VI Semester (R16)

DATA WAREHOUSE AND DATA MINING
INFORMATION TECHNOLOGY

Prepared by
Dr. M Madhu Bala, Professor, CSE
Dr. K Rajendra Prasad, Professor, CSE
Dr. N Rajasekhar, Professor, CSE
Dr. K Suvarchala, Professor, CSE
Ms. K. LaxmiNarayanamma, IT
Ms. M GeethaYadav, CSE
UNIT-I
Data Warehouse

• What is a data warehouse?

• A multi-dimensional data model

• Data warehouse architecture
What is Data Warehouse?

• Defined in many different ways, but not rigorously.
  – A decision support database that is maintained *separately* from the organization’s operational database
  – Support *information processing* by providing a solid platform of consolidated, historical data for analysis.

• “A data warehouse is a *subject-oriented, integrated, time-variant*, and *nonvolatile* collection of data in support of management’s decision-making process.”—W. H. Inmon

• Data warehousing:
  – The process of constructing and using data warehouses
Data Warehouse—Subject-Oriented

• Organized around major subjects, such as customer, product, sales

• Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing

• Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process
Data Warehouse—Integrated

• Constructed by integrating multiple, heterogeneous data sources
  – relational databases, flat files, on-line transaction records
• Data cleaning and data integration techniques are applied.
  – Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    • E.g., Hotel price: currency, tax, breakfast covered, etc.
  – When data is moved to the warehouse, it is converted.
Data Warehouse—Time Variant

• The time horizon for the data warehouse is significantly longer than that of operational systems
  – Operational database: current value data
  – Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)

• Every key structure in the data warehouse
  – Contains an element of time, explicitly or implicitly
  – But the key of operational data may or may not contain “time element”
Data Warehouse—Nonvolatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*
Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration: A query driven approach
  - Build wrappers/mediators on top of heterogeneous databases
  - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
  - Complex information filtering, compete for resources

- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis
Data Warehouse vs. Operational DBMS

- **OLTP (on-line transaction processing)**
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.

- **OLAP (on-line analytical processing)**
  - Major task of data warehouse system
  - Data analysis and decision making

- **Distinct features (OLTP vs. OLAP):**
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - Database design: ER + application vs. star + subject
  - View: current, local vs. evolutionary, integrated
  - Access patterns: update vs. read-only but complex queries
# OLTP vs. OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>users</strong></td>
<td>clerk, IT professional</td>
<td>knowledge worker</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>day to day operations</td>
<td>decision support</td>
</tr>
<tr>
<td><strong>DB design</strong></td>
<td>application-oriented</td>
<td>subject-oriented</td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>current, up-to-date</td>
<td>historical, summarized, multidimensional</td>
</tr>
<tr>
<td></td>
<td>detailed, flat relational</td>
<td>integrated, consolidated</td>
</tr>
<tr>
<td><strong>usage</strong></td>
<td>repetitive</td>
<td>ad-hoc</td>
</tr>
<tr>
<td><strong>access</strong></td>
<td>read/write</td>
<td>lots of scans</td>
</tr>
<tr>
<td></td>
<td>index/hash on prim. key</td>
<td></td>
</tr>
<tr>
<td><strong>unit of work</strong></td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td><strong># records accessed</strong></td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td><strong>#users</strong></td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td><strong>DB size</strong></td>
<td>100MB-GB</td>
<td>100GB-TB</td>
</tr>
<tr>
<td><strong>metric</strong></td>
<td>transaction throughput</td>
<td>query throughput, response</td>
</tr>
</tbody>
</table>

July 16, 2018
Why Separate Data Warehouse?

• High performance for both systems
  – DBMS—tuned for OLTP: access methods, indexing, concurrency control, recovery
  – Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation

• Different functions and different data:
  – **missing data**: Decision support requires historical data which operational DBs do not typically maintain
  – **data consolidation**: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  – **data quality**: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

• Note: There are more and more systems which perform OLAP analysis directly on relational databases
UNIT-I Data Warehouse

• What is a data warehouse?
• A multi-dimensional data model
• Data warehouse architecture
From Tables and Spreadsheets to Data Cubes

• A data warehouse is based on a **multidimensional data model** which views data in the form of a data cube

• A data cube, such as *sales*, allows data to be modeled and viewed in multiple dimensions
  – Dimension tables, such as *item (item_name, brand, type)*, or *time(day, week, month, quarter, year)*
  – Fact table contains measures (such as *dollars_sold*) and keys to each of the related dimension tables

• In data warehousing literature, an n-D base cube is called a **base cuboid**. The top most 0-D cuboid, which holds the highest-level of summarization, is called the **apex cuboid**. The lattice of cuboids forms a **data cube**.
Cube: A Lattice of Cuboids

0-D (apex) cuboid
1-D cuboids
2-D cuboids
3-D cuboids
4-D (base) cuboid
Conceptual Modeling of Data Warehouses

• 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
Example of Star Schema

- **Time**:
  - time_key
  - day
  - day_of_the_week
  - month
  - quarter
  - year

- **Item**:
  - item_key
  - item_name
  - brand
  - type
  - supplier_type

- **Branch**:
  - branch_key
  - branch_name
  - branch_type

- **Location**:
  - location_key
  - street
  - city
  - state_or_province
  - country

- **Sales Fact Table**:
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
  - dollars_sold
  - avg_sales

- **Measures**
Example of Snowflake Schema

- **time**
  - time_key
  - day
  - day_of_the_week
  - month
  - quarter
  - year

- **branch**
  - branch_key
  - branch_name
  - branch_type

- **Sales Fact Table**
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
  - dollars_sold
  - avg_sales

- **item**
  - item_key
  - item_name
  - brand
  - type
  - supplier_key

- **supplier**
  - supplier_key
  - supplier_type

- **location**
  - location_key
  - street
  - city_key

- **city**
  - city_key
  - city
  - state_or_province
  - country
Example of Fact Constellation

