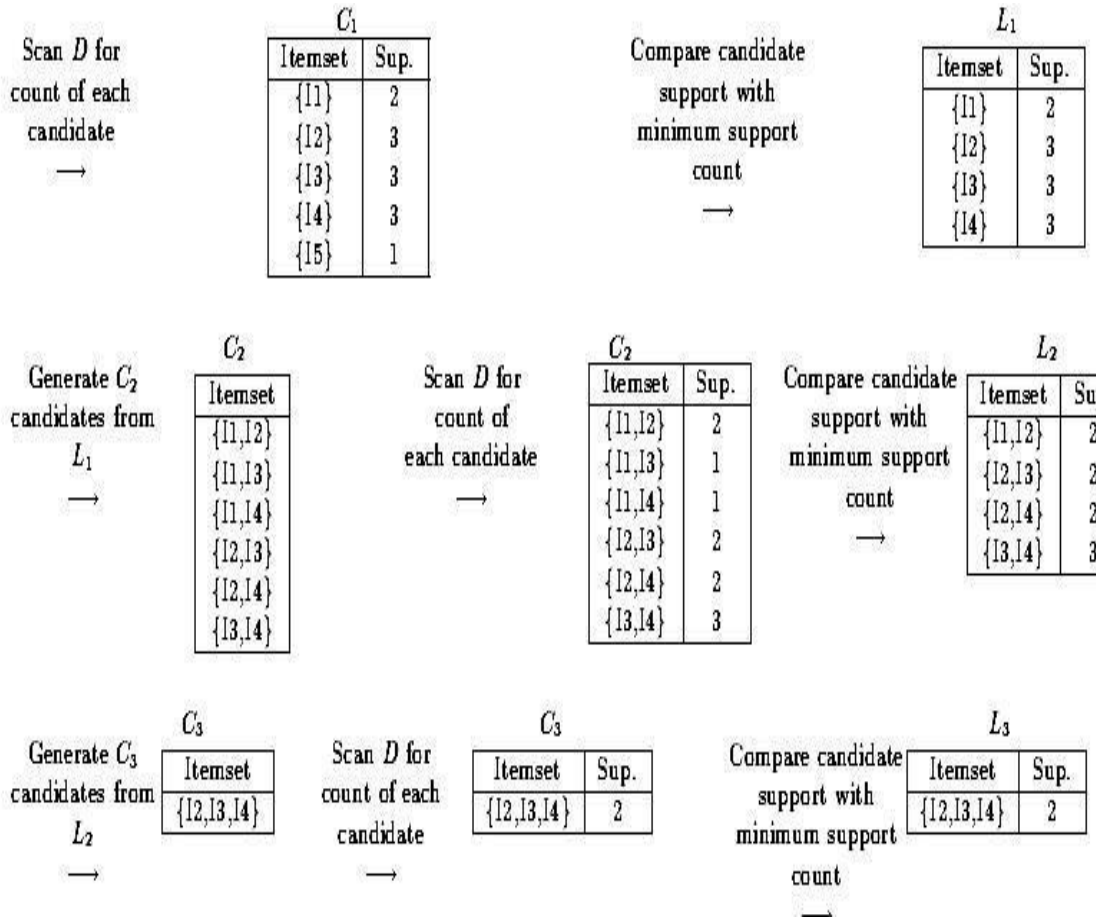
Method:

```
1)  L1 = find_frequent_1-itemsets(D);
2)  for (k = 2; Lk−1 ≠ ∅; k++) {
3)      Ck = apriori_gen(Lk−1, min_sup);
4)      for each transaction t ∈ D { // scan D for counts
5)          Ct = subset(Ck, t); // get the subsets of t that are candidates
6)          for each candidate c ∈ Ct
7)              c.count++;
8)      }
9)      Lk = {c ∈ Ck | c.count ≥ min_sup}
10) }
11) return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k-1) itemsets; min_sup: minimum support)
1)  for each itemset l1 ∈ Lk−1
2)      for each itemset l2 ∈ Lk−1
3)          if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
4)              c = l1 ⋈ l2; // join step: generate candidates
5)              if has_infrequent_subset(c, Lk−1) then
6)                  delete c; // prune step: remove unfruitful candidate
7)              else add c to Ck;
8)          }
9)  return Ck;

procedure has_infrequent_subset(c: candidate k-itemset; Lk−1: frequent (k − 1) itemsets); // use prior knowledge
1)  for each (k − 1)-subset s of c
2)      if s ∉ Lk−1 then
3)          return TRUE;
4)  return FALSE;
```

Example



The method that mines the complete set of frequent itemsets without generation.

Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure

- highly condensed, but complete for frequent pattern mining
- avoid costly database scans

Develop an efficient, FP-tree-based frequent pattern mining method

- A divide-and-conquer methodology: decompose mining tasks into smaller ones
- Avoid candidate generation: sub-database test only!

Construct FP-tree from a Transaction DB

TID	***Items bought***	***(ordered) frequent items***
100	***{f, a, c, d, g, i, m, p}***	***{f, c, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{f, c, a, b, m}***
min_support = 0.5		
300	***{b, f, h, j, o}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{f, c, a, m, p}***

Steps:

Scan DB once, find frequent 1-itemset (single item pattern)
Order frequent items in frequency descending order
Scan DB again, construct FP-tree

Header Table

Item	***frequency head***
f	*4*
c	*4*
	3
	3
m	*3*
p	*3*

Benefits of the FP-tree Structure

Completeness:
- never breaks a long pattern of any transaction
- preserves complete information for frequent pattern mining

Compactness
- reduce irrelevant information—infrequent items are gone
- frequency descending ordering: more frequent items are more likely to be shared

– never be larger than the original database (if not count node-links and counts)
– Example: For Connect-4 DB, compression ratio could be over 100

Mining Frequent Patterns Using FP-tree

General idea (divide-and-conquer)
– Recursively grow frequent pattern path using the FP-tree
Method
– For each item, construct its conditional pattern-base, and then its conditional FP-tree
– Repeat the process on each newly created conditional FP-tree
– Until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

Major Steps to Mine FP-tree

Construct conditional pattern base for each node in the FP-tree
Construct conditional FP-tree from each conditional pattern-base
Recursively mine conditional FP-trees and grow frequent patterns obtained so far
If the conditional FP-tree contains a single path, simply enumerate all the patterns

Principles of Frequent Pattern Growth

Pattern growth property
– Let □ be a frequent itemset in DB, B be □ 's conditional pattern base, and □ be an itemset in B. Then □ □ □ is a frequent itemset in DB iff □ is frequent in B.
"*abcdef* " is a frequent pattern, if and only if
– "*abcde* " is a frequent pattern, and
– "*f* " is frequent in the set of transactions containing "*abcde* "

Why Is Frequent Pattern Growth Fast?

