UNIT-I: Introduction to Data Warehousing and OLAP
Introduction to Data Warehousing: Concepts and Applications, Data Warehouse Architecture:
Components and 3-Tier Architecture, ETL Process: Extraction, Transformation, Loading, OLAP:
Multidimensional Data Model, OLAP Operations, Star, Snowflake, and Fact Constellation Schemas,
Data Cubes and Roll-Up/Drill-Down Analysis

What is Data and Information?

Data is an individual unit that contains raw materials which do not carry any specific meaning.

Information is a group of data that collectively carries a logical meaning.

Data doesn't depend on information.

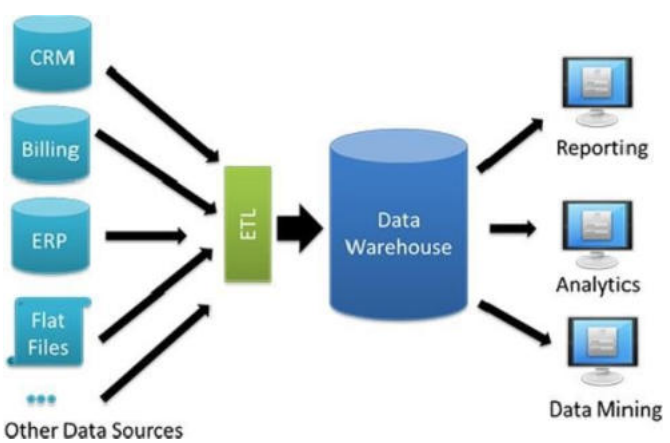
Information depends on data.

Data is measured in bits and bytes.

Information is measured in meaningful units like time, quantity, etc.

Data Warehouse:

Data warehouse is like a relational database designed for analytical needs. It functions on the basis of OLAP (Online Analytical Processing). It is a central location where consolidated data from multiple locations (databases) are stored.



What is Data warehousing?

Data warehousing is the act of organizing & storing data in a way so as to make its retrieval efficient and insightful. It is also called as the process of transforming data into information.

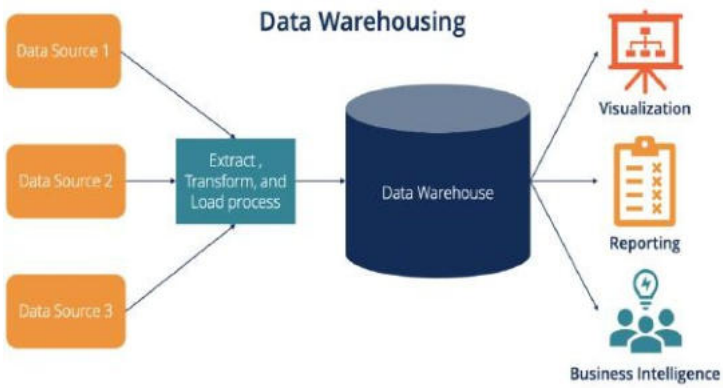


Fig: Data warehousing Process

Data Warehouse Characteristics:

A Data warehouse is a subject-oriented, integrated, time variant and non-volatile collection of data in support of management's decision making process.

Subject-oriented:

A Data warehouse can be used to analyze a particular subject area

Ex: "Sales" can be particular subject

Integrated:

A Data warehouse integrates data from multiple data sources.

Time Variant:

Historical data is kept in a data warehouse.

Ex: one can retrieve data from 3 months, 6months, 12 months or even older data from a data warehouse. This contrasts with a transactions system, where often only the most recent data is kept.

Non-Volatile:

Once data is in the data warehouse, it will not change. So historical data in a data warehouse should never be altered.

Data warehouse Architecture:

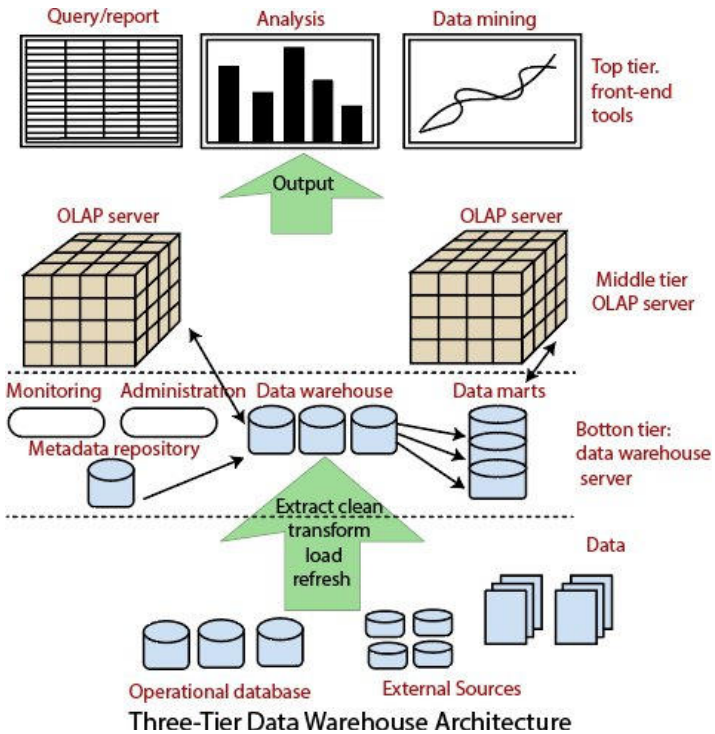


Fig: Data ware housing Architecture

Data warehouses often adopt a three-tier architecture

- The bottom tier is a **warehouse database server** that is almost always a relational database system. Back-end tools and utilities are used to feed the data into the bottom tier from operational database or other external sources. These tools and utilities perform data extraction, cleaning and transformation(ex. To merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse. The data are extracted using application program interfaces known as gateways.A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server.

  Examples of gateways include **ODBC**(Open Database Connection) and **OLEDB**(Open Linking and Embedding for Databases) by Microsoft and **JDBC**(Java Database Connection).This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

The middle tier is an **OLAP** server that is typically implemented using either

(a) a relational OLAP**(ROLAP)** model, that is an extended relational DBMS that maps operations on multidimensional data to standard relational operations, or

(b) a multidimensional OLAP**(MOLAP)** model that is a special-purpose server that directly implements multidimensional data and operations.

The top tier is a front end client layer, which contains query and reporting tools, analysis tools and data mining tools(ex: trend analysis, prediction….)

Multi-dimensional Data Model:

- A multidimensional model views data in the form of a data-cube.
- When data is grouped or combined in multidimensional matrices called Data Cubes.
- A data cube enables data to be modeled and viewed in multiple dimensions.

It is defined by dimensions and facts.

- A multidimensional data model is organized around a central theme, for example, sales. This theme is represented by a fact table. Facts are numerical measures.
- The fact table contains the names of the facts or measures of the related dimensional tables.

  FACT VS DIMENSION

  Fact/Measure(What you want to analyse is your fact)

  Ex: What is My sales, What is my profit, What is my custmes preferences.

  Dimensions(By Which you want to Analyze is your Dimensions)

  Sales By Location/Product/Period

  Total Profit By Location/Product/Period

- These Dimensions allow the store to keep track of things like monthly sales of items and branches and locations at which the items were sold.
- Each dimension may have a table associated with it called a **dimension table,** which further describes the dimension.

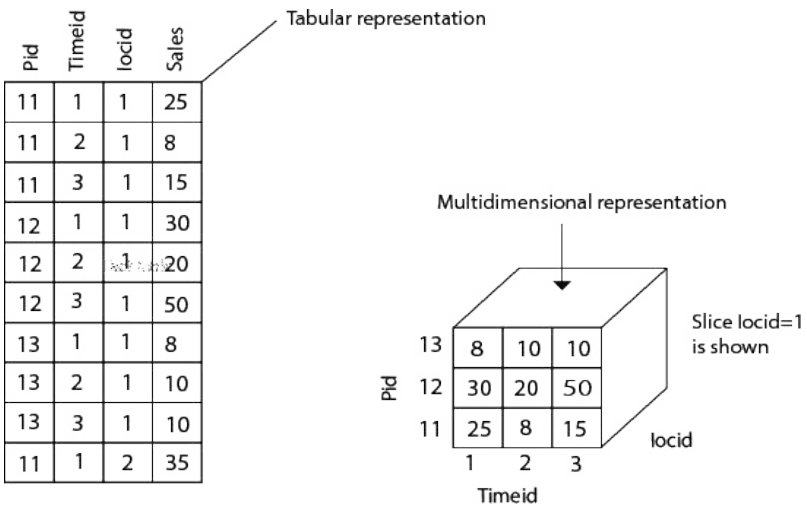


Fig: Multidimensional Representation

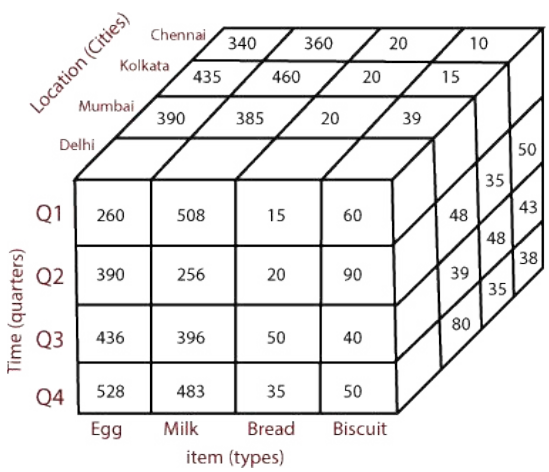
- Consider the data of a shop for items sold per quarter in the city of Delhi. The data is shown in the table.
- In this 2D representation, the sales for Delhi are shown for the time dimension (organized in quarters) and the item dimension (classified according to the types of an item sold).
- The fact or measure displayed in rupee_sold (in thousands).

Location="Delhi"				
Time (quarter)	item (type)			
	Egg	Milk	Bread	Biscuit
Q1	260	508	15	60
Q2	390	256	20	90
Q3	436	396	50	40
Q4	528	483	35	50

Now, if we want to view the sales data with a third dimension, For example, suppose the data according to time and item, as well as the location is considered for the cities Chennai, Kolkata, Mumbai, and Delhi. These 3D data are shown in the table. The 3D data of the table are represented as a series of 2D tables.

	Location="Chennai"				Location="Kolkata"				Location="Mumbai"				Location="Delhi"			
	item				item				item				item			
Time	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit
Q1	340	360	20	10	435	460	20	15	390	385	20	39	260	508	15	60
Q2	490	490	16	50	389	385	45	35	463	366	25	48	390	256	20	90
Q3	680	583	46	43	684	490	39	48	568	594	36	39	436	396	50	40
Q4	535	694	39	38	335	365	83	35	338	484	48	80	528	483	35	50

Conceptually, it may also be represented by the same data in the form of a 3D data cube, as shown in fig:



What is Schema?

- Schema is a logical description of the entire database.
- It includes the name and description of records of all record types including all associated data-Items and aggregates.
- Much like a database, a data warehouse also requires to maintain a schema.
- A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema.
- Modeling data warehouses: dimensions & measures

- Star schema: A fact table in the middle connected to a set of dimension tables
- Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
- Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

Star Schema:

- A star schema is the elementary form of a dimensional model, in which data are organized into **facts** and **dimensions**.
- A fact is an event that is counted or measured, such as a sale or log in. A dimension includes reference data about the fact, such as date, item, or customer.
- A star schema is a relational schema where a relational schema whose design represents a multidimensional data model.
- The star schema is the explicit data warehouse schema. It is known as **star schema** because the entity-relationship diagram of this schemas simulates a star, with points, diverge from a central table.
- The center of the schema consists of a large fact table, and the points of the star are the dimension tables.

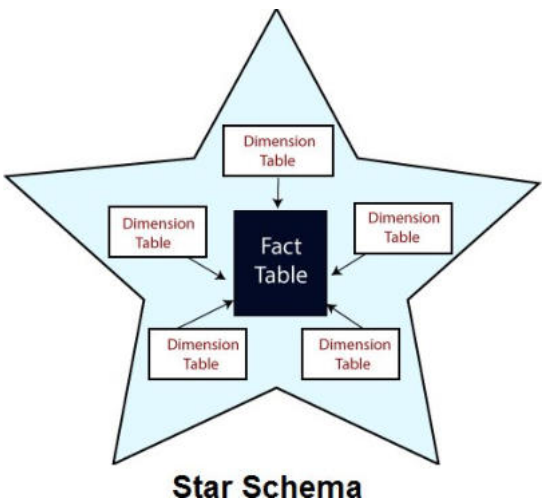
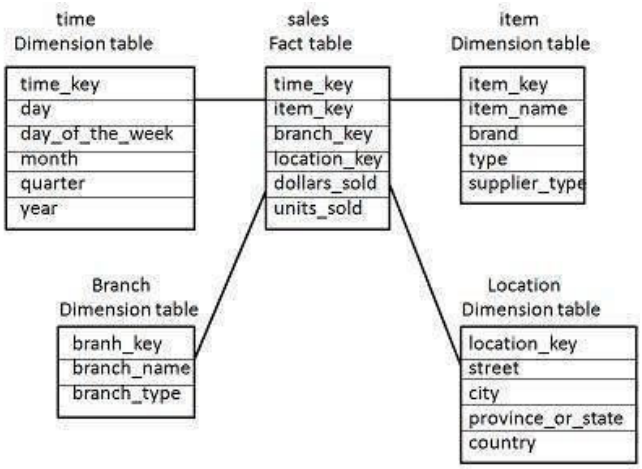


Fig: Star Schema Representation

Star Schema:

- Each dimension in a star schema is represented with only one-dimension table.
- This dimension table contains the set of attributes.
- The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location.