Sales Fact Table
- time_key
- item_key
- branch_key
- location_key
- units_sold
- dollars_sold
- avg_sales

item
- item_key
- item_name
- brand
- type
- supplier_type

branch
- branch_key
- branch_name
- branch_type

Measures

Shipping Fact Table
- time_key
- item_key
- shipper_key
- from_location
- to_location
- dollars_cost
- units_shipped

shipper
- shipper_key
- shipper_name
- location_key
- location_key
- location_type
- country
Cube Definition Syntax (BNF) in DMQL

• Cube Definition (Fact Table)
  ```
  define cube <cube_name> [<dimension_list>]:
  <measure_list>
  ```

• Dimension Definition (Dimension Table)
  ```
  define dimension <dimension_name> as
  (<attribute_or_subdimension_list>)
  ```

• Special Case (Shared Dimension Tables)
  – First time as “cube definition”
  – ```define dimension <dimension_name> as
      <dimension_name_first_time> in cube
      <cube_name_first_time>```
Defining Star Schema in DMQL

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

```sql
define cube sales_snowflake [time, item, branch, location]:
    dollars_sold = sum(sales_in_dollars),
    avg_sales = avg(sales_in_dollars),
    units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city(city_key, province_or_state, country))
```
Defining Fact Constellation in DMQL

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

- **Distributive**: if the result derived by applying the function to \( n \) aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., `count()`, `sum()`, `min()`, `max()`
- **Algebraic**: if it can be computed by an algebraic function with \( M \) arguments (where \( M \) is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., `avg()`, `min_N()`, `standard_deviation()`
- **Holistic**: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., `median()`, `mode()`, `rank()`
A Concept Hierarchy: Dimension (location)

all
region
country
city
office

all
Europe
Germany
Frankfurt
L. Chan

... Spain
Vancouver
M. Wind

... North_America
Canada
Mexico

... Toronto

July 16, 2018
Data Mining: Concepts and Techniques
24
View of Warehouses and Hierarchies

Specification of hierarchies

- Schema hierarchy
  
  \[ \text{day} < \{\text{month} < \text{quarter}; \text{week}\} < \text{year} \]

- Set\_grouping hierarchy
  
  \[ \{1..10\} < \text{inexpensive} \]
Multidimensional Data

• Sales volume as a function of product, month, and region

Dimensions: Product, Location, Time
Hierarchical summarization paths

Industry  Region  Year
Category  Country  Quarter
Product  City  Month  Week
Office  Day
A Sample Data Cube

Total annual sales of TV in U.S.A.
Cuboids Corresponding to the Cube

0-D (apex) cuboid
1-D cuboids
2-D cuboids
3-D (base) cuboid
Browsing a Data Cube

- Visualization
- OLAP capabilities
- Interactive manipulation
Typical OLAP Operations

- **Roll up (drill-up):** summarize data
  - *by climbing up hierarchy or by dimension reduction*
- **Drill down (roll down):** reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- **Slice and dice:** *project and select*
- **Pivot (rotate):**
  - *reorient the cube, visualization, 3D to series of 2D planes*
- **Other operations**
  - **drill across:** involving (across) more than one fact table
  - **drill through:** through the bottom level of the cube to its back-end relational tables (using SQL)
Fig. 3.10 Typical OLAP Operations
A Star-Net Query Model

Each circle is called a footprint.
UNIT-I Data Warehouse

• What is a data warehouse?

• A multi-dimensional data model

• Data warehouse architecture
Design of Data Warehouse: A Business Analysis Framework

• Four views regarding the design of a data warehouse
  – Top-down view
    • allows selection of the relevant information necessary for the data warehouse
  – Data source view
    • exposes the information being captured, stored, and managed by operational systems
  – Data warehouse view
    • consists of fact tables and dimension tables
  – Business query view
    • sees the perspectives of data in the warehouse from the view of end-user
Data Warehouse Design Process

• Top-down, bottom-up approaches or a combination of both
  – **Top-down**: Starts with overall design and planning (mature)
  – **Bottom-up**: Starts with experiments and prototypes (rapid)

• From software engineering point of view
  – **Waterfall**: structured and systematic analysis at each step before proceeding to the next
  – **Spiral**: rapid generation of increasingly functional systems, short turn around time, quick turn around

• Typical data warehouse design process
  – Choose a **business process** to model, e.g., orders, invoices, etc.
  – Choose the **grain (atomic level of data)** of the business process
  – Choose the **dimensions** that will apply to each fact table record
  – Choose the **measure** that will populate each fact table record
**Data Warehouse: A Multi-Tiered Architecture**

- **Data Sources**
  - Operational DBs
  - Other sources

- **Data Storage**
  - Data Warehouse
  - Metadata
  - Extract, Transform, Load, Refresh
  - Data Marts

- **OLAP Engine**
  - Monitor & Integrator
  - OLAP Server

- **Front-End Tools**
  - Analysis
  - Query
  - Reports
  - Data mining

**Data Sources**
July 16, 2018

**Data Mining: Concepts and Techniques**

36
Three Data Warehouse Models

• **Enterprise warehouse**
  – collects all of the information about subjects spanning the entire organization

• **Data Mart**
  – a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    • Independent vs. dependent (directly from warehouse) data mart

• **Virtual warehouse**
  – A set of views over operational databases
  – Only some of the possible summary views may be materialized
Data Warehouse Development: A Recommended Approach

Define a high-level corporate data model

Distributed Data Marts

Data Mart

Data Mart

Model refinement

Model refinement

Multi-Tier Data Warehouse

Enterprise Data Warehouse
Data Warehouse Back-End Tools and Utilities

• Data extraction
  – get data from multiple, heterogeneous, and external sources
• Data cleaning
  – detect errors in the data and rectify them when possible
• Data transformation
  – convert data from legacy or host format to warehouse format
• Load
  – sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions
• Refresh
  – propagate the updates from the data sources to the warehouse
Metadata Repository