Our performance study shows

– FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection

Reasoning

– No candidate generation, no candidate test
– Use compact data structure
– Eliminate repeated database scan

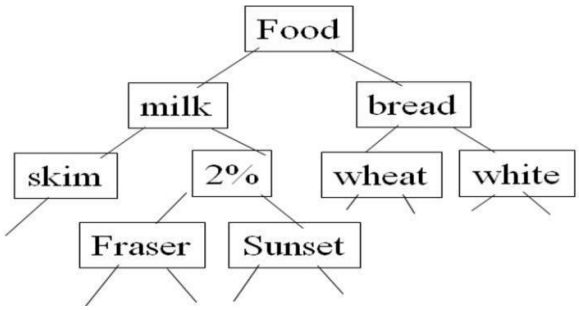
Basic operation is counting and FP-tree building

Mining Various Kinds of Association Rules

Mining multilevel association rules from transactional databases

Multilevel association rule: Multilevel association rules can be defined as applying association rules over different levels of data abstraction

Ex:



Steps to perform multilevel association rules from transactional database are:

Step1:consider frequent item sets

Step2: arrange the items in hierarchy form

Step3: find the Items at the lower level (expected to have lower

support) Step4: Apply association rules on frequent item sets

Step5: Use some methods and identify frequent

itemsets Note: Support is categorized into two types

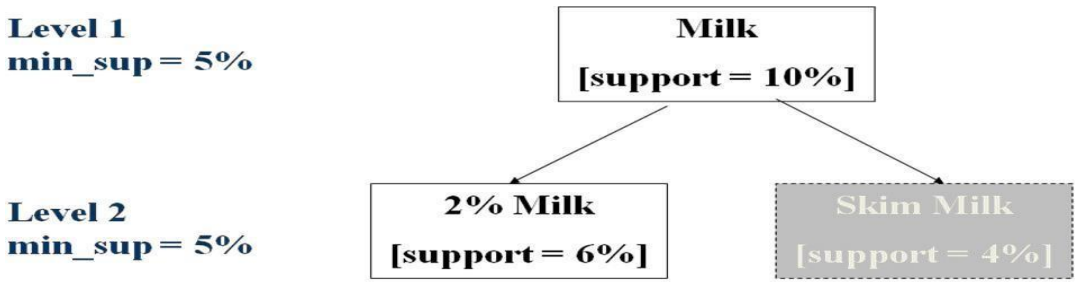
Uniform Support: the same minimum support for all levels

Reduced Support: reduced minimum support at lower levels

*multilevel association rules can be applied on both the

supports Example figure for Uniform Support:

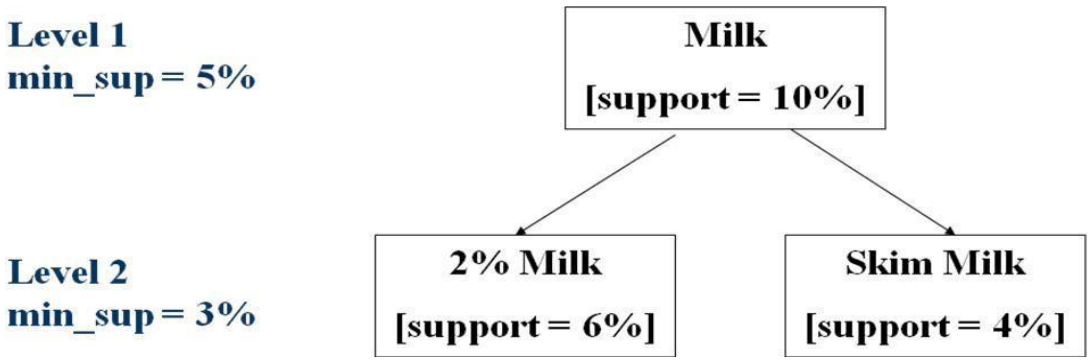
Multi-level mining with uniform support



Back

Reduced Support:

*Multi-level mining with reduced support



Mining multidimensional association rules from Relational databases and data warehouse

Multi dimensional association rule: Multi dimensional association rule can be defined as the statement which contains only two (or) more predicates/dimensions

**Multi dimensional association rule also called as Inter dimensional association rule

We can perform the following association rules on relational database and data warehouse

* 1)Boolean dimensional association rule: Boolean dimensional association rule can be defined as comparing existing predicates/dimensions with non existing predicates/dimensions..

2) Single dimensional association rule: Single dimensional association rule can be defined as the statement which contains only single predicate/dimension

**Single dimensional association rule also called as Intra dimensional association rule as usually multi dimensional association rule..

** **Multi dimensional association rule can be applied on different types of attributes ..here** the attributes are

1.Categorical Attributes

– finite number of possible values, no ordering among values

Quantitative Attributes

– numeric, implicit ordering among values

Note1:Relational database can be viewed in the form of tables…so on tables we are performing the concept hierarchy

In relational database we are using the concept hierarchy that means generalization in order to find out the frequent item sets

Generalization: replacing low level attributes with high level attributes called generalization

Note2: data warehouse can be viewed in the form of multidimensional data model (uses data cubes) in order to find out the frequent patterns.

From association mining to correlation analysis:

*Here Association mining to correlation analysis can be performed Based on interesting measures of frequent items.

*among frequent items we are performing correlation analysis

*correlation analysis means one frequent item is dependent on other frequent item..

Consider Two popular measurements:

support; and

confidence

for each frequent item we are considering the above mentioned two measures to perform mining and correlation analysis..

Note: **Association mining:** Association mining can be defined as finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories

Constraint-based association mining:

*constraint based association mining can be defined as applying different types of constraints on different types of knowledge

*so the kinds of constraints used in the mining are

1.Knowledge type constraint
Data constraint
Dimension/level constraints
Rule constraints
Interestingness constraints

Knowledge type constraint: Based on classification, association, we are applying the knowledge type constraints.

Data constraints: SQL-like queries

Ex: Find product pairs sold together in Vancouver in Dec.'98.

3. Dimension/level constraints:
   in relevance to region, price, brand, customer category.

4. Rule constraints:
   On the form of the rules to be mined (e.g., # of predicates, etc) small sales (price < $10) triggers big sales (sum > $200).

5. Interestingness constraints:
   1. Thresholds on measures of interestingness
   2. strong rules (min_support □ 3%, min_confidence □60%).

UNIT IV
Classification and Prediction

What is classification? What

is prediction? Classification:

*****used for prediction(future analysis) to know the unknown attributes with their values.by using classifier algorithms and decision tree.(in data mining)

*which constructs some models(like decision trees) then which classifies the attributes..

*already we know the types of attributes are 1.categorical attribute and 2.numerical attribute

*these classification can work on both the above mentioned attributes.