- There is a fact table at the center. It contains the keys to each of four dimensions.
- The fact table also contains the attributes, namely dollars sold and units sold.
- Each dimension has only one dimension table and each table holds a set of attributes. For example, the location dimension table contains the attribute set {location_key, street, city, province_or_state,country}. This constraint may cause data redundancy. For example, "Vancouver" and "Victoria" both the cities are in the Canadian province of British Columbia. The entries for such cities may cause data redundancy along the attributes province_or_state and country.

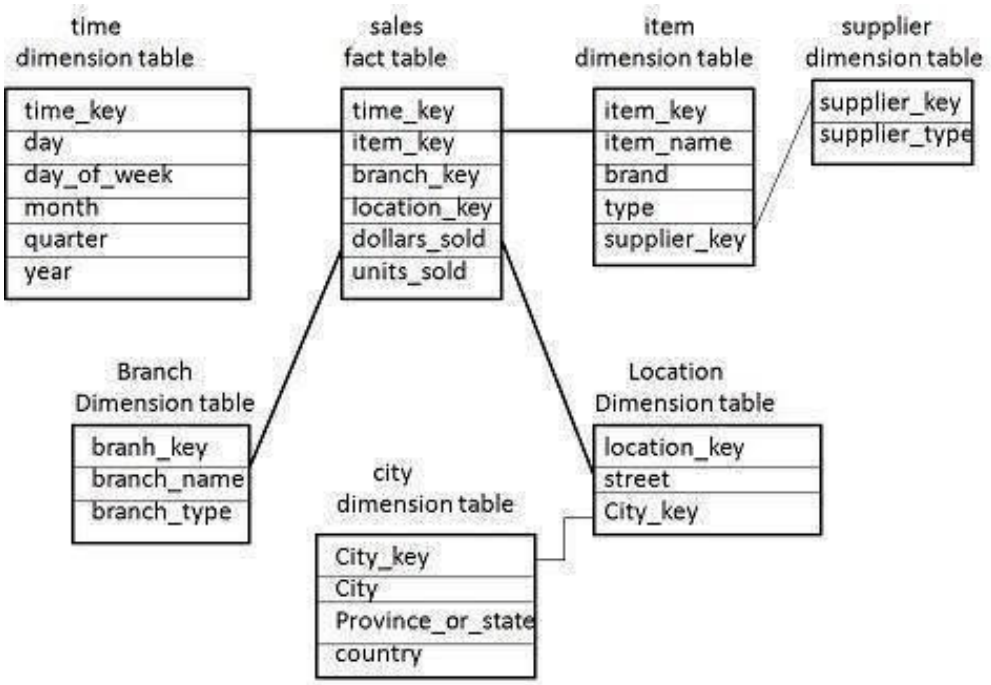
Characteristics of Star Schema:

- Every dimension in a star schema is represented with the only one-dimension table.
- The dimension table should contain the set of attributes.
- The dimension table is joined to the fact table using a foreign key
- The dimension table are not joined to each other
- Fact table would contain key and measure
- The Star schema is easy to understand and provides optimal disk usage.

- The dimension tables are not normalized. For instance, in the above figure, Country_ID does not have Country lookup table as an OLTP design would have.
- The schema is widely supported by BI Tools.
- **Advantages:**
- (i) Simplest and Easiest
- (ii) It optimizes navigation through database
- (iii) Most suitable for Query Processing

Snowflake Schema:

- Some dimension tables in the Snowflake schema are normalized.
- The normalization splits up the data into additional tables.
- Unlike Star schema, the dimensions table in a snowflake schema are normalized.
- For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.



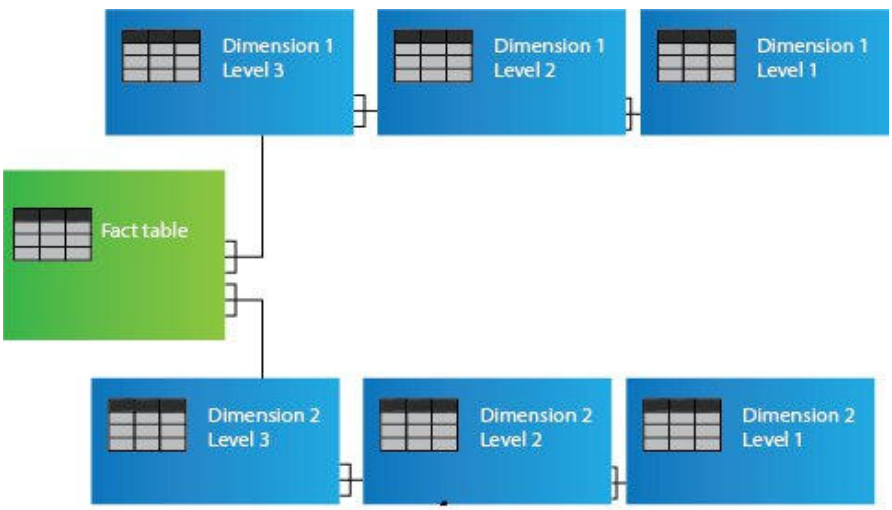
- Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.
- The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and supplier_type.

Note : Due to normalization in the Snowflake schema, the redundancy is reduced and therefore, it becomes easy to maintain and the save storage space.

Fig: Snowflake image

A snowflake schemas can have any number of dimension, and each dimension can have any number of levels.

The following diagram shows a snowflake schema with two dimensions, each having three levels.



Snowflake Schema

Advantages:

(i) Less redundancies due to normalization Dimension Tables.

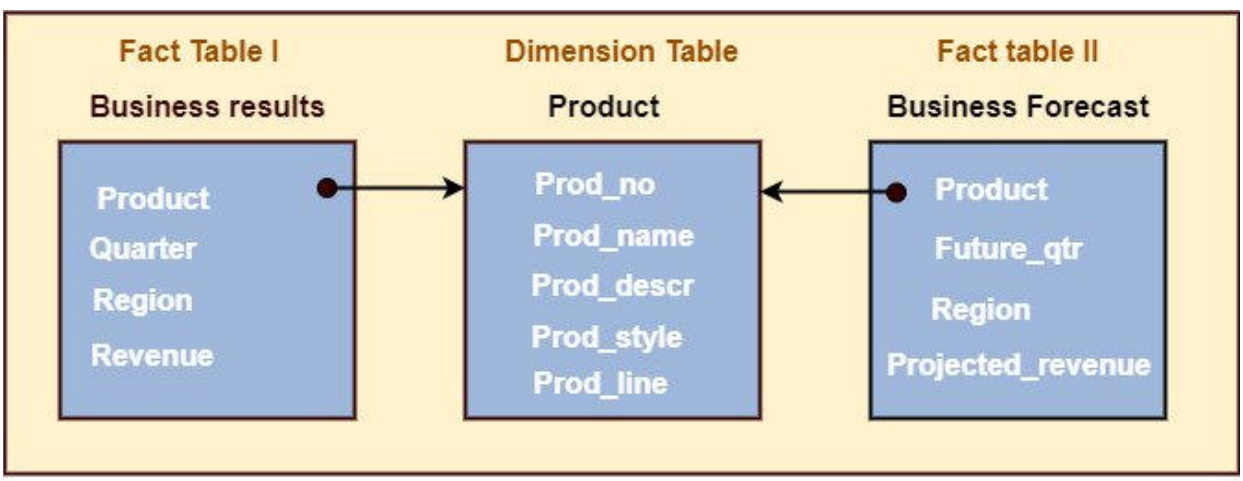
(ii) Dimension Tables are easier to update.

Disadvantages:

It is complex schema when compared to star schema.

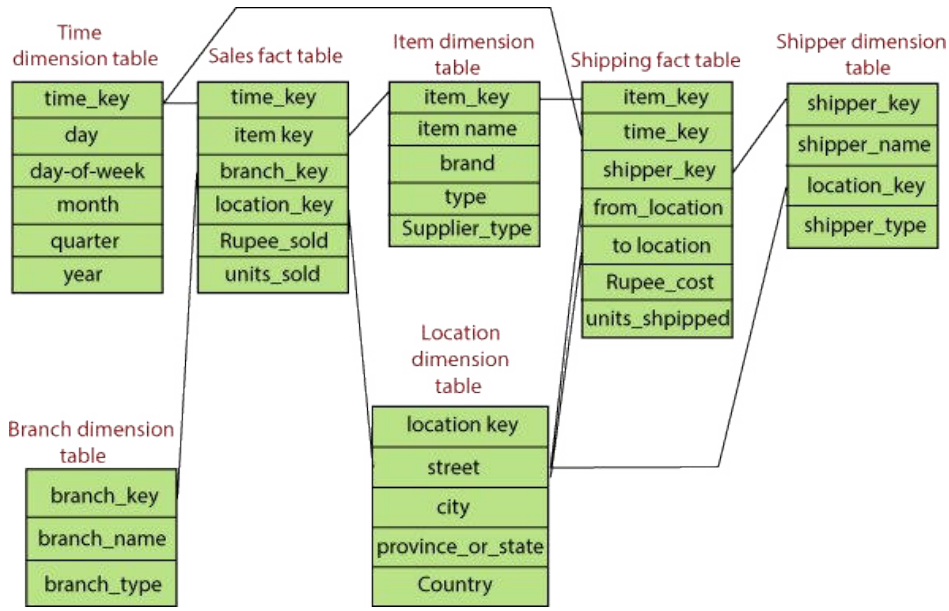
Fact Constellation Schema:

- A Fact constellation means two or more fact tables sharing one or more dimensions. It is also called **Galaxy schema**.
- Fact Constellation Schema describes a logical structure of data warehouse or data mart. Fact Constellation Schema can design with a collection of de-normalized FACT, Shared, and Conformed Dimension tables.



FACT Constellation Schema

A fact constellation schema is shown in the figure below.



- This schema defines two fact tables, sales, and shipping. Sales are treated along four dimensions, namely, time, item, branch, and location.
- The schema contains a fact table for sales that includes keys to each of the four dimensions, along with two measures: Rupee_sold and units_sold.
- The shipping table has five dimensions, or keys: item_key, time_key, shipper_key, from_location, and to_location, and two measures: Rupee_cost and units_shipped.
- It is also possible to share dimension tables between fact tables. For example, time, item, and location dimension tables are shared between the sales and shipping fact table.

Disadvantages:

(i) Complex due to multiple fact tables

(ii) It is difficult to manage

(iii) Dimension Tables are very large.

OLAP OPERATIONS:

- In the multidimensional model, the records are organized into various dimensions, and each dimension includes multiple levels of abstraction described by concept hierarchies.

- This organization support users with the flexibility to view data from various perspectives.
- A number of OLAP data cube operation exist to demonstrate these different views, allowing interactive queries and search of the record at hand. Hence, OLAP supports a user-friendly environment for interactive data analysis.
- Consider the OLAP operations which are to be performed on multidimensional data.
- The data cubes for sales of a shop. The cube contains the dimensions, location, and time and item, where the **location** is aggregated with regard to city values, **time** is aggregated with respect to quarters, and an **item** is aggregated with respect to item types.

OLAP having 5 different operations

(i) Roll-up

(ii) Drill-down

(iii) Slice

(iv) Dice

(v) Pivot

Roll-up:

- The roll-up operation performs aggregation on a data cube, by climbing down concept hierarchies, i.e., dimension reduction. Roll-up is like **zooming-out** on the data cubes.
- **It is also known as drill-up or aggregation operation**
- Figure shows the result of roll-up operations performed on the dimension location. The hierarchy for the location is defined as the Order Street, city, province, or state, country.
- The roll-up operation aggregates the data by ascending the location hierarchy from the level of the city to the level of the country.
- When a roll-up is performed by dimensions reduction, one or more dimensions are removed from the cube.
- For example, consider a sales data cube having two dimensions, location and time. Roll-up may be performed by removing, the time dimensions, appearing in an aggregation of the total sales by location, relatively than by location and by time.