- Meta data is the data defining warehouse objects. It stores:
  - Description of the structure of the data warehouse
    - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
  - Operational meta-data
    - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
  - The algorithms used for summarization
  - The mapping from operational environment to the data warehouse
  - Data related to system performance
    - warehouse schema, view and derived data definitions
  - Business data
    - business terms and definitions, ownership of data, charging policies
OLAP Server Architectures

• **Relational OLAP (ROLAP)**
  – Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  – Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  – Greater scalability

• **Multidimensional OLAP (MOLAP)**
  – Sparse array-based multidimensional storage engine
  – Fast indexing to pre-computed summarized data

• **Hybrid OLAP (HOLAP)** (e.g., Microsoft SQLServer)
  – Flexibility, e.g., low level: relational, high-level: array

• **Specialized SQL servers** (e.g., Redbricks)
  – Specialized support for SQL queries over star/snowflake schemas
UNIT-I Data Warehouse

• What is a data warehouse?

• A multi-dimensional data model

• Data warehouse architecture

• **Summary**
Summary: Data Warehouse and OLAP Technology

- Why data warehousing?
- A **multi-dimensional model** of a data warehouse
  - Star schema, snowflake schema, fact constellations
  - A data cube consists of dimensions & measures
- **OLAP** operations: drilling, rolling, slicing, dicing and pivoting
- Data warehouse architecture
- OLAP servers: ROLAP, MOLAP, HOLAP
- Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
- From OLAP to OLAM (on-line analytical mining)
UNIT - II Data Mining

• Motivation: Why data mining?
• What is data mining?
• Data Mining: On what kind of data?
• Data mining functionality
• Classification of data mining systems
Why Data Mining?

• The Explosive Growth of Data: from terabytes to petabytes
  – Data collection and data availability
    • Automated data collection tools, database systems, Web, computerized society
  – Major sources of abundant data
    • Business: Web, e-commerce, transactions, stocks, ...
    • Science: Remote sensing, bioinformatics, scientific simulation, ...
    • Society and everyone: news, digital cameras, YouTube

• We are drowning in data, but starving for knowledge!
• “Necessity is the mother of invention”—Data mining—Automated analysis of massive data sets
Evolution of Sciences

- Before 1600, empirical science
- 1600-1950s, theoretical science
  - Each discipline has grown a theoretical component. Theoretical models often motivate experiments and generalize our understanding.
- 1950s-1990s, computational science
  - Over the last 50 years, most disciplines have grown a third, computational branch (e.g. empirical, theoretical, and computational ecology, or physics, or linguistics.)
  - Computational Science traditionally meant simulation. It grew out of our inability to find closed-form solutions for complex mathematical models.
- 1990-now, data science
  - The flood of data from new scientific instruments and simulations
  - The ability to economically store and manage petabytes of data online
  - The Internet and computing Grid that makes all these archives universally accessible
  - Scientific info. management, acquisition, organization, query, and visualization tasks scale almost linearly with data volumes. Data mining is a major new challenge!
Evolution of Database Technology

• 1960s:
  – Data collection, database creation, IMS and network DBMS
• 1970s:
  – Relational data model, relational DBMS implementation
• 1980s:
  – RDBMS, advanced data models (extended-relational, OO, deductive, etc.)
  – Application-oriented DBMS (spatial, scientific, engineering, etc.)
• 1990s:
  – Data mining, data warehousing, multimedia databases, and Web databases
• 2000s
  – Stream data management and mining
  – Data mining and its applications
  – Web technology (XML, data integration) and global information systems
What Is Data Mining?

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
  - Data mining: a misnomer?
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.
- Watch out: Is everything “data mining”?
  - Simple search and query processing
  - (Deductive) expert systems
Knowledge Discovery (KDD) Process

- Data mining—core of knowledge discovery process

Data Cleaning → Data Integration → Data Warehouse → Selection → Task-relevant Data → Data Mining → Pattern Evaluation → Knowledge Discovery
Data Mining and Business Intelligence

Increasing potential to support business decisions

Decision Making

Data Presentation
Visualization Techniques

Data Mining
Information Discovery

Data Exploration
Statistical Summary, Querying, and Reporting

Data Preprocessing/Integration, Data Warehouses

Data Sources
Paper, Files, Web documents, Scientific experiments, Database Systems

End User
Business Analyst
Data Analyst
DBA

July 16, 2018
Data Mining: Concepts and Techniques
Data Mining: Confluence of Multiple Disciplines

- Database Technology
- Statistics
- Machine Learning
- Pattern Recognition
- Algorithm
- Visualization
- Other Disciplines

July 16, 2018
Why Not Traditional Data Analysis?

• Tremendous amount of data
  – Algorithms must be highly scalable to handle such as tera-bytes of data
• High-dimensionality of data
  – Micro-array may have tens of thousands of dimensions
• High complexity of data
  – Data streams and sensor data
  – Time-series data, temporal data, sequence data
  – Structure data, graphs, social networks and multi-linked data
  – Heterogeneous databases and legacy databases
  – Spatial, spatiotemporal, multimedia, text and Web data
  – Software programs, scientific simulations
• New and sophisticated applications
Multi-Dimensional View of Data Mining

• **Data to be mined**
  – Relational, data warehouse, transactional, stream, object-oriented/relational, active, spatial, time-series, text, multi-media, heterogeneous, legacy, WWW

• **Knowledge to be mined**
  – Characterization, discrimination, association, classification, clustering, trend/deviation, outlier analysis, etc.
  – Multiple/integrated functions and mining at multiple levels

• **Techniques utilized**
  – Database-oriented, data warehouse (OLAP), machine learning, statistics, visualization, etc.