Prediction: prediction also used for to know the unknown or missing values..
which also uses some models in order to predict the attributes
models like neural networks, if else rules and other mechanisms

Classification and prediction are used in the Applications
like *credit approval
*target marketing
*medical diagnosis

Classification—A Two-Step Process

Model construction: describing a set of predetermined classes

– Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute

– The set of tuples used for model construction: training set

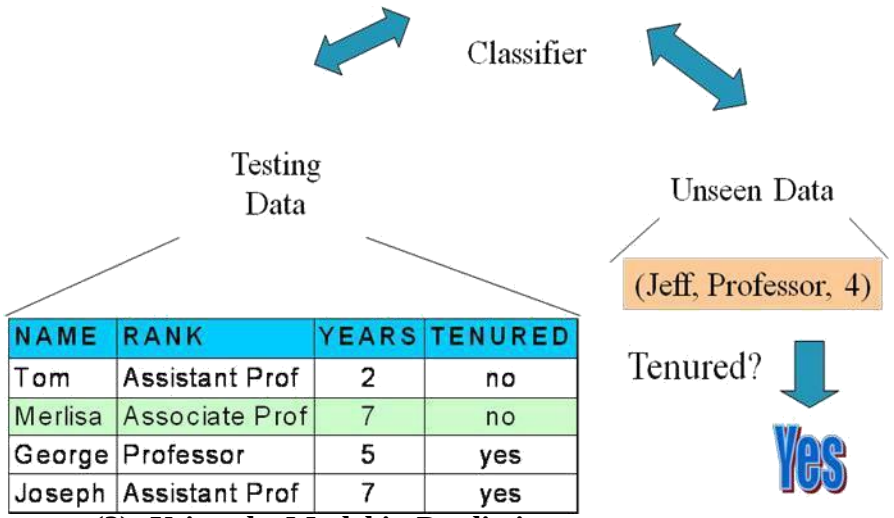
– The model is represented as classification rules, decision trees, or mathematical formulae

Model usage: for classifying future or unknown objects

– Estimate accuracy of the model

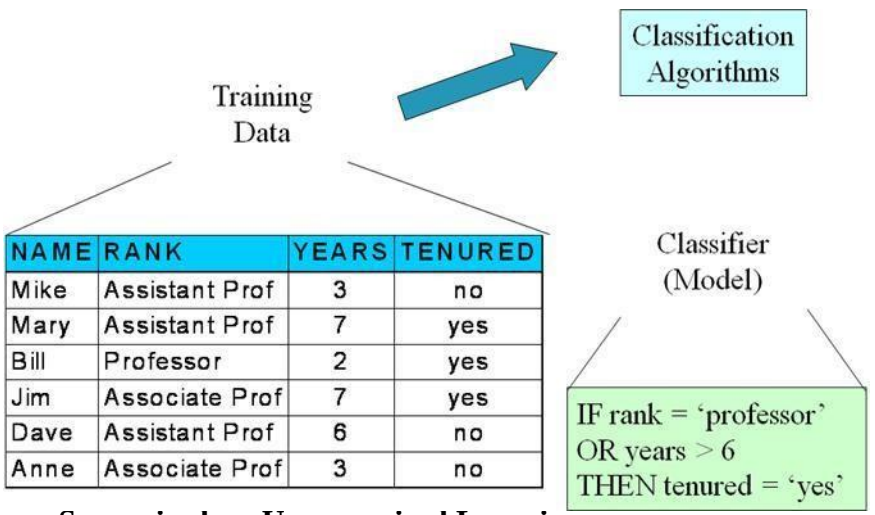
The known label of test sample is compared with the classified result from the model
Accuracy rate is the percentage of test set samples that are correctly classified by the model.
Test set is independent of training set, otherwise over-fitting will occur

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Supervised vs. Unsupervised Learning
Supervised learning (classification)

Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
New data is classified based on the training set

Unsupervised learning (clustering)

The class labels of training data is unknown

Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Issues regarding classification and prediction:

There are two issues regarding classification and prediction they are

Issues (1): Data Preparation

Issues (2): Evaluating Classification Methods

Issues (1): Data Preparation: Issues of data preparation includes the following

1) Data cleaning

*Preprocess data in order to reduce noise and handle missing values (refer preprocessing techniques i.e. data cleaning notes)

2) Relevance analysis (feature selection)

Remove the irrelevant or redundant attributes (refer unit-iv AOI Relevance analysis)

Data transformation (refer preprocessing techniques i.e data cleaning notes)

Generalize and/or normalize data

Issues (2): Evaluating Classification Methods: considering classification methods should satisfy the following properties

Predictive accuracy

Speed and scalability

*time to construct the model

*time to use the model

3. Robustness

*handling noise and missing values

4. Scalability

*efficiency in disk-resident databases

5. **Interpretability**:

*understanding and insight provided by the model

6. Goodness of rules

*decision tree size

*compactness of classification rules

Classification by Decision Tree Induction

Decision tree

- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

Decision tree generation consists of two phases

- Tree construction
  - At start, all the training examples are at the root
  - Partition examples recursively based on selected attributes
- Tree pruning
  - Identify and remove branches that reflect noise or outliers

Use of decision tree: Classifying an unknown sample

- Test the attribute values of the sample against the decision tree

Training Dataset

This follows an example from Quinlan's ID3

age	Income	student	credit_rating
<=30	High	no	fair
<=30	High	no	excellent
31…40	High	no	fair
>40	Medium	no	fair
>40	Low	yes	fair
>40	Low	yes	excellent
31…40	Low	yes	excellent
<=30	Medium	no	fair
<=30	Low	yes	fair
>40	Medium	yes	fair
<=30	Medium	yes	excellent
31…40	Medium	no	excellent
31…40	high	yes	fair
>40	medium	no	excellent

Algorithm for Decision Tree Induction

Basic algorithm (a greedy algorithm)

- Tree is constructed in a top-down recursive divide-and-conquer manner

- At start, all the training examples are at the root
- Attributes are categorical (if continuous-valued, they are discretized in advance)
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

Avoid Overfitting in Classification

The generated tree may overfit the training data

- Too many branches, some may reflect anomalies due to noise or outliers
- Result is in poor accuracy for unseen samples

Two approaches to avoid over fitting

- Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - Difficult to choose an appropriate threshold
- Post pruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
  - Use a set of data different from the training data to decide which is the "best pruned tree"

Tree Mining in Weka and Tree Mining in Clementine.

Tree Mining in Weka

Example:

- Weather problem: build a decision tree to guide the decision about whether or not to play tennis.
- Dataset
  (weather.nominal.arff)

Validation:

- Using training set as a test set will provide optimal classification accuracy.
- Expected accuracy on a different test set will always be less.
- 10-fold cross validation is more robust than using the training set as a test set.
    - Divide data into 10 sets with about same proportion of class label values as in original set.
    - Run classification 10 times independently with the remaining 9/10 of the set as the training set.
    - Average accuracy.
- Ratio validation: 67% training set / 33% test set.
- Best: having a separate training set and test set.