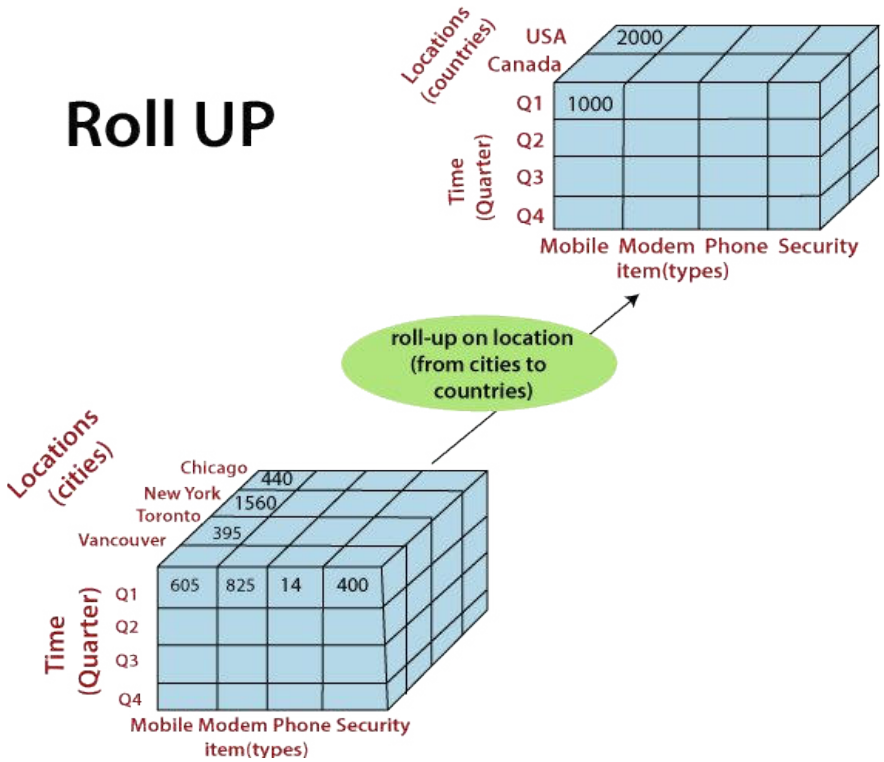


Fig: Roll-up operation on Data Cube

Drill-Down

- The drill-down operation is the reverse operation of **roll-up**.
- **It is also called roll-down operation.**
- Drill-down is like **zooming-in** on the data cube.
- It navigates from less detailed record to more detailed data. Drill-down can be performed by either **stepping down** a concept hierarchy for a dimension or adding additional dimensions.
- Figure shows a drill-down operation performed on the dimension time by stepping down a concept hierarchy which is defined as day, month, quarter, and year.
- Drill-down appears by descending the time hierarchy from the level of the quarter to a more detailed level of the month.
- Because a drill-down adds more details to the given data, it can also be performed by adding a new dimension to a cube.

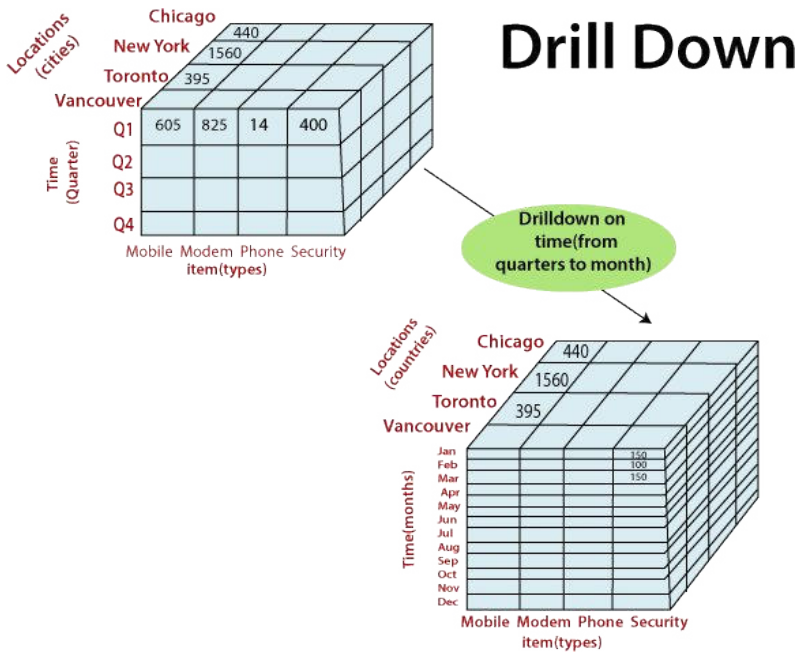


Fig: Drill-down operation

Slice:

- A **slice** is a subset of the cubes corresponding to a single value for one or more members of the dimension.
- The slice operation provides a new sub cube from one particular dimension in a given cube.
- For example, a slice operation is executed when the customer wants a selection on one dimension of a three-dimensional cube resulting in a two-dimensional site. So, the Slice operations perform a selection on one dimension of the given cube, thus resulting in a sub cube.
- Here Slice is functioning for the dimensions "time" using the criterion time = "Q1".
- It will form a new sub-cubes by selecting one or more dimensions.

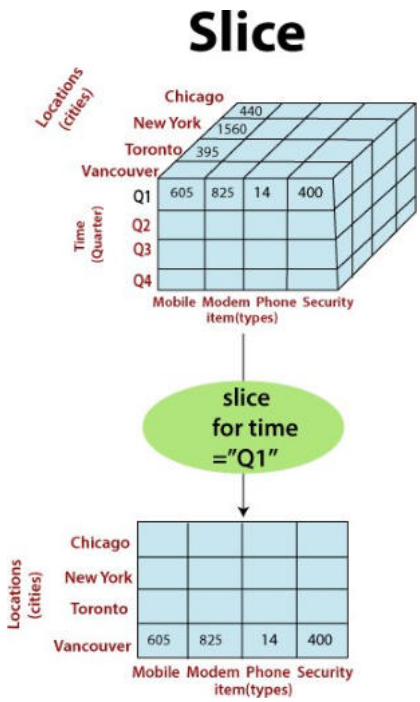


Fig: Slice operation

Dice:

- The dice operation describes a sub cube by operating a selection on two or more dimension.

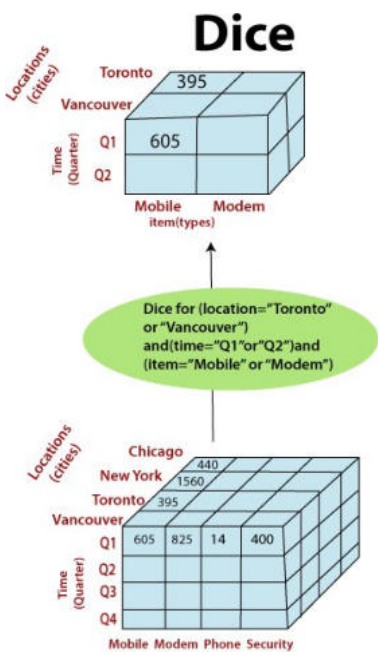


Fig: Dice operation

- The dice operation on the cubes based on the following selection criteria involves three dimensions.

  (location = "Toronto" or "Vancouver")

  (time = "Q1" or "Q2")

  (item =" Mobile" or "Modem")

Pivot:

- The pivot operation is also called a rotation.
- Pivot is a visualization operations which rotates the data axes in view to provide an alternative presentation of the data.
- It may contain swapping the rows and columns or moving one of the row-dimensions into the column dimensions.

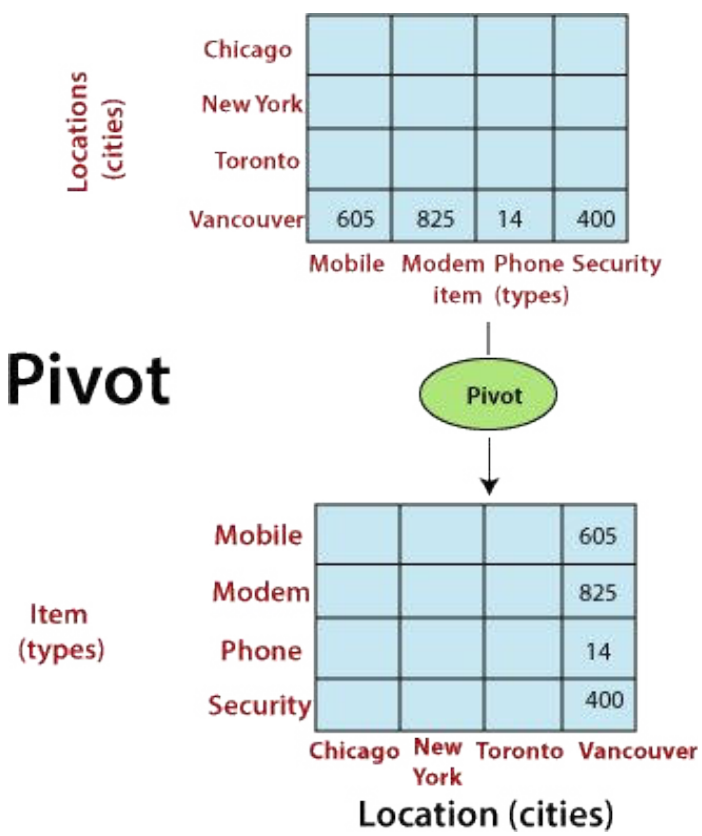


Fig: Pivot Operation

Differences between OLTP and OLAP:

Feature	OLTP	OLAP
Characteristic	operational processing	informational processing
Orientation	Transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements, decision support
DB design	ER 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
Number of records accessed	Tens	millions
Number 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

Parallel DBMS Vendors:

What is a DBMS vendor?

(Data base Management System)Software that controls the organization, storage, retrieval, security and integrity of data in a database.

The major DBMS vendors are Oracle, IBM, Microsoft and Sybase (see Oracle Database, DB2, SQL Server and ASE).

DBMS	Vendor	Type	Primary Market
Access (Jet, MSDE)	Microsoft	R	Desktop
Adabas D	Software AG	R	Enterprise
Adaptive Server Anywhere	Sybase	R	Mobile/Embedded
Adaptive Server Enterprise	Sybase	R	Enterprise
Advantage Database Server	Extended Systems	R	Mobile/Enterprise
Datacom	Computer Associates	R	Enterprise
DB2 Everyplace	IBM	R	Mobile
Filemaker	FileMaker Inc.	R	Desktop
IDMS	Computer Associates	R	Enterprise
Ingres ii	Computer Associates	R	Enterprise
Interbase	Inprise (Borland)	R	Open Source

MySQL	Freeware	R	Open Source
NonStop SQL	Tandem	R	Enterprise
Pervasive.SQL 2000 (Btrieve)	Pervasive Software	R	Embedded
Pervasive.SQL Workgroup	Pervasive Software	R	Enterprise (Windows 32)
Progress	Progress Software	R	Mobile/Embedded
Quadbase SQL Server	Quadbase Systems, Inc.	Relational	Enterprise
R:Base	R:Base Technologies	Relational	Enterprise
Rdb	Oracle	R	Enterprise
Red Brick	Informix (Red Brick)	R	Enterprise (Data Warehousing)
SQL Server	Microsoft	R	Enterprise
SQLBase	Centura Software	R	Mobile/Embedded
SUPRA	Cincom	R	Enterprise
Teradata	NCR	R	VLDB (Data Warehousing)
YARD-SQL	YARD Software Ltd.	R	Enterprise
TimesTen	TimesTen Performance Software	R	In-Memory
Adabas	Software AG	XR	Enterprise
Model 204	Computer Corporation of America	XR	VLDB
UniData	Informix (Ardent)	XR	Enterprise
UniVerse	Informix (Ardent)	XR	Enterprise
Cache'	InterSystems	OR	Enterprise
Cloudscape	Informix	OR	Mobile/Embedded
DB2	IBM	OR	Enterprise/ VLDB

Informix Dynamic Server 2000	Informix	OR	Enterprise
Informix Extended Parallel Server	Informix	OR	VLDB (Data Warehousing)
Oracle Lite	Oracle	OR	Mobile
Oracle 8I	Oracle	OR	Enterprise
PointBase Embedded	PointBase	OR	Embedded
PointBase Mobile	PointBase	OR	Mobile
PointBase Network Server	PointBase	OR	Enterprise
PostgreSQL	Freeware	OR	Open Source
UniSQL	Cincom	OR	Enterprise
Jasmine ii	Computer Associates	OO	Enterprise
Object Store	Exceleron	OO	Enterprise
Objectivity DB	Objectivity	OO	VLDB (Scientific)
POET Object Server Suite	Poet Software	OO	Enterprise
Versant	Versant Corporation	OO	Enterprise
Raima Database Manager	Centura Software	RN	Mobile/Embedded
Velocis	Centura Software	RN	Enterprise/ Embedded
Db.linux	Centura Software	RNH	Open Source/Mobile/Embedded
Db.star	Centura Software	RNH	Open Source/Mobile/Embedded

Types of Data Warehouse:

There are three main types of DWH. Each has its specific role in data management operations.

1. Enterprise Data Warehouse

Enterprise data warehouse (EDW) serves as a central or main database to facilitate decision-making throughout the enterprise. Key benefits of having an EDW include access to cross-organizational information, the ability to run complex queries, and the enablement of enriched, far-sighted insights for data-driven decisions and early risk assessment.

2. ODS (Operational Data Store)

In ODS, the DWH refreshes in real-time. Therefore, organizations often used it for routine enterprise activities, such as storing records of the employees. Business processes also use ODS as a source for providing data to the EDW.

3. Data Mart

It is a subset of a DWH that supports a particular department, region, or business unit. Consider this: You have multiple departments, including sales, marketing, product development, etc. Each department will have a central repository where it stores data. This repository is called a data mart. The EDW stores the data from the data mart in the ODS on a daily/weekly (or as configured) basis. The ODS acts as a staging area for data integration. It then sends the data to the EDW to store it and use it for BI purposes.

DATAWAREHOUSE COMPONENTS:

The data warehouse is based on an RDBMS server which is a central information repository that is surrounded by some key components to make the entire environment functional, manageable and accessible

There are mainly five components of Data Warehouse:

DATA WAREHOUSE DATABASE:

The central database is the foundation of the data warehousing environment. This database is implemented on the RDBMS technology. Although, this kind of implementation is constrained

by the fact that traditional RDBMS system is optimized for transactional database processing and not for data warehousing. For instance, ad-hoc query, multi-table joins, aggregates are resource intensive and slow down performance.