• **Applications adapted**
  – Retail, telecommunication, banking, fraud analysis, bio-data mining, stock market analysis, text mining, Web mining, etc.
Data Mining: Classification Schemes

• General functionality
  – Descriptive data mining
  – Predictive data mining

• Different views lead to different classifications
  – **Data** view: Kinds of data to be mined
  – **Knowledge** view: Kinds of knowledge to be discovered
  – **Method** view: Kinds of techniques utilized
  – **Application** view: Kinds of applications adapted
Data Mining: On What Kinds of Data?

- Database-oriented data sets and applications
  - Relational database, data warehouse, transactional database

- Advanced data sets and advanced applications
  - Data streams and sensor data
  - Time-series data, temporal data, sequence data (incl. bio-sequences)
  - Structure data, graphs, social networks and multi-linked data
  - Object-relational databases
  - Heterogeneous databases and legacy databases
  - Spatial data and spatiotemporal data
  - Multimedia database
  - Text databases
  - The World-Wide Web
Data Mining Functionalities

• Multidimensional concept description: Characterization and discrimination
  – Generalize, summarize, and contrast data characteristics, e.g., dry vs. wet regions

• Frequent patterns, association, correlation vs. causality
  – Diaper $\rightarrow$ Beer [0.5%, 75%] (Correlation or causality?)

• Classification and prediction
  – Construct models (functions) that describe and distinguish classes or concepts for future prediction
    • E.g., classify countries based on (climate), or classify cars based on (gas mileage)
  – Predict some unknown or missing numerical values
Data Mining Functionalities (2)

• Cluster analysis
  – Class label is unknown: Group data to form new classes, e.g., cluster houses to find distribution patterns
  – Maximizing intra-class similarity & minimizing interclass similarity
• Outlier analysis
  – Outlier: Data object that does not comply with the general behavior of the data
  – Noise or exception? Useful in fraud detection, rare events analysis
• Trend and evolution analysis
  – Trend and deviation: e.g., regression analysis
  – Sequential pattern mining: e.g., digital camera → large SD memory
  – Periodicity analysis
  – Similarity-based analysis
• Other pattern-directed or statistical analyses
Why Data Mining?—Potential Applications

• Data analysis and decision support
  – Market analysis and management
    • Target marketing, customer relationship management (CRM), market basket analysis, cross selling, market segmentation
  – Risk analysis and management
    • Forecasting, customer retention, improved underwriting, quality control, competitive analysis
  – Fraud detection and detection of unusual patterns (outliers)

• Other Applications
  – Text mining (news group, email, documents) and Web mining
  – Stream data mining
  – Bioinformatics and bio-data analysis
Ex. 1: Market Analysis and Management

- Where does the data come from?—Credit card transactions, loyalty cards, discount coupons, customer complaint calls, plus (public) lifestyle studies
- Target marketing
  - Find clusters of “model” customers who share the same characteristics: interest, income level, spending habits, etc.
  - Determine customer purchasing patterns over time
- Cross-market analysis—Find associations/co-relations between product sales, & predict based on such association
- Customer profiling—What types of customers buy what products (clustering or classification)
- Customer requirement analysis
  - Identify the best products for different groups of customers
  - Predict what factors will attract new customers
- Provision of summary information
  - Multidimensional summary reports
  - Statistical summary information (data central tendency and variation)
Ex. 2: Corporate Analysis & Risk Management

• Finance planning and asset evaluation
  – cash flow analysis and prediction
  – contingent claim analysis to evaluate assets
  – cross-sectional and time series analysis (financial-ratio, trend analysis, etc.)

• Resource planning
  – summarize and compare the resources and spending

• Competition
  – monitor competitors and market directions
  – group customers into classes and a class-based pricing procedure
  – set pricing strategy in a highly competitive market
Ex. 3: Fraud Detection & Mining Unusual Patterns

• Approaches: Clustering & model construction for frauds, outlier analysis
• Applications: Health care, retail, credit card service, telecomm.
  – **Auto insurance**: ring of collisions
  – **Money laundering**: suspicious monetary transactions
  – **Medical insurance**
    • Professional patients, ring of doctors, and ring of references
    • Unnecessary or correlated screening tests
  – **Telecommunications: phone-call fraud**
    • Phone call model: destination of the call, duration, time of day or week. Analyze patterns that deviate from an expected norm
  – **Retail industry**
    • Analysts estimate that 38% of retail shrink is due to dishonest employees
  – **Anti-terrorism**
KDD Process: Several Key Steps

• Learning the application domain
  – relevant prior knowledge and goals of application
• Creating a target data set: data selection
• Data cleaning and preprocessing: (may take 60% of effort!)
• Data reduction and transformation
  – Find useful features, dimensionality/variable reduction, invariant representation
• Choosing functions of data mining
  – summarization, classification, regression, association, clustering
• Choosing the mining algorithm(s)
• Data mining: search for patterns of interest
• Pattern evaluation and knowledge presentation
  – visualization, transformation, removing redundant patterns, etc.
• Use of discovered knowledge
Are All the “Discovered” Patterns Interesting?

- Data mining may generate thousands of patterns: Not all of them are interesting
  - Suggested approach: Human-centered, query-based, focused mining

- **Interestingness measures**
  - A pattern is *interesting* if it is *easily understood* by humans, *valid* on new or test data with some degree of *certainty*, *potentially useful*, *novel*, or *validates some hypothesis* that a user seeks to confirm

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

• Find all the interesting patterns: **Completeness**
  – Can a data mining system find all the interesting patterns? Do we need to find all of the interesting patterns?
  – Heuristic vs. exhaustive search
  – Association vs. classification vs. clustering

• Search for only interesting patterns: An optimization problem
  – Can a data mining system find only the interesting patterns?
  – Approaches
    • First general all the patterns and then filter out the uninteresting ones
    • Generate only the interesting patterns—mining query optimization
Other Pattern Mining Issues

• Precise patterns vs. approximate patterns
  – Association and correlation mining: possible find sets of precise patterns
  • But approximate patterns can be more compact and sufficient
  • How to find high quality approximate patterns??
  – Gene sequence mining: approximate patterns are inherent
  • How to derive efficient approximate pattern mining algorithms??
• Constrained vs. non-constrained patterns
  – Why constraint-based mining?
  – What are the possible kinds of constraints? How to push constraints into the mining process?
Architecture: Typical Data Mining System

- Graphical User Interface
- Pattern Evaluation
- Data Mining Engine
- Database or Data Warehouse Server
- Knowledge Base

Data cleaning, integration, and selection:
- Database
- Data Warehouse
- World-Wide Web
- Other Info Repositories

July 16, 2018
UNIT-II DATA PREPROCESSING

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary
Why Data Preprocessing?