Results:

- Classification accuracy (correctly classified instances).
- Errors (absolute mean, root squared mean, …)
- Kappa statistic (measures agreement between predicted and observed classification; -100%-100% is the proportion of agreements after chance agreement has been excluded; 0% means complete agreement by chance)

Results:

- TP (True Positive) rate per class label
- FP (False Positive) rate
- Precision = TP rate = TP / (TP + FN)) * 100%
- Recall = TP / (TP + FP)) * 100%
- F-measure = 2* recall * precision / recall + precision

ID3 characteristics:

- Requires nominal values
- Improved into C4.5
    - Dealing with numeric attributes
    - Dealing with missing values
    - Dealing with noisy data
    - Generating rules from trees

Tree Mining in Clementine

Methods:

– C5.0: target field must be categorical, predictor fields may be numeric or categorical, provides multiple splits on the field that provides the maximum information gain at each level
– QUEST: target field must be categorical, predictor fields may be numeric ranges or categorical, statistical binary split
– C&RT: target and predictor fields may be numeric ranges or categorical, statistical binary split based on regression
– CHAID: target and predictor fields may be numeric ranges or categorical, statistical binary split based on chi-square

Attribute Selection Measures

Information Gain
Gain ratio
Gini Index

6.3.3 Pruning of decision trees

Discarding one or more sub trees and replacing them with leaves simplify a decision tree, and that is the main task in decision-tree pruning. In replacing the sub tree with a leaf, the algorithm expects to lower the *predicted error rate and* increase the quality of a classification model. But computation of error rate is not simple. An error rate based only on a training data set does not provide a suitable estimate. One possibility to estimate the predicted error rate is to use a new, additional set of test samples if they are available, or to use the cross-validation techniques. This technique divides initially available samples into equal sized blocks and, for each block, the tree is constructed from all samples except this block and tested with a given block of samples. With the available training and testing samples, the basic idea of decision tree-pruning is to remove parts of the tree (sub trees) that do not contribute to the classification accuracy of unseen testing samples, producing a less complex and thus more comprehensible tree. There are two ways in which the recursive-partitioning method can be modified:

Deciding not to divide a set of samples any further under some conditions. The stopping criterion is usually based on some statistical tests, such as the χ^2 test: *If there are no significant differences in classification accuracy before and after division, then represent a current node as a leaf*. The decision is made in advance, before splitting, and therefore this approach is called *pre pruning*.

Removing restrospectively some of the tree structure using selected accuracy criteria. The decision in this process of *post pruning* is made after the tree has been built.

C4.5 follows the *post pruning* approach, but it uses a specific technique to estimate the predicted error rate. This method is called *pessimistic pruning*. For every node in a tree, the estimation of the upper confidence limit u_{cf} is computed using the statistical tables for binomial distribution (given in most textbooks on statistics). Parameter U_{cf} is a function of $| T_i|$ and E for a given node. C4.5 uses the default confidence level of 25%, and compares $U_{25\%}$ ($| T_i|$ /E) for a given node T_i with a weighted confidence of its leaves. Weights are the total number of cases for every leaf. If the predicted error for a root node in a sub tree is less than weighted sum of $U_{25\%}$ for the leaves (predicted error for the sub tree), then a sub tree will be replaced with its root node, which becomes a new leaf in a pruned tree.

Let us illustrate this procedure with one simple example. A sub tree of a decision tree is given in Figure, where the root node is the test x_1 on three possible values {1, 2, 3} of the attribute A. The children of the root node are leaves denoted with corresponding classes and ($| T_i|$ /E) parameters. The question is to estimate the possibility of pruning the sub tree and replacing it with its root node as a new, generalized leaf node.

To analyze the possibility of replacing the sub tree with a leaf node it is necessary to compute a predicted error PE for the initial tree and for a replaced node. Using default confidence of 25%, the upper confidence limits for all nodes are collected from statistical tables: $U_{25\%}$ (6, 0) = 0.206, $U_{25\%}$(9, 0) = 0.143, $U_{25\%}$(1, 0) = 0.750, and $U_{25\%}$(16, 1) = 0.157. Using these values, the predicted errors for the initial tree and the replaced node are

PE_{tree} = 6.0.206 + 9.0.143 + 1.0.750 = 3.257

PE_{node} = 16.0.157 = 2.512

Since the existing subtree has a higher value of predicted error than the replaced node, it is recommended that the decision tree be pruned and the subtree replaced with the new leaf node.

Bayesian Classification:

Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
Probabilistic prediction: Predict multiple hypotheses, weighted by th**eir probabilities**
Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem

Given training data *D, posteriori probability of a hypothesis h, P(h|D)* follows the Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayes Classifier (I)

A simplified assumption: attributes are conditionally independent:

$$P(C_j \mid V) = P(C_j) \prod_{i=1}^{n} P(v_i \mid C_j)$$

- Greatly reduces the computation cost, only count the class distribution.

Naive Bayesian Classifier (II)

Given a training set, we can compute the probabilities

Outlook	P	N		Humidity	P	N
sunny	2/9	3/5		high	3/9	4/5
overcast	4/9	0		normal	6/9	1/5
rain	3/9	2/5				
Temperature				W indy		
hot	2/9	2/5		true	3/9	3/5
mild	4/9	2/5		false	6/9	2/5

cool	3/9	1/5				

Bayesian classification

The classification problem may be formalized using a-posteriori probabilities:
P(C|X) = prob. that the sample tuple
X=<x1,…,xk> is of class C.
E.g. P(class=N | outlook=sunny, windy=true,…)
Idea: assign to sample X the class label C such that P(C|X) is maximal

Estimating a-posteriori probabilities

Bayes theorem:
$P(C|X) = P(X|C)\cdot P(C) / P(X)$

P(X) is constant for all classes
P(C) = relative freq of class C samples
C such that P(C|X) is maximum =
C such that $P(X|C)\cdot P(C)$ is maximum
Problem: computing P(X|C) is unfeasible!

Naïve Bayesian Classification

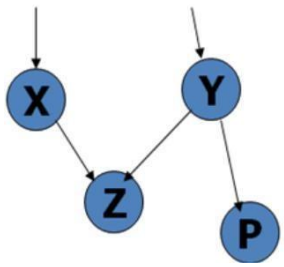
Naïve assumption: attribute independence
$P(x1,\ldots,xk|C) = P(x1|C)\cdot\ldots\cdot P(xk|C)$

If i-th attribute is categorical:
P(xi|C) is estimated as the relative freq of samples having value xi as i-th attribute in class C
If i-th attribute is continuous:
P(xi|C) is estimated thru a Gaussian density function
Computationally easy in both cases

Bayesian Belief Networks

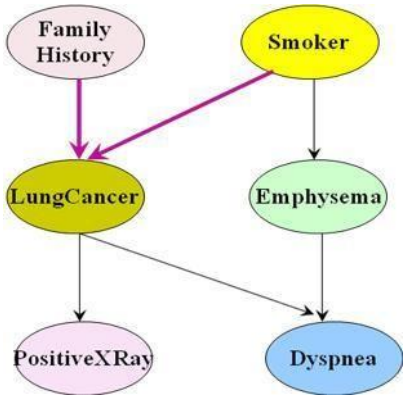
Bayesian belief network allows a *subset* of the variables conditionally independent
A graphical model of causal relationships
Represents dependency among the variables

Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops or cycles

Bayesian Belief Network: An Example



The conditional probability table (CPT) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of X, from CPT:

$$P(x_1,..., x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

Association-Based Classification

Several methods for association-based classification
– ARCS: Quantitative association mining and clustering of association rules (Lent et al'97)
It beats C4.5 in (mainly) scalability and also accuracy
– Associative classification: (Liu et al'98)
It mines high support and high confidence rules in the form of
"cond_set => y", where y is a class label
– CAEP (Classification by aggregating emerging patterns) (Dong et al'99)

Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
Mine Eps based on minimum support and growth rate

6.9 Lazy Learners

The *k*-Nearest Neighbor Algorithm

All instances correspond to points in the n-D space

The nearest neighbor are defined in terms of Euclidean distance, dist(X_1, X_2)

Target function could be discrete- or real- valued

For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q

k-NN for real-valued prediction for a given unknown tuple

Returns the mean values of the *k* nearest neighbors

Distance-weighted nearest neighbor algorithm

Weight the contribution of each of the k neighbors according to their distance to the query x_q

Give greater weight to closer neighbors

Robust to noisy data by averaging k-nearest neighbors

Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes

To overcome it, axes stretch or elimination of the least relevant attributes

UNIT V

Cluster Analysis

What is Cluster Analysis?

Cluster: a collection of data objects
- Similar to one another within the same cluster
- Dissimilar to the objects in other clusters

Cluster analysis
- Grouping a set of data objects into clusters

Clustering is unsupervised classification: no predefined classes

Typical applications
- As a stand-alone tool to get insight into data distribution
- As a preprocessing step for other algorithms

General Applications of Clustering

Pattern Recognition

Spatial Data Analysis
- create thematic maps in GIS by clustering feature spaces
- detect spatial clusters and explain them in spatial data mining

Image Processing

Economic Science (especially market research)

WWW
- Document classification
- Cluster Weblog data to discover groups of similar access patterns

Examples of Clustering Applications

Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs

Land use: Identification of areas of similar land use in an earth observation database

Insurance: Identifying groups of motor insurance policy holders with a high average claim cost

City-planning: Identifying groups of houses according to their house type, value, and geographical location

Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

What Is Good Clustering?

A good clustering method will produce high quality clusters with

- high intra-class similarity
- low inter-class similarity

The quality of a clustering result depends on both the similarity measure used by the method and its implementation.

The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

Scalability
Ability to deal with different types of attributes
Discovery of clusters with arbitrary shape
Minimal requirements for domain knowledge to determine input parameters
Able to deal with noise and outliers
Insensitive to order of input records
High dimensionality
Incorporation of user-specified constraints
Interpretability and usability

Type of data in clustering analysis

Interval-scaled variables:
Binary variables:
Nominal, ordinal, and ratio variables:
Variables of mixed types:

Interval-valued variables

Standardize data

- Calculate the mean absolute deviation:

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf})$$

Where

- Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Between Objects

Distances are normally used to measure the similarity or dissimilarity between two data objects

Some popular ones include: *Minkowski distance*:

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

where i = (xi1, xi2, …, xip) and j = (xj1, xj2, …, xjp) are two p-dimensional data objects, and q is a positive integer

If q = 1, d is Manhattan distance

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

If q = 2, d is Euclidean distance:

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

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|)$$

– *Properties*

$d(i,j) \geq 0$

$d(i,i) = 0$

$d(i,j) = d(j,i)$

$d(i,j) \leq d(i,k) + d(k,j)$

Also one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures.

Binary Variables

A contingency table for binary data

		Object j	
	1	0	*sum*
1	*a*	*b*	$a+b$
0	*c*	*d*	$c+d$
sum	$a+c$	$b+d$	p

Object i

Simple matching coefficient (invariant, if the binary variable is symmetric):
Jaccard coefficient (noninvariant if the binary variable is asymmetric):

Dissimilarity between Binary Variables

Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3		Test-4
Jack	M	Y	N	P	N	N	N	
Mary	F	Y	N	P	N	P	N	
Jim	M	Y	P	N	N	N	N	

– gender is a symmetric attribute
– the remaining attributes are asymmetric binary
– let the values Y and P be set to 1, and the value N be set to 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+1} = 0.75$$

Nominal Variables

A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green

Method 1: Simple matching

- *m: # of matches, p: total # of variables*

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

- *Method 2: use a large number of binary variables*
  - *creating a new binary variable for each of the M nominal states*

Ordinal Variables

An ordinal variable can be

- *discrete or continuous*
- *order is important, e.g., rank*
- *Can be treated like interval-scaled*
  - *replacing xif by their rank*
  - *map the range of each variable onto [0, 1] by replacing i-th object in the f-th variable by*

$$r_{if} \in \{1, \ldots, M_f\}$$

$$z = \frac{r - 1}{M - 1}$$

compute the dissimilarity using methodsf for interval-scaled variables

Ratio-Scaled Variables

Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale,
such as AeBt or Ae-Bt

Methods:

- *treat them like interval-scaled variables — not a good choice! (why?)*
- *apply logarithmic transformation*

yif = log(xif)

- *treat them as continuous ordinal data treat their rank as interval-scaled.*

Variables of Mixed Types

A database may contain all the six types of variables

- *symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio.*

One may use a weighted formula to combine their effects.

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

- *f is binary or nominal:*

dij(f) = 0 if xif = xjf , or dij(f) = 1 o.w.

- *f is interval-based: use the normalized distance*
- *f is ordinal or ratio-scaled*
  - *compute ranks rif and*
  - *and treat zif as interval-scaled*

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

A Categorization of Major Clustering Methods:

Partitioning algorithms: Construct various partitions and then evaluate them by some criterion

Hierarchy algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion

Density-based: based on connectivity and density functions

Grid-based: based on a multiple-level granularity structure

Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Partitioning Algorithms: Basic Concept

Partitioning method: Construct a partition of a database *D* of *n* objects into a set of *k* clusters

Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion

- Global optimal: exhaustively enumerate all partitions
- Heuristic methods: *k-means* and *k-medoids* algorithms
- *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
- *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

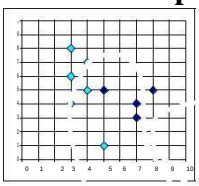
The *K-Means* Clustering Method

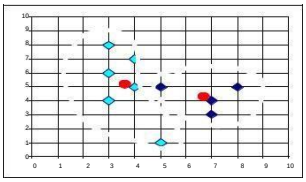
Given *k*, the *k-means* algorithm is implemented in 4 steps:

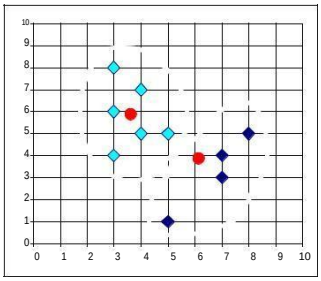
- Partition objects into *k* nonempty subsets
- Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
- Assign each object to the cluster with the nearest seed point.
- Go back to Step 2, stop when no more new assignment.