Hence, alternative approaches to Database are used as listed below-

In a data warehouse, relational databases are deployed in parallel to allow for scalability. Parallel relational databases also allow shared memory or shared nothing model on various multiprocessor configurations or massively parallel processors.

New index structures are used to bypass relational table scan and improve speed.

Use of multidimensional database (MDDBs) to overcome any limitations which are placed because of the relational data model. Example: Essbase from Oracle.

SOURCING, ACQUISITION, CLEAN-UP AND TRANSFORMATION TOOLS (ETL):

The data sourcing, transformation, and migration tools are used for performing all the conversions, summarizations, and all the changes needed to transform data into a unified format in the datawarehouse. They are also called Extract, Transform and Load (ETL) Tools.

These Extract, Transform, and Load tools may generate cron jobs, background jobs, Cobol programs, shell scripts, etc. that regularly update data in datawarehouse. These tools are also helpful to maintain the Metadata.

These ETL Tools have to deal with challenges of Database & Data heterogeneity.

METADATA:

The name Meta Data suggests some high- level technological concept. However, it is quite simple. Metadata is data about data which defines the data warehouse. It is used for building, maintaining and managing the data warehouse.

In the Data Warehouse Architecture, meta-data plays an important role as it specifies the source, usage, values, and features of data warehouse data. It also defines how data can be changed and processed. It is closely connected to the data warehouse.

QUERY TOOLS:

One of the primary objects of data warehousing is to provide information to businesses to make strategic decisions. Query tools allow users to interact with the data warehouse system.

These tools fall into four different categories:

Query and reporting tools

Application Development tools

Data mining tools

OLAP tools

Characteristics of OLAP:

The main characteristics of OLAP are as follows:

Multidimensional conceptual view: OLAP systems let business users have a dimensional and logical view of the data in the data warehouse. It helps in carrying slice and dice operations.

Multi-User Support: Since the OLAP techniques are shared, the OLAP operation should provide normal database operations, containing retrieval, update, adequacy control, integrity, and security.

Accessibility: OLAP acts as a mediator between data warehouses and front-end. The OLAP operations should be sitting between data sources (e.g., data warehouses) and an OLAP front-end.

Storing OLAP results: OLAP results are kept separate from data sources.

Uniform documenting performance: Increasing the number of dimensions or database size should not significantly degrade the reporting performance of the OLAP system.

OLAP provides for distinguishing between zero values and missing values so that aggregates are computed correctly.

OLAP system should ignore all missing values and compute correct aggregate values.

OLAP facilitate interactive query and complex analysis for the users.

OLAP allows users to drill down for greater details or roll up for aggregations of metrics along a single business dimension or across multiple dimension.

OLAP provides the ability to perform intricate calculations and comparisons.

OLAP presents results in a number of meaningful ways, including charts and graphs.

OLAP Types:

Three types of OLAP servers are:-

1. Relational OLAP (ROLAP)
2. Multidimensional OLAP (MOLAP)
3. Hybrid OLAP (HOLAP)

1. Relational OLAP (ROLAP):

Relational On-Line Analytical Processing (ROLAP) work mainly for the data that resides in a relational database, where the base data and dimension tables are stored as relational tables. ROLAP servers are placed between the relational back-end server and client front-end tools. ROLAP servers use RDBMS to store and manage warehouse data, and OLAP middleware to support missing pieces.

Advantages of ROLAP:

1. ROLAP can handle large amounts of data.
2. Can be used with data warehouse and OLTP systems.

Disadvantages of ROLAP:

1. Limited by SQL functionalities.
2. Hard to maintain aggregate tables.

2. Multidimensional OLAP (MOLAP):

Multidimensional On-Line Analytical Processing (MOLAP) support multidimensional views of data through array-based multidimensional storage engines. With multidimensional data stores, the storage utilization may be low if the data set is sparse.

Advantages of MOLAP

1. Optimal for slice and dice operations.
2. Performs better than ROLAP when data is dense.
3. Can perform complex calculations.

Disadvantages of MOLAP

1. Difficult to change dimension without re-aggregation.

2. MOLAP can handle limited amount of data.

3. Hybrid OLAP (HOLAP):

Hybrid On-Line Analytical Processing (HOLAP) is a combination of ROLAP and MOLAP. HOLAP provide greater scalability of ROLAP and the faster computation of MOLAP.

Advantages of HOLAP

1. HOLAP provide advantages of both MOLAP and ROLAP.

2. Provide fast access at all levels of aggregation.

Disadvantages of HOLAP

1. HOLAP architecture is very complex because it support both MOLAP and ROLAP servers.

UNIT-II: Data Preprocessing and Data Understanding
Data Cleaning, Integration, and Transformation, Data Reduction Techniques and Feature Selection,
Handling Missing, Noisy, and Inconsistent Data, Data Discretization and Normalization, Measures of
Similarity and Dissimilarity, Exploratory Data Analysis and Visualization.

Data Preprocessing in Data Mining

Data preprocessing is the process of preparing raw data for analysis by cleaning and transforming it into a usable format. In data mining it refers to preparing raw data for mining by performing tasks like cleaning, transforming, and organizing it into a format suitable for mining algorithms.

- Goal is to improve the quality of the data.
- Helps in handling missing values, removing duplicates, and normalizing data.
- Ensures the accuracy and consistency of the dataset.

Steps in Data Preprocessing

Some key steps in data preprocessing are Data Cleaning, Data Integration, Data Transformation, and Data Reduction.

1. Data Cleaning: It is the process of identifying and correcting errors or inconsistencies in the dataset. It involves handling missing values, removing duplicates, and correcting incorrect or outlier data to ensure the dataset is accurate and reliable. Clean data is essential for effective analysis, as it improves the quality of results and enhances the performance of data models.

- **Missing Values:** This occur when data is absent from a dataset. You can either ignore the rows with missing data or fill the gaps manually, with the attribute mean, or by using the most probable value. This ensures the dataset remains accurate and complete for analysis.
- **Noisy Data:** It refers to irrelevant or incorrect data that is difficult for machines to interpret, often caused by errors in data collection or entry. It can be handled in several ways:
    - **Binning Method:** The data is sorted into equal segments, and each segment is smoothed by replacing values with the mean or boundary values.
    - **Regression:** Data can be smoothed by fitting it to a regression function, either linear or multiple, to predict values.
    - **Clustering:** This method groups similar data points together, with outliers either being undetected or falling outside the clusters. These techniques help remove noise and improve data quality.
- **Removing Duplicates:** It involves identifying and eliminating repeated data entries to ensure accuracy and consistency in the dataset. This process prevents errors and ensures reliable analysis by keeping only unique records.

2. Data Integration: It involves merging data from various sources into a single, unified dataset. It can be challenging due to differences in data formats, structures, and meanings. Techniques like record linkage and data fusion help in combining data efficiently, ensuring consistency and accuracy.

- **Record Linkage** is the process of identifying and matching records from different datasets that refer to the same entity, even if they are represented differently. It helps in combining data from various sources by finding corresponding records based on common identifiers or attributes.
- **Data Fusion** involves combining data from multiple sources to create a more comprehensive and accurate dataset. It integrates information that may be inconsistent or incomplete from different sources, ensuring a unified and richer dataset for analysis.

3. Data Transformation: It involves converting data into a format suitable for analysis. Common techniques include normalization, which scales data to a common range; standardization, which adjusts data to have zero mean and unit variance; and discretization, which converts continuous

data into discrete categories. These techniques help prepare the data for more accurate analysis.

- **Data Normalization**: The process of scaling data to a common range to ensure consistency across variables.
- **Discretization**: Converting continuous data into discrete categories for easier analysis.
- **Data Aggregation**: Combining multiple data points into a summary form, such as averages or totals, to simplify analysis.
- **Concept Hierarchy Generation**: Organizing data into a hierarchy of concepts to provide a higher-level view for better understanding and analysis.

4. Data Reduction: It reduces the dataset's size while maintaining key information. This can be done through feature selection, which chooses the most relevant features, and feature extraction, which transforms the data into a lower-dimensional space while preserving important details. It uses various reduction techniques such as,

- **Dimensionality Reduction (e.g., Principal Component Analysis)**: A technique that reduces the number of variables in a dataset while retaining its essential information.
- **Numerosity Reduction**: Reducing the number of data points by methods like sampling to simplify the dataset without losing critical patterns.
- **Data Compression**: Reducing the size of data by encoding it in a more compact form, making it easier to store and process.

Uses of Data Preprocessing

Data preprocessing is utilized across various fields to ensure that raw data is transformed into a usable format for analysis and decision-making. Here are some key areas where data preprocessing is applied:

1. Data Warehousing: In data warehousing, preprocessing is essential for cleaning, integrating, and structuring data before it is stored in a centralized repository. This ensures the data is consistent and reliable for future queries and reporting.

2. Data Mining: Data preprocessing in data mining involves cleaning and transforming raw data to make it suitable for analysis. This step is crucial for identifying patterns and extracting insights from large datasets.

3. Machine Learning: In machine learning, preprocessing prepares raw data for model training. This includes handling missing values, normalizing features, encoding categorical variables, and splitting datasets into training and testing sets to improve model performance and accuracy.

4. Data Science: Data preprocessing is a fundamental step in data science projects, ensuring that the data used for analysis or building predictive models is clean, structured, and relevant. It enhances the overall quality of insights derived from the data.

5. Web Mining: In web mining, preprocessing helps analyze web usage logs to extract meaningful user behavior patterns. This can inform marketing strategies and improve user experience through personalized recommendations.
6. Business Intelligence (BI): Preprocessing supports BI by organizing and cleaning data to create dashboards and reports that provide actionable insights for decision-makers.
7. Deep Learning Purpose: Similar to machine learning, deep learning applications require preprocessing to normalize or enhance features of the input data, optimizing model training processes.

Advantages of Data Preprocessing

- **Improved Data Quality**: Ensures data is clean, consistent, and reliable for analysis.
- **Better Model Performance**: Reduces noise and irrelevant data, leading to more accurate predictions and insights.
- **Efficient Data Analysis**: Streamlines data for faster and easier processing.
- **Enhanced Decision-Making**: Provides clear and well-organized data for better business decisions.

Disadvantages of Data Preprocessing

- **Time-Consuming**: Requires significant time and effort to clean, transform, and organize data.
- **Resource-Intensive**: Demands computational power and skilled personnel for complex preprocessing tasks.
- **Potential Data Loss**: Incorrect handling may result in losing valuable information.
- **Complexity**: Handling large datasets or diverse formats can be challenging.
  Here are concise notes on the topics you specified:

  ---

1. **Feature Selection**

   Feature selection is the process of identifying and selecting a subset of relevant features (variables, predictors) for use in model construction. It enhances model performance by reducing overfitting, improving accuracy, and decreasing computational cost.

   Techniques:

   * **Filter Methods**: Evaluate the relevance of features using statistical tests (e.g., Chi-square test, correlation coefficient).

* **Wrapper Methods**: Evaluate feature subsets based on model performance (e.g., Recursive Feature Elimination).
* **Embedded Methods**: Perform feature selection during model training (e.g., Lasso, Decision Trees).

---

2. **Handling Missing, Noisy, and Inconsistent Data**

Data preprocessing is crucial to ensure the quality and reliability of data.

* **Missing Data**:

  * *Deletion*: Remove rows or columns with missing values.
  * *Imputation*: Fill missing values using mean, median, mode, or predictive models.

* **Noisy Data**:

  * *Smoothing*: Apply techniques like binning, regression, or clustering.
  * *Outlier Detection*: Identify and handle outliers using statistical methods or domain knowledge.

* **Inconsistent Data**:

  * *Standardization*: Ensure uniform formats (e.g., date formats, categorical values).
  * *Validation*: Cross-check data entries against predefined rules or external sources.

---

3. **Data Discretization and Normalization**

* **Discretization**: Converts continuous data into discrete bins or intervals.

  * *Equal-width*: Divides the range into equal intervals.
  * *Equal-frequency*: Divides data such that each bin has the same number of data points.

* **Normalization**: Scales data to a specific range, typically \[0, 1].

  * *Min-Max Scaling*: Rescales features to a fixed range.
  * *Z-score Normalization*: Centers data by subtracting the mean and dividing by the standard deviation.

---

4. **Measures of Similarity and Dissimilarity**

These measures quantify the similarity or dissimilarity between data points.

* **Cosine Similarity**: Measures the cosine of the angle between two vectors, indicating their orientation similarity.
* **Jaccard Index**: Measures similarity between finite sample sets, defined as the size of the intersection divided by the size of the union.
* **Dice-Sørensen Coefficient**: Similar to Jaccard but gives more weight to matches.
* **Simple Matching Coefficient (SMC)**: Measures similarity by comparing the number of matching attributes.

---

5. **Exploratory Data Analysis (EDA) and Visualization**

EDA involves analyzing data sets to summarize their main characteristics, often with visual methods.

* **Techniques**:

  * *Univariate Analysis*: Examine individual variables using histograms, box plots, and density plots.
  * *Bivariate Analysis*: Explore relationships between two variables using scatter plots and correlation matrices.
  * *Multivariate Analysis*: Analyze interactions among multiple variables using pair plots, heatmaps, and principal component analysis (PCA).