• Data in the real world is dirty
  – **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    • e.g., occupation=""
  – **noisy**: containing errors or outliers
    • e.g., Salary="-10"
  – **inconsistent**: containing discrepancies in codes or names
    • e.g., Age="42" Birthday="03/07/1997"
    • e.g., Was rating "1,2,3", now rating "A, B, C"
    • e.g., discrepancy between duplicate records
Why Is Data Dirty?

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

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

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

• Duplicate records also need data cleaning
Why Is Data Preprocessing Important?

• No quality data, no quality mining results!
  – Quality decisions must be based on quality data
    • e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  – Data warehouse needs consistent integration of quality data
• Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse
Multi-Dimensional Measure of Data Quality

• A well-accepted multidimensional view:
  – Accuracy
  – Completeness
  – Consistency
  – Timeliness
  – Believability
  – Value added
  – Interpretability
  – Accessibility

• Broad categories:
  – Intrinsic, contextual, representational, and accessibility
Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies

- **Data integration**
  - Integration of multiple databases, data cubes, or files

- **Data transformation**
  - Normalization and aggregation

- **Data reduction**
  - Obtains reduced representation in volume but produces the same or similar analytical results

- **Data discretization**
  - Part of data reduction but with particular importance, especially for numerical data
Forms of Data Preprocessing

Data Cleaning

[water to clean dirty-looking data] → ['clean'-looking data]
[shave soap suds on data]

Data Integration


Data Transformation

-2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.59, 0.48

Data Reduction

\[
\begin{array}{cccc}
T1 & A1 & A2 & A3 \\
T2 & A3 & ... & A126 \\
T3 & & & \\
T4 & & & \\
... & & & \\
T2000 & & & \\
\end{array}
\rightarrow
\begin{array}{cccc}
T1 & A1 & A3 & A115 \\
T4 & & ... & \\
T1456 & & & \\
\end{array}
\]
UNIT-II DATA PREPROCESSING

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary
Mining Data Descriptive Characteristics

- **Motivation**
  - To better understand the data: central tendency, variation and spread

- **Data dispersion characteristics**
  - median, max, min, quantiles, outliers, variance, etc.

- **Numerical dimensions** correspond to sorted intervals
  - Data dispersion: analyzed with multiple granularities of precision
  - Boxplot or quantile analysis on sorted intervals

- **Dispersion analysis on computed measures**
  - Folding measures into numerical dimensions
  - Boxplot or quantile analysis on the transformed cube
Measuring the Central Tendency

• **Mean (algebraic measure) (sample vs. population):**
  - Weighted arithmetic mean:
  - Trimmed mean: chopping extreme values

• **Median**: A holistic measure
  - Middle value if odd number of values, or average of the middle two values otherwise
  - Estimated by interpolation (for grouped data):

• **Mode**
  - Value that occurs most frequently in the data
  - Unimodal, bimodal, trimodal
  - Empirical formula: \( \frac{\text{mean} - \text{mode}}{\text{mean} - \text{median}} = 3 \times \left( \frac{\text{mean} - \text{median}}{\text{mean} - \text{median}} \right) \)

\[ \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \mu = \frac{\sum x}{N} \]
\[ \bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \]
\[ \text{median} = L_1 + \left( \frac{n / 2 - (\sum f) l}{f_{\text{median}}} \right) c \]
Symmetric vs. Skewed Data

- Median, mean and mode of symmetric, positively and negatively skewed data
Measuring the Dispersion of Data

- Quartiles, outliers and boxplots
  - **Quartiles**: $Q_1$ (25th percentile), $Q_3$ (75th percentile)
  - **Inter-quartile range**: $IQR = Q_3 - Q_1$
  - **Five number summary**: min, $Q_1$, M, $Q_3$, max
  - **Boxplot**: ends of the box are the quartiles, median is marked, whiskers, and plot outlier individually
  - **Outlier**: usually, a value higher/lower than $1.5 \times IQR$

- Variance and standard deviation (**sample**: $s$, **population**: $\sigma$)
  - **Variance**: (algebraic, scalable computation)
    \[
    s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 = \frac{1}{n-1} \left[ \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left( \sum_{i=1}^{n} x_i \right)^2 \right]
    \]
    \[
    \sigma^2 = \frac{1}{N} \sum_{i=1}^{n} (x_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^{n} x_i^2 - \mu^2
    \]
  - **Standard deviation** $s$ (**or** $\sigma$) is the square root of variance $s^2$ (**or** $\sigma^2$)
Properties of Normal Distribution Curve

- The normal (distribution) curve
  - From $\mu - \sigma$ to $\mu + \sigma$: contains about 68% of the measurements ($\mu$: mean, $\sigma$: standard deviation)
  - From $\mu - 2\sigma$ to $\mu + 2\sigma$: contains about 95% of it
  - From $\mu - 3\sigma$ to $\mu + 3\sigma$: contains about 99.7% of it
Boxplot Analysis

• Five-number summary of a distribution:
  Minimum, Q1, M, Q3, Maximum

• Boxplot
  – Data is represented with a box
  – The ends of the box are at the first and third quartiles, i.e., the height of the box is IRQ
  – The median is marked by a line within the box
  – Whiskers: two lines outside the box extend to Minimum and Maximum
Visualization of Data Dispersion: Boxplot Analysis
Histogram Analysis