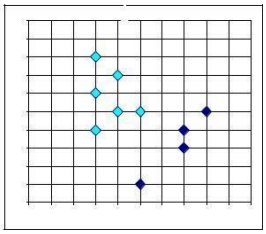
The *K-Means* Clustering Method

- **Example**









Comments on the *K-Means* Method

Strength

- *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
- Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*

Weakness

- Applicable only when *mean* is defined, then what about categorical data?
- Need to specify *k*, the *number* of clusters, in advance
- Unable to handle noisy data and *outliers*
- Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

A few variants of the *k-means* which differ in

- Selection of the initial *k* means
- Dissimilarity calculations
- Strategies to calculate cluster means

Handling categorical data: *k-modes* (Huang'98)

- Replacing means of clusters with modes
- Using new dissimilarity measures to deal with categorical objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: *k-prototype* method

The *K-Medoids* Clustering Method

Find *representative* objects, called medoids, in clusters

PAM (Partitioning Around Medoids, 1987)

- starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
- *PAM* works effectively for small data sets, but does not scale well for large data sets

CLARA (Kaufmann & Rousseeuw, 1990)

CLARANS (Ng & Han, 1994): Randomized sampling

Focusing + spatial data structure (Ester et al., 1995)

edoids Clustering Method

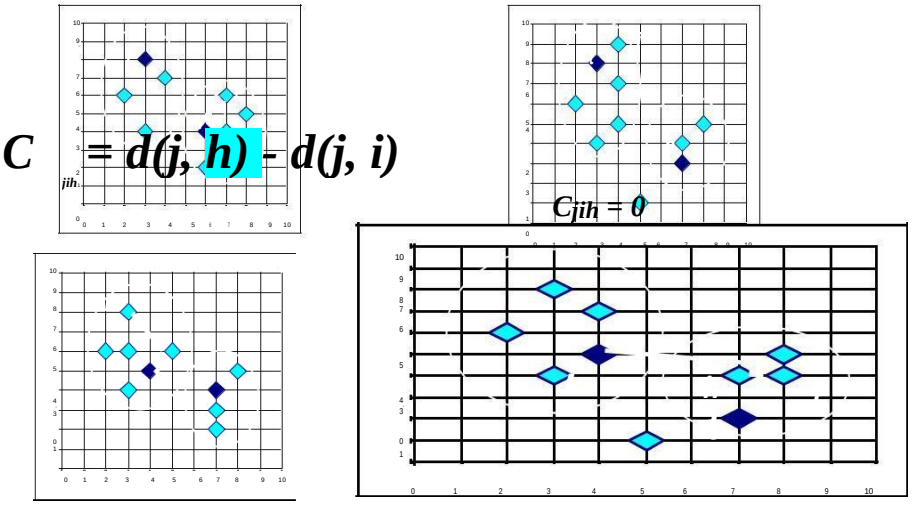
Find *representative* objects, called medoids, in clusters
PAM (Partitioning Around Medoids, 1987)
- starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
- *PAM* works effectively for small data sets, but does not scale well for large data sets

CLARA (Kaufmann & Rousseeuw, 1990)
CLARANS (Ng & Han, 1994): Randomized sampling
Focusing + spatial data structure (Ester et al., 1995)

PAM (Partitioning Around Medoids) (1987)

PAM (Kaufman and Rousseeuw, 1987), built in Splus
Use real object to represent the cluster
- Select ***k*** representative objects arbitrarily
- For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost ***TCih***
- For each pair of ***i*** and ***h***,
  If *TCih* < 0, ***i*** is replaced by ***h***
  Then assign each non-selected object to the most similar representative object
- repeat steps 2-3 until there is no change

PAM Clustering: Total swapping cost *TCih*=□*jCjih*



$$C_{jih} = d(j, h) - d(j, t)$$

CLARA (Clustering Large Applications) (1990)

CLARA (Kaufmann and Rousseeuw in 1990)

- Built in statistical analysis packages, such as S+

It draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output

Strength: deals with larger data sets than PAM

Weakness:

- Efficiency depends on the sample size
- A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS ("Randomized" CLARA) (1994)

CLARANS (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)

CLARANS draws sample of neighbors dynamically

The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids

If the local optimum is found, CLARANS starts with new randomly selected node in search for a new local optimum

It is more efficient and scalable than both PAM and CLARA

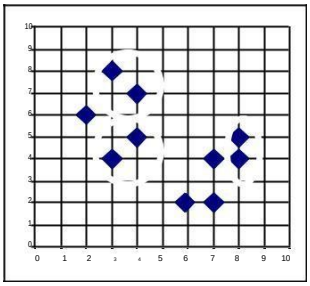
Focusing techniques and spatial access structures may further improve its performance (Ester et al.'95)

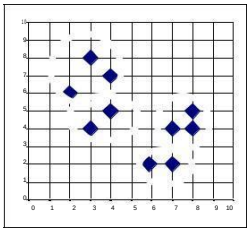
Hierarchical Clustering

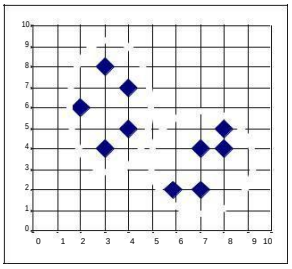
Use distance matrix as clustering criteria. This method does not require the number of clusters *k* as an input, but needs a termination condition

7.5.1 AGNES (Agglomerative Nesting)

Introduced in Kaufmann and Rousseeuw (1990)
Implemented in statistical analysis packages, e.g., Splus
Use the Single-Link method and the dissimilarity matrix.
Merge nodes that have the least dissimilarity
Go on in a non-descending fashion
Eventually all nodes belong to the same cluster







A *Dendrogram* Shows How the Clusters are Merged Hierarchically

Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

DIANA (Divisive Analysis)

Introduced in Kaufmann and Rousseeuw (1990)
Implemented in statistical analysis packages, e.g., Splus
Inverse order of AGNES
Eventually each node forms a cluster on its own

More on Hierarchical Clustering Methods

Major weakness of agglomerative clustering methods
- do not scale well: time complexity of at least $O(n2)$, where n is the number of total objects
- can never undo what was done previously

Integration of hierarchical with distance-based clustering
- BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
- CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
- CHAMELEON (1999): hierarchical clustering using dynamic modeling

7.5.2 BIRCH (1996)

Birch: Balanced Iterative Reducing and Clustering using Hierarchies, by Zhang, Ramakrishnan, Livny (SIGMOD'96)

Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
- Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
- Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

Scales linearly: finds a good clustering with a single scan and improves the quality with a few additional scans

Weakness: handles only numeric data, and sensitive to the order of the data record.

Rock Algorithm and CHAMELEON.