* **Visualization Tools**:

  * *Matplotlib*: Basic plotting library in Python.
  * *Seaborn*: Statistical data visualization built on Matplotlib.
  * *Tableau*: Interactive data visualization tool.
  * *Power BI*: Business analytics service for visualizing data.

If you need further details or examples on any of these topics, feel free to ask!

UNIT-III: Association Rule Mining and Classification

Basics of Association Rule Mining: Support, Confidence, Lift, Apriori Algorithm and FP-Growth Algorithm, Applications of Association Rule Mining, Classification Techniques: Decision Trees (ID3, C4.5), Bayesian Classifiers and Naïve Bayes, Rule-Based and Model-Based Classification.

Mining Frequent Patterns, Associations and Correlations – Mining Methods – Mining Various Kinds of Association Rules – Correlation Analysis – Constraint Based Association Mining – Classification and Prediction - Basic Concepts - Decision Tree Induction - Bayesian Classification – Rule Based Classification – Classification by Back propagation – Support Vector Machines – Associative Classification – Lazy Learners – Other Classification Methods – Prediction

---

Association Mining

- 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

- $A \rightarrow C$ (50%, 66.6%)
- $C \rightarrow A$ (50%, 100%)

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

Association Rule Mining: A Road Map

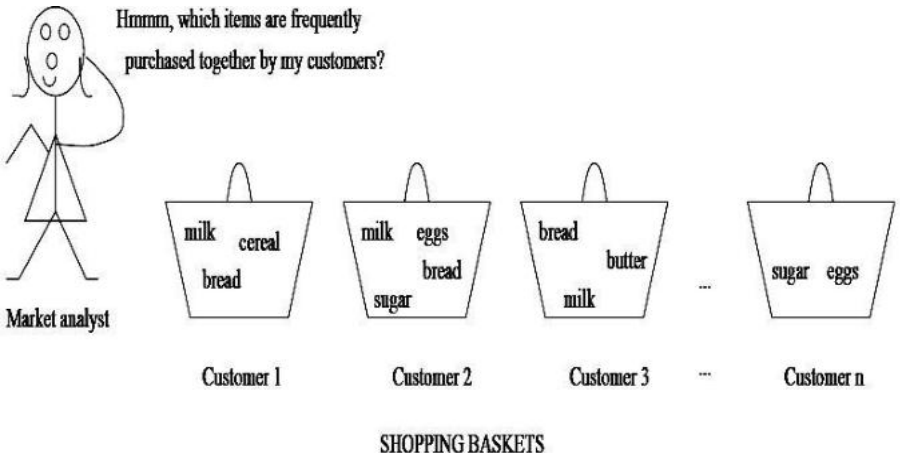
- Boolean vs. quantitative associations (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") ® buys(x, "DBMiner") [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") ® buys(x, "PC") [1%, 75%]
- Single dimension vs. multiple dimensional associations (see ex. Above)
- 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)?

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 itemset's *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.

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 can not 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.