• Graph displays of basic statistical class descriptions
  – Frequency histograms
    • A univariate graphical method
    • Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots quantile information
  - For a data $x_i$ data sorted in increasing order, $f_i$ indicates that approximately $100 \times f_i\%$ of the data are below or equal to the value $x_i$
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Allows the user to view whether there is a shift in going from one distribution to another
Scatter plot

- Provides a first look at bivariate data to see clusters of points, outliers, etc.
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane.
Loess Curve

- Adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence.
- Loess curve is fitted by setting two parameters: a smoothing parameter, and the degree of the polynomials that are fitted by the regression.
Positively and Negatively Correlated Data
Not Correlated Data
Graphic Displays of Basic Statistical Descriptions

- Histogram: (shown before)
- Boxplot: (covered before)
- Quantile plot: each value $x_i$ is paired with $f_i$, indicating that approximately 100 $f_i$% of data are $\leq x_i$
- Quantile-quantile (q-q) plot: graphs the quantiles of one univariant distribution against the corresponding quantiles of another
- Scatter plot: each pair of values is a pair of coordinates and plotted as points in the plane
- Loess (local regression) curve: add a smooth curve to a scatter plot to provide better perception of the pattern of dependence
UNIT-II DATA PREPROCESSING

• Why preprocess the data?
• Descriptive data summarization
• Data cleaning
• Data integration and transformation
• Data reduction
• Discretization and concept hierarchy generation
• Summary
Data Cleaning

• Importance
  – “Data cleaning is one of the three biggest problems in data warehousing” — Ralph Kimball
  – “Data cleaning is the number one problem in data warehousing” — DCI survey

• Data cleaning tasks
  – Fill in missing values
  – Identify outliers and smooth out noisy data
  – Correct inconsistent data
  – Resolve redundancy caused by data integration
Missing Data

• Data is not always available
  – E.g., many tuples have no recorded value for several attributes, such as customer income in sales data

• Missing data may be due to
  – equipment malfunction
  – inconsistent with other recorded data and thus deleted
  – data not entered due to misunderstanding
  – certain data may not be considered important at the time of entry
  – not register history or changes of the data

• Missing data may need to be inferred.
How to Handle Missing Data?

• Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.

• Fill in the missing value manually: tedious + infeasible?

• Fill in it automatically with
  – a global constant: e.g., “unknown”, a new class?!
  – the attribute mean
  – the attribute mean for all samples belonging to the same class: smarter
  – the most probable value: inference-based such as Bayesian formula or decision tree
Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which requires data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data
How to Handle Noisy Data?

- **Binning**
  - first sort data and partition into (equal-frequency) bins
  - then one can *smooth by bin means*, *smooth by bin median*, *smooth by bin boundaries*, etc.

- **Regression**
  - smooth by fitting the data into regression functions

- **Clustering**
  - detect and remove outliers

- **Combined computer and human inspection**
  - detect suspicious values and check by human (e.g., deal with possible outliers)
Simple Discretization Methods: Binning

• **Equal-width** (distance) partitioning
  - Divides the range into $N$ intervals of equal size: uniform grid
  - if $A$ and $B$ are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well

• **Equal-depth** (frequency) partitioning
  - Divides the range into $N$ intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky
Binning Methods for Data Smoothing

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
  - Partition into equal-frequency (equi-depth) bins:
    - Bin 1: 4, 8, 9, 15
    - Bin 2: 21, 21, 24, 25
    - Bin 3: 26, 28, 29, 34
  - Smoothing by bin means:
    - Bin 1: 9, 9, 9, 9
    - Bin 2: 23, 23, 23, 23
    - Bin 3: 29, 29, 29, 29
  - Smoothing by bin boundaries:
    - Bin 1: 4, 4, 4, 15
    - Bin 2: 21, 21, 25, 25
    - Bin 3: 26, 26, 26, 34
Regression

\[ y = x + 1 \]

\[ Y_1 \]
\[ Y_1' \]
\[ X_1 \]
\[ x \]
\[ y \]
Cluster Analysis
Data Cleaning as a Process

• Data discrepancy detection
  – Use metadata (e.g., domain, range, dependency, distribution)
  – Check field overloading
  – Check uniqueness rule, consecutive rule and null rule
  – Use commercial tools
    • Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
    • Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)

• Data migration and integration
  – Data migration tools: allow transformations to be specified
  – ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface

• Integration of the two processes
  – Iterative and interactive (e.g., Potter’s Wheels)
UNIT-II DATA PREPROCESSING

• Why preprocess the data?

• Data cleaning

• Data integration and transformation

• Data reduction

• Discretization and concept hierarchy generation

• Summary
Data Integration

• Data integration:
  – Combines data from multiple sources into a coherent store
• Schema integration: e.g., A.cust-id ≡ B.cust-#
  – Integrate metadata from different sources
• Entity identification problem:
  – Identify real world entities from multiple data sources, e.g.,
    Bill Clinton = William Clinton
• Detecting and resolving data value conflicts
  – For the same real world entity, attribute values from different
    sources are different
  – Possible reasons: different representations, different scales,
    e.g., metric vs. British units
Handling Redundancy in Data Integration

• Redundant data occur often when integration of multiple databases
  – *Object identification*: The same attribute or object may have different names in different databases
  – *Derivable data*: One attribute may be a “derived” attribute in another table, e.g., annual revenue

• Redundant attributes may be able to be detected by *correlation analysis*

• Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality
Correlation Analysis (Numerical Data)

- Correlation coefficient (also called Pearson’s product moment coefficient)

\[
r_{A,B} = \frac{\sum (A - \bar{A})(B - \bar{B})}{(n - 1) \sigma_A \sigma_B} = \frac{\sum (AB) - n \bar{A} \bar{B}}{(n - 1) \sigma_A \sigma_B}
\]

where \( n \) is the number of tuples, \( \bar{A} \) and \( \bar{B} \) are the respective means of \( A \) and \( B \), \( \sigma_A \) and \( \sigma_B \) are the respective standard deviation of \( A \) and \( B \), and \( \Sigma(AB) \) is the sum of the \( AB \) cross-product.