ROCK: Robust Clustering using linKs,
by S. Guha, R. Rastogi, K. Shim (ICDE'99).
- Use links to measure similarity/proximity
- Not distance based
- Computational complexity:

Basic ideas:
- Similarity function and neighbors:

Let $T1$ = {1,2,3}, $T2$={3,4,5}

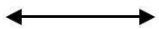
Rock: Algorithm

• Links: The number of common neighbours for the two points.

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}

{1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}

{1,2,3} 3 {1,2,4}



Algorithm

- Draw random sample
- Cluster with links
- Label data in disk

CHAMELEON

CHAMELEON: hierarchical clustering using dynamic modeling, by G. Karypis, E.H. Han and V. Kumar'99

Measures the similarity based on a dynamic model

- Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters

A two phase algorithm

- 1. Use a graph partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
- 2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters

AGGLOMERATIVE HIERARCHICAL CLUSTERING

Algorithms of hierarchical cluster analysis are divided into the two categories divisible algorithms and agglomerative algorithms. A *divisible algorithm* starts from the entire set of samples X and divides it into a partition of subsets, then divides each subset into smaller sets, and so on. Thus, a divisible algorithm generates a sequence of partitions that is ordered from a coarser one to a finer one. An *agglomerative algorithm* first regards each object as an initial cluster. The clusters are merged into a coarser partition, and the merging process proceeds until the trivial partition is obtained: all objects are in one large cluster. This process of clustering is a bottom-up process, where partitions from a finer one to a coarser one.

Most agglomerative hierarchical clustering algorithms are variants of the *single-link* or *complete-link* algorithms. In the single-link method, the distance between two clusters is the *minimum* of the distances between all pairs of samples drawn from the two clusters (one element from the first cluster, the other from the second). In the complete-link algorithm, the distance between two clusters is the *maximum* of all distances between all pairs drawn from the two clusters. A graphical illustration of these two distance measures is given.

The basic steps of the agglomerative clustering algorithm are the same. These steps are

Place each sample in its own cluster. Construct the list of inter-cluster distances for all distinct unordered pairs of samples, and sort this list in ascending order.

Step through the sorted list of distances, forming for each distinct threshold value d_k a graph of the samples where pairs samples closer than d_k are connected into a new cluster by a graph edge. If all the samples are members of a connected graph, stop. Otherwise, repeat this step.

The output of the algorithm is a nested hierarchy of graphs, which can be cut at the desired dissimilarity level forming a partition (clusters) identified by simple connected components in the corresponding sub graph.

Let us consider five points $\{x_1, x_2, x_3, x_4, x_5\}$ with the following coordinates as a two-dimensional sample for clustering:

For this example, we selected two-dimensional points because it is easier to graphically represent these points and to trace all the steps in the clustering algorithm.

The distances between these points using the Euclidian measure are

$d(x_1 , x_2) = 2$, $d(x_1, x_3) = 2.5$, $d(x_1, x_4) = 5.39$, $d(x_1, x_5) = 5$

$d(x_1 , x_3) = 1.5$, $d(x_2, x_4) = 5$, $d(x_2, x_5) = 5.29$,

$d(x_3 , x_4) = 3.5$, $d(x_3, x_5) = 4.03$, $d(x_4, x_5) = 2$

The distances between points as clusters in the first iteration are the same for both single-link and complete-link clustering. Further computation for these two algorithms is different. Using agglomerative single-link clustering, the following

steps are performed to create a cluster and to represent the cluster structure as a dendrogram.

Hierarchical and Non-Hierarchical Clustering

There are two main types of clustering techniques, those that create a hierarchy of clusters and those that do not. The hierarchical clustering techniques create a hierarchy of clusters from small too big. The main reason for this is that, as was already stated, clustering is an unsupervised learning technique, and as such, there is no absolutely correct answer. For this reason and depending on the particular application of the clustering, fewer or greater numbers of clusters may be desired. With a hierarchy of clusters defined it is possible to choose the number of clusters that are desired. At the extreme it is possible to have as many clusters as there are records in the database. In this case the records within the cluster are optimally similar to each other (since there is only one) and certainly different from the other clusters. But of course such a clustering technique misses the point in the sense that the idea of clustering is to find useful patters in the database that summarize it and make it easier to understand. Any clustering algorithm that ends up with as many clusters as there are records has not helped the user understand the data any better. Thus one of the main points about clustering is that there be many fewer clusters than there are original records. Exactly how many clusters should be formed is a matter of interpretation. The advantage of hierarchical clustering methods is that they allow the end user to choose from either many clusters or only a few.

The hierarchy of clusters is usually viewed as a tree where the smallest clusters merge together to create the next highest level of clusters and those at that level merge together to create the next highest level of clusters. Figure 1.5 below shows how several clusters might form a hierarchy. When a hierarchy of clusters like this is created the user can determine what the right number of clusters is that adequately summarizes the data while still providing useful information (at the other extreme a single cluster containing all the records is a great summarization but does not contain enough specific information to be useful).

This hierarchy of clusters is created through the algorithm that builds the clusters. There are two main types of hierarchical clustering algorithms:

Agglomerative - Agglomerative clustering techniques start with as many clusters as there are records where each cluster contains just one record. The clusters that are nearest each other are merged together to form the next

largest cluster. This merging is continued until a hierarchy of clusters is built with just a single cluster containing all the records at the top of the hierarchy.

Divisive - Divisive clustering techniques take the opposite approach from agglomerative techniques. These techniques start with all the records in one cluster and then try to split that cluster into smaller pieces and then in turn to try to split those smaller pieces.

Of the two the agglomerative techniques are the most commonly used for clustering and have more algorithms developed for them. We'll talk about these in more detail in the next section. The non-hierarchical techniques in general are faster to create from the historical database but require that the user make some decision about the number of clusters desired or the minimum "nearness" required for two records to be within the same cluster. These non-hierarchical techniques often times are run multiple times starting off with some arbitrary or even random clustering and then iteratively improving the clustering by shuffling some records around. Or these techniques sometimes create clusters that are created with only one pass through the database adding records to existing clusters when they exist and creating new clusters when no existing cluster is a good candidate for the given record. Because the definition of which clusters are formed can depend on these initial choices of which starting clusters should be chosen or even how many clusters these techniques can be less repeatable than the hierarchical techniques and can sometimes create either too many or too few clusters because the number of clusters is predetermined by the user not determined solely by the patterns inherent in the database.