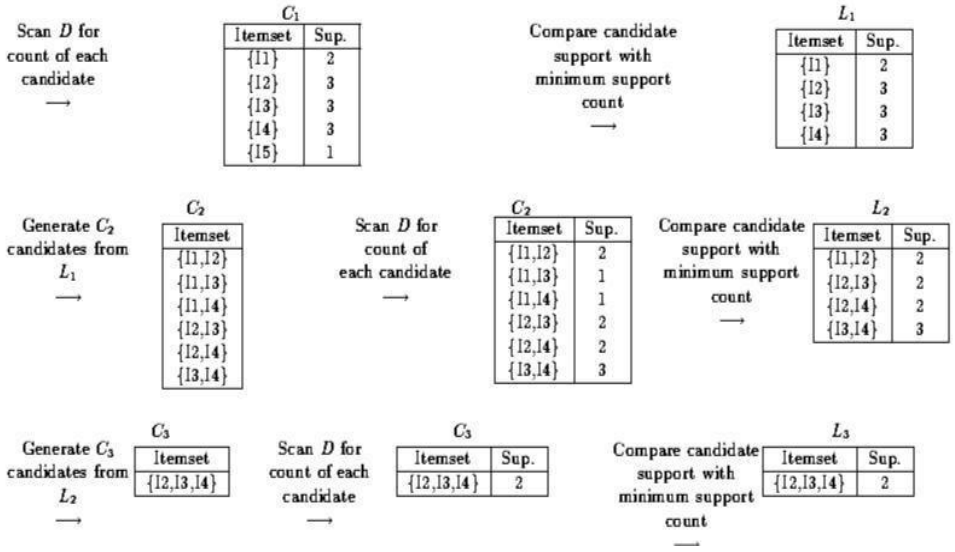
Method:

```
1)  L1 = find_frequent_1-itemsets(D);
2)  for (k = 2; Lk-1 ≠ φ; 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:

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

Header Table

Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p3		

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

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) 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

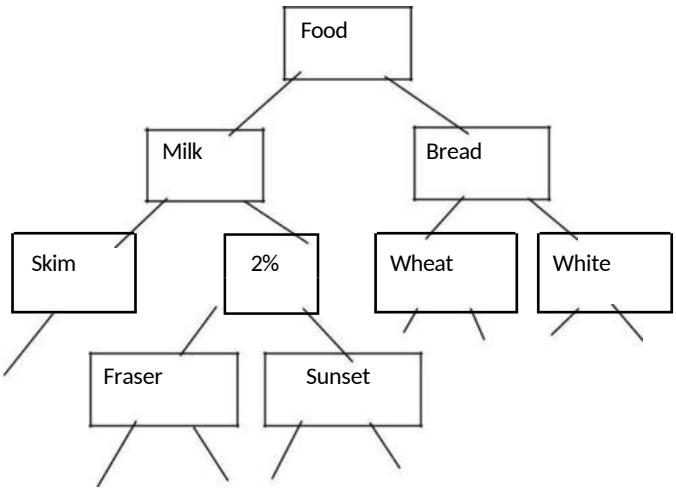
Mining multilevel association rules from transactional databases.

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.

- Rules regarding itemsets at

  appropriate levels could be quite useful.

- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - First find high-level strong rules:

    milk ® bread [20%, 60%].

  - Then find their lower-level "weaker" rules:

    2% milk ® wheat bread [6%, 50%].

- Variations at mining multiple-level association rules.
  - Level-crossed association rules:

2% *milk ® Wonder wheat bread*

– Association rules with multiple, alternative hierarchies:

2% *milk* ® *Wonder bread*

Multi-level Association: Uniform Support vs. Reduced Support

- Uniform Support: the same minimum support for all levels
  - + One minimum support threshold. No need to examine itemsets containing any item whose ancestors do not have minimum support.
  - – Lower level items do not occur as frequently. If support threshold
    - too high ✍ miss low level associations
    - too low ✍ generate too many high level associations
- Reduced Support: reduced minimum support at lower levels
  - There are 4 search strategies:
    - Level-by-level independent
    - Level-cross filtering by k-itemset
    - Level-cross filtering by single item
    - Controlled level-cross filtering by single item

Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Example
  - milk ✍ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ✍ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the "expected" value, based on the rule's ancestor

Multi-Level Mining: Progressive Deepening

- A top-down, progressive deepening approach:
  - First mine high-level frequent items:

    milk (15%), bread (10%)

– Then mine their lower-level "weaker" frequent itemsets:

2% milk (5%), wheat bread (4%)

- Different min_support threshold across multi-levels lead to different algorithms:

– If adopting the same *min_support* across multi-levels then toss *t* if any of *t*'s ancestors is infrequent.

– If adopting reduced *min_support* at lower levels then examine only those descendents whose ancestor's support is frequent/non-negligible.

Correlation in detail.

- Interest (correlation, lift)
  - taking both P(A) and P(B) in consideration
  - P(A^B)=P(B)*P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

Correlation

- 2 measures correlation between categorical attributes

$$\chi^2 = \sum \frac{(observed - expected)^2}{expected}$$

	game	not game	sum(row)
video	4000(4500)	3500(3000)	7500
not video	2000(1500)	500 (1000)	2500
sum(col.)	6000	4000	10000

- expected(i,j) = count(row i) * count(column j) / N
- 2 = (4000 - 4500)2 / 4500 - (3500 - 3000)2 / 3000 - (2000 - 1500)2 / 1500 - (500 - 1000)2 / 1000 = 555.6
- 2 > 1 and observed value of (game, video) < expected value, there is a negative correlation

Numeric correlation

- Correlation concept in statistics
  - Used to study the relationship existing between 2 or more numeric variables
  - A correlation is a measure of the linear relationship between variables
    Ex: number of hours spent studying in a class with grade received
  - Outcomes:
    - positively related
    - Not related
    - negatively related
  - Statistical relationships
    - Covariance
    - Correlation coefficient

Constraint-Based Mining in detail.

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.
  - Dimension/level constraints:
    - in relevance to region, price, brand, customer category.
  - Rule constraints
    - small sales (price < $10) triggers big sales (sum > $200).

- Interestingness constraints:
  - strong rules (min_support ≥ 3%, min_confidence ≥ 60%).

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - P(x, y) ^ Q(x, w) ® takes(x, "database systems").
  - Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD'98).
    - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000
- 1-variable vs. 2-variable constraints (Lakshmanan, et al. SIGMOD'99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Constrain-Based Association Query

- Database: (1) trans (TID, Itemset), (2) itemInfo (Item, Type, Price)
- A constrained asso. query (CAQ) is in the form of {(*S1, S2*)|*C* },
  - where C is a set of constraints on S1, S2 including frequency constraint
- A classification of (single-variable) constraints:
  - Class constraint: S ∈ A. *e.g. S ∈ Item*
  - Domain constraint:
    - *S* θ *v,* θ ∈ { =, ≠, <, ≤, >, ≥ }. *e.g. S.Price < 100*
    - *v* θ *S,* θ is ∈ or ∋. *e.g. snacks ∋ S.Type*
    - *V* θ *S,* or *S* θ *V,* θ ∈ { =, ⊂, ⊃, ⊆, ⊇ }
      - *e.g.* {*snacks, sodas* } ⊂ *S.Type*
  - Aggregation constraint: *agg(S)* θ *v,* where *agg* is in {*min, max, sum, count, avg*}, and θ ∈ { =, ≠, <, ≤, >, ≥ }.
    - *e.g. count(S1.Type)* = *1 , avg(S2.Price)* < *100*

Constrained Association Query Optimization Problem

- Given a CAQ = { (*S1, S2*) | *C* }, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- Our approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Categories of Constraints.

1. Anti-monotone and Monotone Constraints

- constraint Ca is anti-monotone iff. for any pattern S not satisfying Ca, none of the super-patterns of S can satisfy Ca
- A constraint Cm is monotone iff. for any pattern S satisfying Cm, every super-pattern of S also satisfies it

2. Succinct Constraint

- A subset of item Is is a succinct set, if it can be expressed as ? p(I) for some selection predicate p, where ? is a selection operator
- SP 2I is a succinct power set, if there is a fixed number of succinct set I1, …, Ik I, s.t. SP can be expressed in terms of the strict power sets of I1, …, Ik using union and minus
- A constraint Cs is succinct provided SATCs(I) is a succinct power set

3. Convertible Constraint

- Suppose all items in patterns are listed in a total order R
- A constraint C is convertible anti-monotone iff a pattern S satisfying the constraint implies that each suffix of S w.r.t. R also satisfies C
- A constraint C is convertible monotone iff a pattern S satisfying the constraint implies that each pattern of which S is a suffix w.r.t. R also satisfies C

Property of Constraints: Anti-Monotone

- Anti-monotonicity: *If a set S violates the constraint, any superset of S violates the constraint.*
- Examples:
  - *sum(S.Price)* ☒ *v* is anti-monotone
  - *sum(S.Price)* ☙ *v* is not anti-monotone
  - *sum(S.Price)* = *v* is partly anti-monotone
- Application:
  - Push "*sum(S.price)* ☒ 1000" deeply into iterative frequent set computation.

Example of Convertible Constraints: Avg(S) ☋ V

- Let R be the value descending order over the set of items
  - E.g. I={9, 8, 6, 4, 3, 1}
- Avg(S) ☙ v is convertible monotone w.r.t. R
  - If S is a suffix of S1, avg(S1) ☙ avg(S)
    - {8, 4, 3} is a suffix of {9, 8, 4, 3}
    - avg({9, 8, 4, 3})=6 ☙ avg({8, 4, 3})=5
  - If S satisfies avg(S) ☙v, so does S1
    - {8, 4, 3} satisfies constraint avg(S) ☙ 4, so does {9, 8, 4, 3}

Property of Constraints: Succinctness

- Succinctness:
  - For any set S1 and S2 satisfying C, S1 ▯ S2 satisfies C
  - Given A1 is the sets of size 1 satisfying C, then any set S satisfying C are based on A1 , i.e., it contains a subset belongs to A1 ,
- Example :
  - *sum(S.Price)* ☙ *v* is not succinct
  - *min(S.Price)* ☒ *v* is succinct

- Optimization:
  – If C is succinct, then C is pre-counting prunable. The satisfaction of the constraint alone is not affected by the iterative support counting.

Classification and Prediction

- **Classification:**
  – predicts categorical class labels
  – 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
- **Prediction**
  – models continuous-valued functions, i.e., predicts unknown or missing values
- **Typical applications**
  – Credit approval
  – Target marketing
  – Medical diagnosis
  – Fraud detection
  –

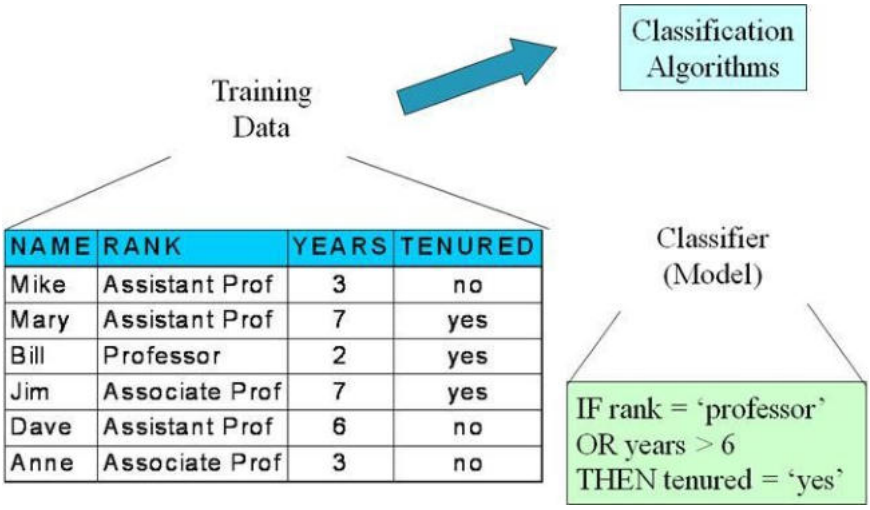
Classification—A Two-Step Process

- 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

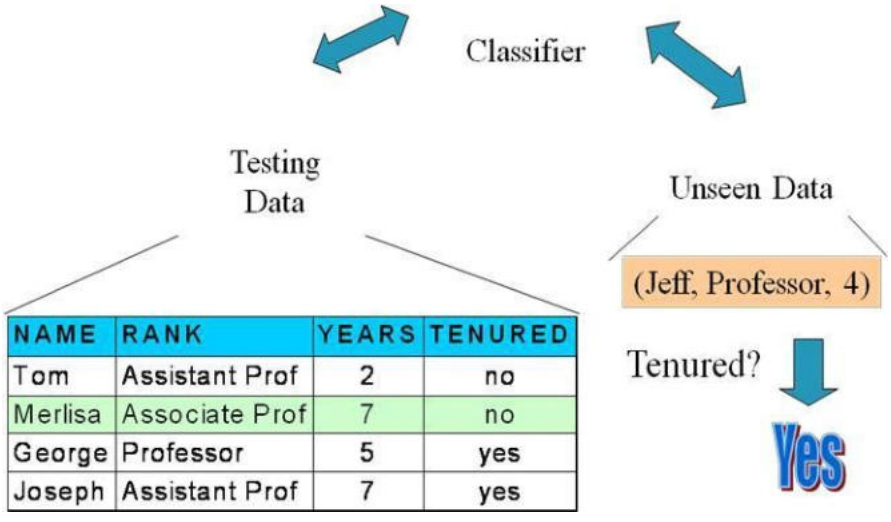
- Model usage: for classifying future or unknown objects
  – Estimate accuracy of the model

- 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



Process (2): Using the Model in Prediction



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

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

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
-
- Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

IF *age* = "31…40" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

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 overfitting
  - 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
  - Postpruning: 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

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

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

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h).$$

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

Naïve Bayes Classifier (I)

- A simplified assumption: attributes are conditionally independent:

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

- 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

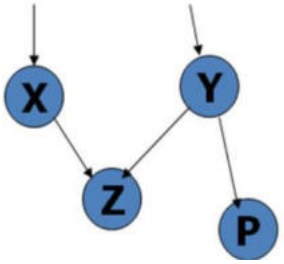
- Naïve assumption: attribute independence

$P(x1,\ldots,xk|C) = P(x1|C)\cdot\ldots\cdot P(xk|C)$

- 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

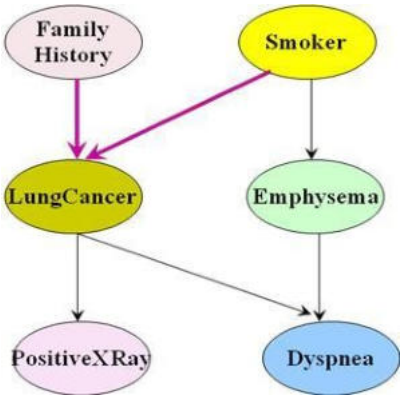
Bayesian Belief Networks

- 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:

$$P(x_1,\ldots,x_n) \quad \circ \; P(x_i \mid Parents(Y_i))$$

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

Pruning of decision trees

Discarding one or more subtrees and replacing them with leaves simplify a decision tree, and that is the main task in decision-tree pruning. In replacing the subtree 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 (subtrees) 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:

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

C4.5 follows the *postpruning* 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 subtree is less than weighted sum of $U_{25\%}$ for the leaves (predicted error for the subtree), then a subtree 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 subtree 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 subtree and replacing it with its root node as a new, generalized leaf node.

Figure : Pruning a subtree by replacing it with one leaf node

To analyze the possibility of replacing the subtree 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.

Rule Based Classification

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

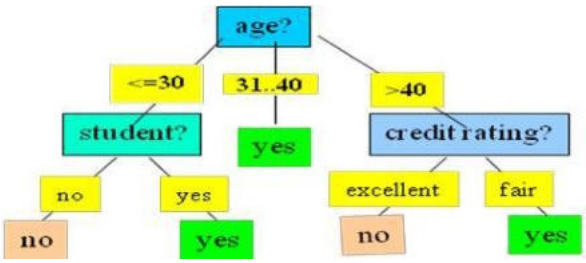
coverage(R) = n_{covers} /|D| /* D: training data set */

accuracy(R) = $n_{correct}$ / n_{covers}

- If more than one rule is triggered, need conflict resolution
  - Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute test*)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



- **Example: Rule extraction from our *buys_computer* decision-tree**

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

Rule Extraction from the Training Data

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

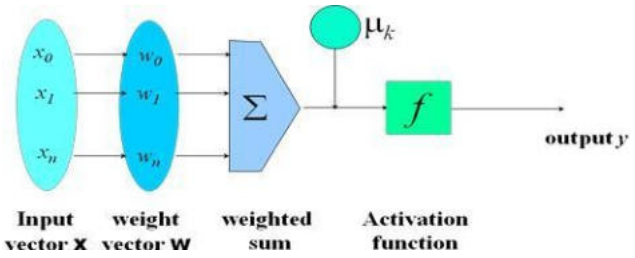
Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

Neural Network as a Classifier

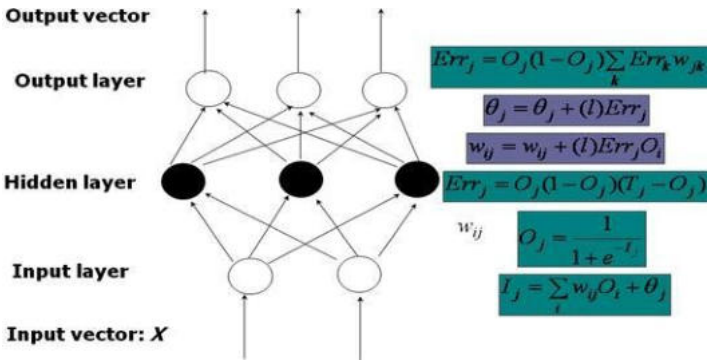
- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

A Neuron (= a perceptron)



- The *n*-dimensional input vector x is mapped into variable y by means of the scalar product and a nonlinear function mapping

A Multi-Layer Feed-Forward Neural Network



- The inputs to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the input layer
- They are then weighted and fed simultaneously to a hidden layer
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network's prediction
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform nonlinear regression: Given enough hidden units and enough training samples, they can closely approximate any function

Backpropagation

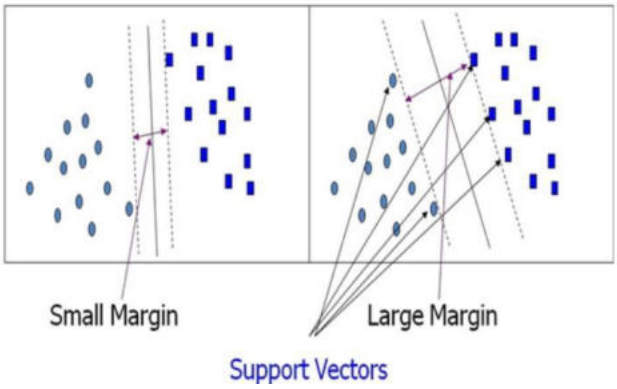
- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value
- Modifications are made in the “backwards” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “backpropagation”
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

- Efficiency of backpropagation: Each epoch (one interation through the training set) takes O(|D| * *w*), with |D| tuples and *w* weights, but # of epochs can be exponential to n, the number of inputs, in the worst case
- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

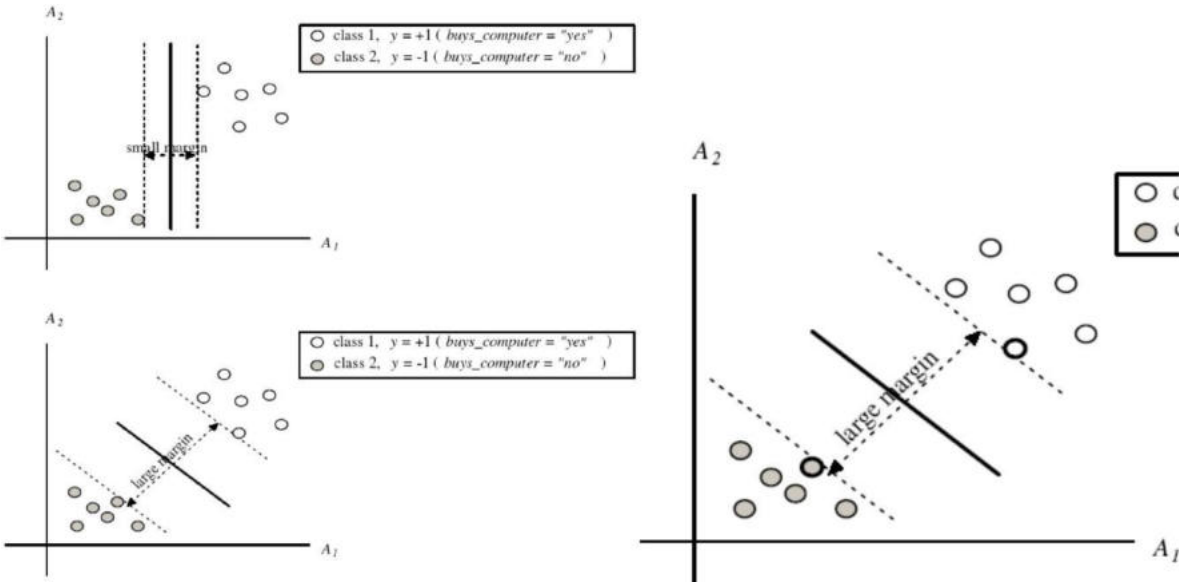
SVM—Support Vector Machines

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)
- **Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)**
- **Used both for classification and prediction**
- **Applications:**
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM—General Philosophy



SVM—Margins and Support Vectors



SVM—Linearly Separable

- A separating hyperplane can be written as

$W \bullet X + b = 0$

where $W=\{w_1, w_2, ..., w_n\}$ is a weight vector and b a scalar (bias)

- For 2-D it can be written as

$w_0 + w_1 x_1 + w_2 x_2 = 0$

- The hyperplane defining the sides of the margin:

H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are support vectors

- This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints ☾ *Quadratic Programming (QP)* ☾ Lagrangian multipliers

Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

Associative Classification

- Associative classification