- If \( r_{A,B} > 0 \), \( A \) and \( B \) are positively correlated (\( A \)’s values increase as \( B \)’s). The higher, the stronger correlation.

- \( r_{A,B} = 0 \): independent; \( r_{A,B} < 0 \): negatively correlated
Correlation Analysis (Categorical Data)

- $X^2$ (chi-square) test

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

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

<table>
<thead>
<tr>
<th></th>
<th>Play chess</th>
<th>Not play chess</th>
<th>Sum (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like science fiction</td>
<td>250(90)</td>
<td>200(360)</td>
<td>450</td>
</tr>
<tr>
<td>Not like science fiction</td>
<td>50(210)</td>
<td>1000(840)</td>
<td>1050</td>
</tr>
<tr>
<td>Sum(col.)</td>
<td>300</td>
<td>1200</td>
<td>1500</td>
</tr>
</tbody>
</table>

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

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

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

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones
Data Transformation: Normalization

• Min-max normalization: to \([\text{new}_\text{min}_A, \text{new}_\text{max}_A]\)

\[
v' = \frac{v - \text{min}_A}{\text{max}_A - \text{min}_A} (\text{new}_\text{max}_A - \text{new}_\text{min}_A) + \text{new}_\text{min}_A
\]

– Ex. Let income range $12,000 to $98,000 normalized to [0.0, 1.0]. Then $73,000 is mapped to

\[
\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716
\]

• Z-score normalization (\(\mu\): mean, \(\sigma\): standard deviation):

\[
v' = \frac{v - \mu_A}{\sigma_A}
\]

– Ex. Let \(\mu = 54,000, \sigma = 16,000\). Then

\[
\frac{73,600 - 54,000}{16,000} = 1.225
\]

• Normalization by decimal scaling

\[
v' = \frac{v}{10^j}
\]

Where \(j\) is the smallest integer such that Max(|\(v'\)|) < 1
UNIT-II DATA PREPROCESSING

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- **Data reduction**
- Discretization and concept hierarchy generation
- Summary
Data Reduction Strategies

• Why data reduction?
  – A database/data warehouse may store terabytes of data
  – Complex data analysis/mining may take a very long time to run on the complete data set

• Data reduction
  – Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results

• Data reduction strategies
  – Data cube aggregation:
  – Dimensionality reduction — e.g., remove unimportant attributes
  – Data Compression
  – Numerosity reduction — e.g., fit data into models
  – Discretization and concept hierarchy generation
Data Cube Aggregation

• The lowest level of a data cube (base cuboid)
  – The aggregated data for an individual entity of interest
  – E.g., a customer in a phone calling data warehouse

• Multiple levels of aggregation in data cubes
  – Further reduce the size of data to deal with

• Reference appropriate levels
  – Use the smallest representation which is enough to solve the task

• Queries regarding aggregated information should be answered using data cube, when possible
Attribute Subset Selection

• Feature selection (i.e., attribute subset selection):
  – Select a minimum set of features such that the probability
distribution of different classes given the values for those
features is as close as possible to the original distribution
given the values of all features
  – reduce # of patterns in the patterns, easier to understand

• Heuristic methods (due to exponential # of choices):
  – Step-wise forward selection
  – Step-wise backward elimination
  – Combining forward selection and backward elimination
  – Decision-tree induction
Example of Decision Tree Induction

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

\[
\begin{array}{c}
\text{A4?} \\
\text{A1?} \\
\text{A6?} \\
\text{Class 1} \\
\text{Class 2} \\
\text{Class 1} \\
\text{Class 2}
\end{array}
\]

\[\rightarrow\text{ Reduced attribute set: } \{A1, A4, A6\}\]
Heuristic Feature Selection Methods

• There are $2^d$ possible sub-features of $d$ features
• Several heuristic feature selection methods:
  – Best single features under the feature independence assumption: choose by significance tests
  – Best step-wise feature selection:
    • The best single-feature is picked first
    • Then next best feature condition to the first, ...
  – Step-wise feature elimination:
    • Repeatedly eliminate the worst feature
  – Best combined feature selection and elimination
  – Optimal branch and bound:
    • Use feature elimination and backtracking
Data Compression

• String compression
  – There are extensive theories and well-tuned algorithms
  – Typically lossless
  – But only limited manipulation is possible without expansion

• Audio/video compression
  – Typically lossy compression, with progressive refinement
  – Sometimes small fragments of signal can be reconstructed without reconstructing the whole

• Time sequence is not audio
  – Typically short and vary slowly with time
Data Compression

Original Data

Compressed Data

Original Data Approximated

lossless

lossy
Dimensionality Reduction: Wavelet Transformation

• Discrete wavelet transform (DWT): linear signal processing, multi-resolutional analysis
• Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
• Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
• Method:
  – Length, $L$, must be an integer power of 2 (padding with 0’s, when necessary)
  – Each transform has 2 functions: smoothing, difference
  – Applies to pairs of data, resulting in two set of data of length $L/2$
  – Applies two functions recursively, until reaches the desired length
DWT for Image Compression

- Image
  - Low Pass
  - High Pass
    - Low Pass
    - High Pass
      - Low Pass
      - High Pass

July 16, 2018
Dimensionality Reduction: Principal Component Analysis (PCA)

• Given $N$ data vectors from $n$-dimensions, find $k \leq n$ orthogonal vectors (principal components) that can be best used to represent data

• Steps
  – Normalize input data: Each attribute falls within the same range
  – Compute $k$ orthonormal (unit) vectors, i.e., principal components
  – Each input data (vector) is a linear combination of the $k$ principal component vectors
  – The principal components are sorted in order of decreasing “significance” or strength
  – Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)