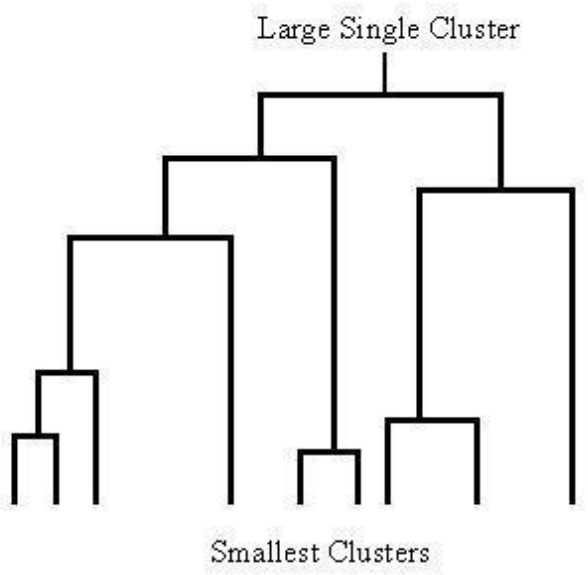


Figure 1.5 ***Diagram showing a hierarchy of clusters. Clusters at the lowest level are merged together to form larger clusters at the next level of the hierarchy.***

Non-Hierarchical Clustering

There are two main non-hierarchical clustering techniques. Both of them are very fast to compute on the database but have some drawbacks. The first are the single pass methods. They derive their name from the fact that the database must only be passed through once in order to create the clusters (i.e. each record is only read from the database once). The other class of techniques are called reallocation methods. They get their name from the movement or "reallocation" of records from one cluster to another in order to create better clusters. The reallocation techniques do use multiple passes through the database but are relatively fast in comparison to the hierarchical techniques.

Hierarchical Clustering

Hierarchical clustering has the advantage over non-hierarchical techniques in that the clusters are defined solely by the data (not by the users predetermining the number of clusters) and that the number of clusters can be increased or decreased by simple moving up and down the hierarchy.

The hierarchy is created by starting either at the top (one cluster that includes all records) and subdividing (divisive clustering) or by starting at the bottom with as many clusters as there are records and merging (agglomerative clustering). Usually the merging and subdividing are done two clusters at a time.

The main distinction between the techniques is their ability to favor long, scraggly clusters that are linked together record by record, or to favor the detection of the more classical, compact or spherical cluster that was shown at the beginning of this section. It may seem strange to want to form these long snaking chain like clusters, but in some cases they are the patters that the user would like to have detected in the database. These are the times when the underlying space looks quite different from the spherical clusters and the clusters that should be formed are not based on the distance from the center of the cluster but instead based on the records being "linked" together. Consider the example shown in Figure 1.6 or in Figure 1.7. In these cases there are two clusters that are not very spherical in shape but could be detected by the single link technique.

When looking at the layout of the data in Figure1.6 there appears to be two relatively flat clusters running parallel to each along the income axis. Neither the complete link nor Ward's method would, however, return these two clusters to the user. These techniques rely on creating a "center" for each cluster and picking these centers so that they average distance of each record from this center is minimized. Points that are very distant from these centers would necessarily fall into a different cluster.

.

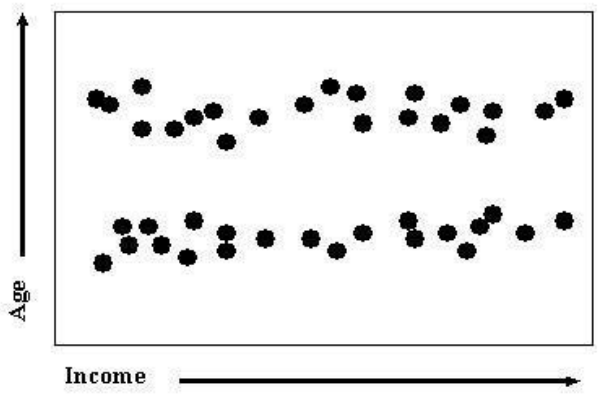


Figure 1.6 ***an example of elongated clusters which would not be recovered by the complete link or Ward's methods but would be by the single-link method.***

Density-Based Clustering Methods

Clustering based on density (local cluster criterion), such as density-connected points

Major features:

- Discover clusters of arbitrary shape
- Handle noise
- One scan
- Need density parameters as termination condition

Several interesting studies:

- DBSCAN: Ester, et al. (KDD'96)
- OPTICS: Ankerst, et al (SIGMOD'99).
- DENCLUE: Hinneburg & D. Keim (KDD'98)
- CLIQUE: Agrawal, et al. (SIGMOD'98)

Density-Based Clustering: Background

Two parameters*:*

- *Eps*: Maximum radius of the neighbour hood
- *MinPts*: Minimum number of points in an Eps-neighbour hood of that point

NEps(p):*{q belongs to D | dist(p,q) <= Eps}*

Directly density-reachable: A point *p* is directly density-reachable from a point *q* wrt. *Eps, MinPts* if

- 1) *p* belongs to *NEps(q)*
- 2) core point condition:

|NEps (q)| >= MinPts

Density-Based Clustering: Background (II)

Density-reachable:

- A point *p* is density-reachable from a point *q* wrt. *Eps, MinPts* if there is a chain of points *p1, …, pn, p1 = q, pn = p* such that *pi+1* is directly density-reachable from *pi*

Density-connected

- A point *p* is density-connected to a point *q* wrt. *Eps, MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* wrt. *Eps* and *MinPts*.

DBSCAN: Density Based Spatial Clustering of Applications with Noise

Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points

Discovers clusters of arbitrary shape in spatial databases with noise.

Outlier Analysis

What Is Outlier Discovery?

What are outliers?

- The set of objects are considerably dissimilar from the remainder of the data
- Example: Sports: Michael Jordon, Wayne Gretzky, ...

Problem

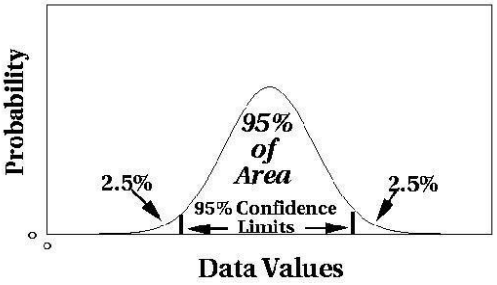
- Find top n outlier points

Applications:

- Credit card fraud detection
- Telecom fraud detection
- Customer segmentation

– Medical analysis

Outlier Discovery: Statistical Approaches



Assume a model underlying distribution that generates data set (e.g. normal distribution)

Use discordance tests depending on

– data distribution
– distribution parameter (e.g., mean, variance)
– number of expected outliers

Drawbacks

– most tests are for single attribute
– In many cases, data distribution may not be known

Outlier Discovery: Distance-Based Approach

Introduced to counter the main limitations imposed by statistical methods

– We need multi-dimensional analysis without knowing data distribution.

Distance-based outlier: A DB(p, D)-outlier is an object O in a dataset T such that at least a fraction p of the objects in T lies at a distance greater than D from O

Algorithms for mining distance-based outliers

– Index-based algorithm
– Nested-loop algorithm
– Cell-based algorithm

Outlier Discovery: Deviation-Based Approach

• Identifies outliers by examining the main characteristics of objects in a group

Objects that "deviate" from this description are considered outliers

sequential exception technique

– simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects

OLAP data cube technique

– uses data cubes to identify regions of anomalies in large multidimensional data