  - Association rules are generated and analyzed for use in classification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Classification: Based on evaluating a set of rules in the form of ☾

$P_1 \wedge p_2 \ldots \wedge p_l$ "$A_{class} = C$" (conf, sup)

- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time

In many studies, associative classification has been found to be more accurate than some traditional classification methods, such as C4.

Typical Associative Classification Methods

- CBA (Classification By Association: Liu, Hsu & Ma, KDD'98)
  - Mine association possible rules in the form of

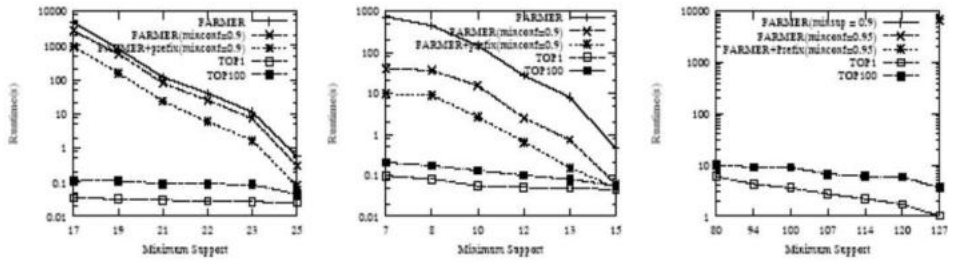
- Cond-set (a set of attribute-value pairs) ☾ class label
- Build classifier: Organize rules according to decreasing precedence based on confidence and then support

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- CPAR (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis)
  - High efficiency, accuracy similar to CMAR
- RCBT (Mining top-*k* covering rule groups for gene expression data, Cong et al. SIGMOD'05)
  - Explore high-dimensional classification, using top-k rule groups
  - Achieve high classification accuracy and high run-time efficiency

Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

Table 2: Classification Results



(a) ALL-AML leukemia (b) Lung Cancer (c) Ovarian Cancer

Other Classification Methods

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

Genetic Algorithms

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
  - If an attribute has k > 2 values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offsprings
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offsprings are generated by *crossover* and *mutation*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

Rough Set Approach:

- Rough sets are used to **approximately or "roughly" define equivalent classes**

- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)

Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity

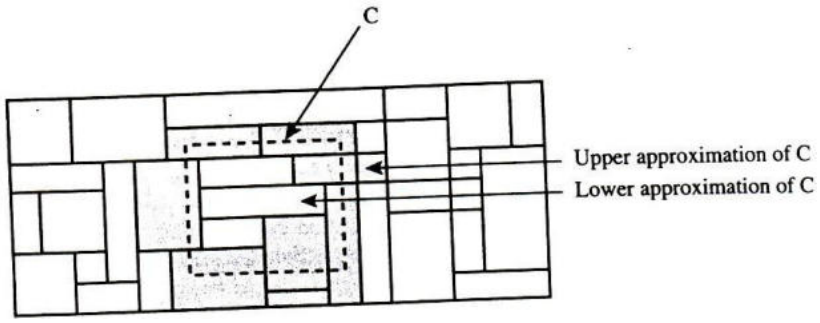
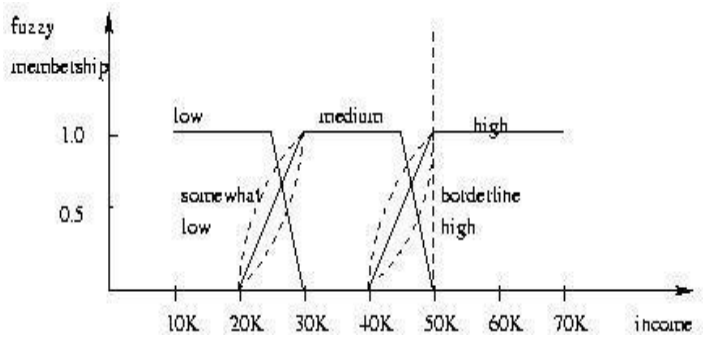


Figure: A rough set approximation of the set of tuples of the class C suing lower and upper approximation sets of C. The rectangular regions represent equivalence classes

Fuzzy Set approaches

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined



Prediction

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or predictor variables and a *dependent* or response variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

Linear Regression

- Linear regression: involves a response variable y and a single predictor variable

x y = w_0 + w_1 x

where w_0 (y-intercept) and w_1 (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line
  - Multiple linear regression: involves more than one predictor variable
  - Training data is of the form ($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2),…, ($\mathbf{X}_{|D|}$, $y_{|D|}$)
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$

- Solvable by extension of least square method or using SAS, S-Plus
- Many nonlinear functions can be transformed into the above

Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

convertible to linear with new variables: $x_2 = x_2$, $x_3 = x_3$

$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

UNIT-IV:ClusteringandAdvancedMiningMethods

Introduction to Clustering: Types and Applications, Partitioning Methods: K-Means, K-Medoids, Hierarchical Clustering: Agglomerative and Divisive, Density-Based Clustering:DBSCAN, OPTICS, Evaluation of Clustering Results, Outlier Detection and Handling.

CLUSTERING AND ADVANCED MINING METHODS

Introduction to Clustering

1. Definition

- **Clustering** is an **unsupervised learning technique** in data mining.
- It **groups similar data points** into clusters such that:
  - Data points in the same cluster are **similar**.
  - Data points in different clusters are **dissimilar**.

Key Idea: No class labels are provided; the algorithm identifies patterns automatically.

2. Types of Clustering

A. Based on Methodology

1. **Partitioning Clustering**
   - Divides data into **non-overlapping subsets** (clusters).
   - Each object belongs to **exactly one cluster**.
   - Example: **K-Means**, K-Medoids.
2. **Hierarchical Clustering**
   - Creates a **tree of clusters (dendrogram)**.
   - Types:
     - **Agglomerative (Bottom-Up):** Start with individual points, merge clusters.
     - **Divisive (Top-Down):** Start with all data, split recursively.
3. **Density-Based Clustering**
   - Clusters are formed based on **dense regions of points**.
   - Example: **DBSCAN** – finds clusters of arbitrary shape, identifies noise/outliers.
4. **Grid-Based Clustering**
   - Divides space into a **finite number of cells** and performs clustering on the grid.

- Example: **STING (Statistical Information Grid).**

5. **Model-Based Clustering**
    - Assumes data is generated by a **probabilistic model**.
    - Example: **Expectation-Maximization (EM)** using Gaussian Mixture Models.

B. Based on Data Type

- **Interval/Ratio data:** Use distance-based methods (Euclidean, Manhattan).
- **Categorical data:** Use similarity measures (Jaccard, Hamming).
- **Mixed data:** Combine distance and similarity measures.

3. Applications of Clustering

Domain	Use Case
Marketing	Customer segmentation (VIP, Regular, Occasional)
Biology	Gene expression analysis, species classification
Finance	Fraud detection, risk grouping
Web / Social Networks	User behavior analysis, community detection
Image Processing	Image segmentation, object recognition
Urban Planning	Traffic pattern analysis, city zoning

Partitioning Clustering Methods

1. Definition

- **Partitioning clustering** divides a dataset into **non-overlapping clusters**.
- Each data point belongs to **exactly one cluster.**
- Goal: Minimize **intra-cluster distance** and maximize **inter-cluster distance**.

2. K-Means Clustering

Algorithm Steps:

1. Choose **K**, the number of clusters.
2. Initialize **K centroids** randomly.
3. Assign each data point to the **nearest centroid**.
4. Recompute **centroids** as the mean of points in each cluster.

5. Repeat steps 3–4 until **centroids do not change** or a maximum number of iterations is reached.

Distance Metric:

- Euclidean distance (most common):

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Pros:

- Simple and fast.
- Works well with **large datasets**.

Cons:

- Requires predefining **K**.
- Sensitive to **outliers**.
- Works best with **spherical clusters**.

3. K-Medoids Clustering

Key Difference from K-Means:

- Uses **medoids** (actual data points) as cluster centers instead of mean.
- Less sensitive to **outliers**.

Algorithm (PAM – Partitioning Around Medoids):

1. Choose K representative objects (medoids).
2. Assign each data point to the **nearest medoid**.
3. Swap medoid with a non-medoid point and compute **total cost** (sum of distances).
4. If cost decreases, update medoid; otherwise, keep original.
5. Repeat until **no improvement**.

Pros:

- Robust to **outliers and noise**.
- Works for **arbitrary distance measures**.

Cons:

- More **computationally expensive** than K-Means.

Hierarchical Clustering

1. Definition

- **Hierarchical clustering** creates a **tree-like structure (dendrogram)** of clusters.
- Does **not require a predefined number of clusters**.
- Clusters can be formed **bottom-up (agglomerative)** or **top-down (divisive)**.

2. Types of Hierarchical Clustering

A. Agglomerative (Bottom-Up)

- Start with **each data point as a single cluster**.
- Iteratively **merge the closest clusters** until all points belong to one cluster or stopping criterion is met.

Steps:

1. Compute **distance matrix** between all points.
2. Merge the **closest two clusters**.
3. Update distance matrix using a **linkage method**:
    - **Single linkage:** Minimum distance between points in clusters.
    - **Complete linkage:** Maximum distance.
    - **Average linkage:** Average distance.
4. Repeat until all points are in one cluster.

Pros: Simple, produces dendrogram.
Cons: Computationally expensive for large datasets.

B. Divisive (Top-Down)

- Start with **all data points in one cluster**.
- Recursively **split clusters** until each point is a single cluster or stopping criterion is met.

Steps:

1. Treat all points as one cluster.
2. Identify the **most heterogeneous cluster** to split.
3. Repeat splitting recursively.

Pros: Can produce more balanced clusters.

Cons: Computationally expensive, less commonly used than agglomerative.

3. Dendrogram

- Tree-like diagram showing **nested cluster structure**.
- **Height of the branches** represents **distance or dissimilarity** between clusters.
- Cutting the dendrogram at a certain height gives the desired **number of clusters**.

Density-Based Clustering and Evaluation

1. Density-Based Clustering

A. Definition

- Groups points based on **dense regions** in the dataset.
- Points in sparse regions are treated as **noise or outliers**.
- Does **not require the number of clusters** to be specified in advance.

B. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) Key Parameters:

1. **Eps (ε):** Radius to search for neighbors.
2. **MinPts:** Minimum number of points required to form a dense region.

Steps:

1. For each point, find **ε-neighborhood**.
2. If neighbors ≥ MinPts → **core point**, else noise or border point.
3. Connect **density-reachable points** to form clusters.

Pros:

- Finds **arbitrary-shaped clusters**.
- Can **detect outliers naturally**.

Cons:

- Sensitive to ε and MinPts values.
- Not suitable for datasets with **varying densities**.

C. OPTICS (Ordering Points To Identify the Clustering Structure)

- Improvement over DBSCAN for **varying density clusters**.
- Produces **reachability plot** instead of explicit clusters.
- Allows extraction of clusters at different density levels.

Pros: Handles varying densities better than DBSCAN.
Cons: More complex and computationally intensive.

2. Evaluation of Clustering Results

A. Internal Evaluation (Without ground truth)

- **Silhouette Coefficient (SC):** Measures how similar a point is to its own cluster vs other clusters.

$$SC = \frac{b - a}{\max(a, b)}$$

- **a:** Average intra-cluster distance
- **b:** Average nearest-cluster distance
- **Davies-Bouldin Index, Dunn Index:** Measures cluster compactness and separation.

B. External Evaluation (With ground truth)

- **Purity:** Fraction of dominant class in each cluster.
- **Rand Index, F-Measure:** Compare clustering results to known labels.

3. Outlier Detection and Handling

A. Definition:

- Points that **deviate significantly** from the rest of the data.

B. Methods:

1. **Statistical methods:** Detect points beyond **mean ± 3σ**.
2. **Distance-based methods:** Points far from cluster centroids.
3. **Density-based methods:** DBSCAN naturally identifies noise points.

C. Handling Outliers:

- Remove from dataset
- Treat separately for analysis

UNIT-V:DataMiningApplicationsandTools

Web Mining, Text Mining, and Spatial Data Mining, Temporal and Sequence Data Mining, Introduction to Big Data and Scalable Mining, Data Mining Tools: WEKA, RapidMiner, Orange,CaseStudies:BusinessIntelligence,FraudDetection, E-commerce,EthicalandPrivacyIssuesinData Mining.

Web Mining

1. Definition

- **Web Mining** is the application of **data mining techniques** to discover useful patterns and knowledge from the **World Wide Web**.
- It deals with analyzing:
  - **Web content** (text, images, multimedia)
  - **Web structure** (links, hierarchy)
  - **Web usage** (clickstreams, user behavior)

2. Types of Web Mining

A. Web Content Mining

- Extracts information from **webpage content**.
- Works on text, images, audio, video, and metadata.
- Techniques: Information Retrieval (IR), Natural Language Processing (NLP), Multimedia Mining.
- **Applications:** Sentiment analysis, product recommendation, fake news detection.

B. Web Structure Mining

- Analyzes the **structure of hyperlinks** between web pages.
- Uses **graph theory** (nodes = web pages, edges = links).
- Algorithms: **PageRank, HITS**.
- **Applications:** Search engines ranking, community detection, website categorization.

C. Web Usage Mining

- Studies **user interaction data** collected in web server logs, browser history, cookies, and clickstreams.
- Steps: Data collection → Preprocessing → Pattern discovery → Pattern analysis.
- **Applications:** Personalized recommendations (Amazon, Netflix), targeted

advertising, fraud detection.

3. Applications of Web Mining

1. **E-commerce:** Product recommendations, customer profiling.

2. **Search Engines:** Ranking pages (Google PageRank).
3. **Social Media Analysis:** Community detection, sentiment analysis.
4. **Cybersecurity:** Detecting fraud, phishing websites.
5. **Digital Marketing:** Targeted ads, customer segmentation.

4. Challenges in Web Mining

- Data is **heterogeneous** (text, multimedia, links).
- **Scalability**: Huge volume of web data.
- **Privacy and security** concerns.
- **Dynamic nature**: Web content changes frequently.

Text Mining

1. Definition

- **Text Mining** (also called Text Data Mining or Text Analytics) is the process of extracting **useful information, patterns, and knowledge** from **unstructured text data**.
- Involves converting raw text into structured data for analysis.

2. Steps in Text Mining

1. **Text Preprocessing**
    - **Tokenization:** Splitting text into words/sentences.
    - **Stop-word Removal:** Removing common words (e.g., "the", "is").
    - **Stemming/Lemmatization:** Reducing words to their base/root form.
    - **Part-of-Speech (POS) Tagging:** Identifying nouns, verbs, adjectives, etc.
2. **Text Representation**
    - **Bag of Words (BoW):** Counts word frequency.
    - **TF-IDF (Term Frequency – Inverse Document Frequency):** Weighs words based on importance.
    - **Word Embeddings (Word2Vec, GloVe, BERT):** Captures semantic meaning of words.
3. **Pattern Discovery**
    - **Clustering:** Grouping similar documents.
    - **Classification:** Assigning categories (e.g., spam vs non-spam).

   - **Association Rule Mining:** Finding relationships between terms.
4. **Knowledge Extraction & Visualization**
   - Generate **summaries, word clouds, sentiment scores**.

3. Applications of Text Mining

- **Search Engines:** Ranking and retrieving relevant documents.
- **Sentiment Analysis:** Understanding opinions in reviews/social media.
- **Spam Detection:** Filtering unwanted emails.
- **Healthcare:** Mining medical literature for disease-drug relationships.
- **Legal and Business:** Contract analysis, market intelligence.

4. Challenges

- **Ambiguity:** Same word may have different meanings.
- **Language diversity:** Multiple languages, slang, abbreviations.
- **High dimensionality:** Large vocabulary increases complexity.
- **Unstructured data:** Text lacks fixed format.

Spatial Data Mining (Spatial Mining)

1. Definition

- **Spatial Data Mining** is the process of discovering interesting, useful, and non-trivial patterns from **spatial datasets** (data with location or geographical components).
- Example: “Where do most customers live?” or “Which regions are prone to earthquakes?”

2. Spatial Data Characteristics

Spatial data is **different from traditional data** because it includes:

- **Location** (latitude, longitude, coordinates)
- **Spatial relationships** (adjacency, distance, connectivity, overlap)
- **Attributes** (population, temperature, soil type, etc.)

3. Spatial Data Mining Tasks

1. **Spatial Classification**
   - Assigns a class label to spatial objects.
   - Example: Classifying land into urban, rural, forest, etc.
2. **Spatial Clustering**

- Groups nearby objects into meaningful clusters.
- Example: Identifying earthquake zones or disease outbreak clusters.

3. **Spatial Association Rules**
   - Finds relationships among spatial and non-spatial attributes.
   - Example: "Cities near rivers tend to have higher population density."
4. **Spatial Trend Detection**
   - Detects patterns that change with location.
   - Example: Pollution levels decreasing as you move away from industrial areas.

4. Applications of Spatial Mining

- **Geographic Information Systems (GIS):** Land use analysis, urban planning.
- **Agriculture:** Soil and crop pattern analysis.
- **Environmental Studies:** Tracking deforestation, climate change, pollution spread.
- **Public Health:** Disease spread mapping (COVID-19 hotspots).
- **Retail & Marketing:** Store placement, customer location analysis.
- **Transportation:** Traffic congestion analysis, route optimization.

5. Challenges

- Very **large and complex datasets**.
- **High dimensionality** (spatial + temporal + attribute data).
- **Heterogeneity**: Maps, satellite images, GPS, sensors.
- **Scalability**: Requires powerful algorithms for real-time analysis.

Temporal and Sequence Data Mining

1. Temporal Data Mining

Definition:

- Deals with **time-oriented data** where records are ordered by time.
- Goal: Discover patterns, trends, or correlations across time.

Examples of Temporal Data:

- Stock market prices
- Weather data (temperature, rainfall)
- Sensor readings (IoT, smart devices)

- Sales records over months

Key Tasks:

1. **Trend Analysis:** Detecting long-term increase/decrease.
    - Example: Rising global temperature trends.
2. **Periodic Patterns:** Discovering repeating cycles.
    - Example: Seasonal sales increase during festivals.
3. **Temporal Association Rules:** Finding time-based rules.
    - Example: "If flu cases rise in January, medicine sales rise in February."
4. **Time-Series Forecasting:** Predicting future values.
    - Example: Forecasting stock prices, rainfall.

2. Sequence Data Mining

Definition:

- Finds patterns in **ordered sequences of events/items** (order matters).
- A sequence is an ordered list like:
- $<(A \rightarrow B \rightarrow C)>$

where A, B, C are events/items.

Examples:

- Customer purchase sequences (milk → bread → butter).
- Web clickstream (Homepage → Product Page → Checkout).
- DNA/protein sequences in bioinformatics.

Key Techniques:

1. **Sequential Pattern Mining (SPM):** Discover frequent subsequences.
    - Example: "60% of customers who buy laptop → later buy mouse."
    - Algorithms: Apriori-based, GSP (Generalized Sequential Pattern).
2. **Episode Mining:** Finding frequent events within time intervals.
    - Example: "Machine error followed by shutdown within 10 minutes."

3. Applications

- **Business:** Market basket analysis, customer behavior.
- **Healthcare:** Patient treatment sequences, disease progression.

- **Web Mining:** User clickstreams, navigation patterns.
- **Finance:** Stock trends, fraud detection.
- **Bioinformatics:** DNA/RNA sequence analysis.

4. Challenges

- Large volumes of **time-series and sequential data**.
- **Noise and missing values** in real-world data.
- **Scalability** for long sequences.
- Complex **temporal relationships** (overlaps, gaps).

Introduction to Big Data and Scalable Mining

1. What is Big Data?

- **Big Data** refers to extremely **large, complex, and fast-growing datasets** that cannot be efficiently processed using traditional database systems.
- Defined by the **5 V's**:
    1. **Volume** – Huge amounts of data (terabytes, petabytes).
    2. **Velocity** – Speed of data generation and processing (real-time streams).
    3. **Variety** – Different data formats (structured, semi-structured, unstructured).
    4. **Veracity** – Uncertainty, inconsistency, or noise in data.
    5. **Value** – Extracting meaningful insights for decision-making.

Examples:

- Social media data (Twitter, Facebook).
- IoT sensor data.
- Online transactions.
- Genomic data in healthcare.

2. Challenges of Big Data

- **Storage and Management** (huge size).
- **Processing Speed** (real-time analysis needed).
- **Data Integration** (variety of formats).
- **Scalability** (algorithms must handle growing datasets).
- **Data Privacy and Security**.

3. Scalable Data Mining

Definition:

- **Scalable data mining** means adapting mining algorithms so they can handle **massive datasets efficiently** without losing accuracy.

Techniques for Scalability:

1. **Parallel and Distributed Computing**
    - Use multiple processors or clusters.
    - Example: **MapReduce, Apache Hadoop, Apache Spark**.
2. **Sampling and Approximation**
    - Instead of full dataset, use representative samples.
3. **Incremental Algorithms**
    - Update results dynamically as new data arrives (stream mining).
4. **Dimensionality Reduction**
    - Techniques like PCA, feature selection to reduce complexity.
5. **Cloud Computing**
    - Elastic resources to handle data storage & mining at scale.

4. Applications of Big Data Mining

- **Business Intelligence:** Customer profiling, recommendation engines (Amazon, Netflix).
- **Healthcare:** Analyzing patient records, disease prediction.
- **Finance:** Fraud detection, stock market analysis.
- **Smart Cities:** Traffic prediction, energy optimization.
- **Social Media:** Trend detection, sentiment analysis.

5. Tools and Frameworks

- **Hadoop Ecosystem:** HDFS (storage), MapReduce (processing), Hive, Pig.
- **Apache Spark:** In-memory distributed processing for fast analytics.
- **NoSQL Databases:** MongoDB, Cassandra, HBase.
- **Cloud Platforms:** AWS, Google BigQuery, Azure HDInsight.

Data Mining Tools

1. WEKA (Waikato Environment for Knowledge Analysis)

- **Developed by:** University of Waikato, New Zealand.
- **Type:** Open-source, Java-based software.
- **Features:**
  - Collection of **machine learning algorithms** for classification, regression, clustering, association rule mining.
  - Supports **data preprocessing, visualization, and evaluation**.
  - Easy-to-use **GUI (Explorer, Experimenter, Knowledge Flow)**.
- **Applications:** Academic research, teaching, small-scale industry projects.
- **Strengths:** User-friendly, wide range of algorithms.
- **Limitations:** Not ideal for **big data or real-time processing**.

2. RapidMiner

- **Type:** Commercial data science platform (open-source version also available).
- **Features:**
  - Provides **drag-and-drop GUI** (no coding needed).
  - Supports **data preprocessing, machine learning, text mining, deep learning, predictive analytics**.
  - Integrates with **Python, R, and Hadoop/Spark** for scalability.
- **Applications:** Business analytics, customer segmentation, fraud detection.
- **Strengths:** Beginner-friendly, scalable, integrates with big data tools.
- **Limitations:** Advanced features may require paid version.

3. Orange

- **Type:** Open-source data mining and machine learning tool (Python-based).
- **Features:**
  - Provides **visual programming interface** (widgets for workflows).
  - Supports **classification, clustering, regression, association rules, text mining**.
  - Strong in **data visualization** (scatter plots, heatmaps, decision trees).
- **Applications:** Education, rapid prototyping, research projects.
- **Strengths:** Highly visual, Python integration, interactive analysis.
- **Limitations:** Not as scalable as industrial-grade tools like Spark.

4. Comparison Table

Tool	Type	Strengths	Limitations
WEKA	Open-source (Java)	Easy GUI, wide algorithms, good for teaching	Not scalable for big data
RapidMiner	Commercial + Open-source	Drag-and-drop, scalable, business-friendly	Paid for advanced use
Orange	Open-source (Python)	Visual, interactive, strong in visualization	Limited big data handling

Case Studies in Data Mining

1. Business Intelligence (BI)

◆ **Definition**

Business Intelligence refers to techniques and tools for transforming raw data into meaningful information to support decision-making. Data mining is at the **core of BI**.

◆ **Role of Data Mining in BI**

- Identifies **patterns and trends** in large datasets.
- Helps in **predictive analytics** for forecasting demand, sales, or risks.
- Improves **customer segmentation** and marketing strategies.
- Enables **real-time dashboards** and KPI monitoring.

◆ **Applications**

- **Sales Forecasting:** Predicting seasonal demand or customer churn.
- **Market Basket Analysis:** Finding products often bought together.
- **Customer Relationship Management (CRM):** Identifying high-value customers.
- **Operational Efficiency:** Optimizing supply chains, staffing, and logistics.

2. Fraud Detection

◆ **Definition**

Fraud detection uses data mining to identify unusual patterns that may indicate **illegal or unauthorized activities**.

◆ **Common Domains**

- **Banking & Finance:** Detecting credit card fraud, money laundering.
- **Insurance:** Spotting false claims.
- **Telecom:** Preventing fake subscriptions or billing fraud.

◆ **Techniques**

- **Classification models (Decision Trees, Naïve Bayes):** Classify transactions as fraudulent/non-fraudulent.
- **Clustering & Outlier Detection:** Spot abnormal behaviors compared to typical user patterns.
- **Sequential Analysis:** Identify suspicious transaction sequences.
- **Real-time Mining:** Streaming analytics to flag fraud instantly.

◆ **Example**

- Credit card fraud detection systems: When a purchase deviates from usual spending patterns (e.g., sudden international purchase), it's flagged for verification.

3. E-Commerce

◆ **Importance**

E-commerce platforms generate **massive volumes of transactional and behavioral data**, making them ideal for data mining applications.

◆ **Applications**

- **Recommendation Systems:** Suggesting products using collaborative filtering (e.g., Amazon, Flipkart).
- **Customer Segmentation:** Identifying groups of buyers with similar habits.
- **Dynamic Pricing:** Using data mining to adjust prices based on demand and competition.
- **Churn Prediction:** Identifying customers likely to stop using the service.
- **Sentiment Analysis:** Mining reviews and feedback to understand customer satisfaction.

◆ **Example**

- Netflix and Amazon use **FP-Growth / Collaborative Filtering** to recommend shows or products.
- Flipkart adjusts product prices during festive sales using demand forecasts.

Ethical and Privacy Issues in Data Mining

1. Privacy Concerns

- **Personal Data Collection:** Mining involves analyzing sensitive personal information (financial records, health data, browsing habits).
- **Unauthorized Access:** If data is leaked or stolen, individuals' privacy can be violated.
- **Data Sharing Without Consent:** Companies may sell customer data to third parties.
- **Example:** Social media platforms tracking user activity to target ads.

2. Security Risks

- **Data Breaches:** Large datasets are attractive targets for hackers.
- **Improper Storage:** Weak encryption or poor database practices increase risk.
- **Result:** Identity theft, financial fraud, misuse of medical data.

3. Ethical Issues

- **Lack of Informed Consent:** Users may not know their data is being mined.
- **Surveillance Society:** Over-mining can lead to constant monitoring of citizens.
- **Bias in Data:** If datasets are biased, the models reinforce discrimination.
    - Example: Loan approvals biased against minority groups.
- **Transparency Issues:** Companies may not explain how mined data is used.

4. Legal and Regulatory Issues

- **Data Protection Laws:**
    - **GDPR (Europe):** Grants individuals rights to control personal data.
    - **HIPAA (US):** Protects medical information.
    - **IT Act (India):** Covers data security and misuse.
- Organizations must comply with these regulations or face penalties.

5. Solutions / Best Practices

- **Data Anonymization:** Remove personal identifiers before mining.
- **Access Control:** Only authorized personnel should handle sensitive data.
- **Fair Algorithms:** Regularly audit models to avoid bias.
- **Transparency:** Inform users about how their data is collected and used.
- **Opt-out Options:** Allow individuals to refuse participation in data collection.