• Works for numeric data only
• Used when the number of dimensions is large
Principal Component Analysis
Numerosity Reduction

• Reduce data volume by choosing alternative, smaller forms of data representation

• Parametric methods
  – Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  – Example: Log-linear models—obtain value at a point in m-D space as the product on appropriate marginal subspaces

• Non-parametric methods
  – Do not assume models
  – Major families: histograms, clustering, sampling
Data Reduction Method (1): Regression and Log-Linear Models

• Linear regression: Data are modeled to fit a straight line
  – Often uses the least-square method to fit the line

• Multiple regression: allows a response variable $Y$ to be modeled as a linear function of multidimensional feature vector

• Log-linear model: approximates discrete multidimensional probability distributions
Regress Analysis and Log-Linear Models

• **Linear regression:** $Y = w X + b$
  
  – Two regression coefficients, $w$ and $b$, specify the line and are to be estimated by using the data at hand
  
  – Using the least squares criterion to the known values of $Y_1$, $Y_2$, ..., $X_1$, $X_2$, ....

• **Multiple regression:** $Y = b_0 + b_1 X_1 + b_2 X_2$.
  
  – Many nonlinear functions can be transformed into the above

• **Log-linear models:**
  
  – The multi-way table of joint probabilities is approximated by a product of lower-order tables
  
  – Probability: $p(a, b, c, d) = \alpha_{ab} \beta_{ac} \chi_{ad} \delta_{bcd}$
Data Reduction Method (2): Histograms

• Divide data into buckets and store average (sum) for each bucket

• Partitioning rules:
  – Equal-width: equal bucket range
  – Equal-frequency (or equal-depth)
  – V-optimal: with the least $\text{histogram}_v$ variance (weighted sum of the original values that each bucket represents)
  – MaxDiff: set bucket boundary between each pair for pairs have the $\beta-1$ largest differences
Data Reduction Method (3): Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 7
Data Reduction Method (4): Sampling

- Sampling: obtaining a small sample $s$ to represent the whole data set $N$
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Note: Sampling may not reduce database I/Os (page at a time)
Sampling: with or without Replacement

Raw Data

SRSWOR
(simple random sample without replacement)

SRSWR
Sampling: Cluster or Stratified Sampling

Raw Data

Cluster/Stratified Sample
UNIT-II DATA PREPROCESSING

• Why preprocess the data?
• Data cleaning
• Data integration and transformation
• Data reduction
• Discretization and concept hierarchy generation
• Summary
Discretization

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

• Discretization:
  – Divide the range of a continuous attribute into intervals
  – Some classification algorithms only accept categorical attributes.
  – Reduce data size by discretization
  – Prepare for further analysis
Discretization and Concept Hierarchy

• Discretization
  – Reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals
  – Interval labels can then be used to replace actual data values
  – Supervised vs. unsupervised
  – Split (top-down) vs. merge (bottom-up)
  – Discretization can be performed recursively on an attribute

• Concept hierarchy formation
  – Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as young, middle-aged, or senior)
Discretization and Concept Hierarchy Generation for Numeric Data

• Typical methods: All the methods can be applied recursively
  – Binning (covered above)
    • Top-down split, unsupervised,
  – Histogram analysis (covered above)
    • Top-down split, unsupervised
  – Clustering analysis (covered above)
    • Either top-down split or bottom-up merge, unsupervised
  – Entropy-based discretization: supervised, top-down split
  – Interval merging by $\chi^2$ Analysis: unsupervised, bottom-up merge
  – Segmentation by natural partitioning: top-down split, unsupervised
Entropy-Based Discretization

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

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

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

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

where $p_i$ is the probability of class $i$ in $S_1$

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization

- The process is recursively applied to partitions obtained until some stopping criterion is met

- Such a boundary may reduce data size and improve classification accuracy
Interval Merge by $\chi^2$ Analysis

- Merging-based (bottom-up) vs. splitting-based methods
- Merge: Find the best neighboring intervals and merge them to form larger intervals recursively
- ChiMerge [Kerber AAAI 1992, See also Liu et al. DMKD 2002]
  - Initially, each distinct value of a numerical attr. A is considered to be one interval
  - $\chi^2$ tests are performed for every pair of adjacent intervals
  - Adjacent intervals with the least $\chi^2$ values are merged together, since low $\chi^2$ values for a pair indicate similar class distributions
  - This merge process proceeds recursively until a predefined stopping criterion is met (such as significance level, max-interval, max inconsistency, etc.)
Segmentation by Natural Partitioning

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

Step 1:
- $351  - $159  \quad \text{profit}  \quad $1,838  \quad $4,700
  \text{Min} \quad \text{Low (i.e, 5\%-tile)} \quad \text{High (i.e, 95\%-tile)} \quad \text{Max}

Step 2:
- $400 - $5,000
  \text{msd}=1,000 \quad \text{Low}=-$1,000 \quad \text{High}=$2,000

Step 3:
- $400 - $5,000
  \text{count}
  \text{(-$1,000 - $2,000)}
    \text{(-$1,000 - 0)}
    \text{(0 - $1,000)}
    \text{($1,000 - $2,000)}

Step 4:
- $400 - $5,000
  \text{(-$400 - 0)}
  \text{(-$400 - $300)}
  \text{(-$300 - $200)}
  \text{(-$200 - $100)}
  \text{(-$100 -)}
  \text{(0 - $200)}
  \text{($200 - $400)}
  \text{($400 - $600)}
  \text{($600 - $800)}
  \text{($800 - $1,000)}
  \text{($1,000 - $2,000)}
  \text{($1,200 - $1,400)}
  \text{($1,400 - $1,600)}
  \text{($1,600 - $1,800)}
  \text{($1,800 - $2,000)}
  \text{($2,000 - $3,000)}
  \text{($3,000 - $4,000)}
  \text{($4,000 - $5,000)}

July 01, 2018

Data Mining: Concepts and Techniques

136
Concept Hierarchy Generation for Categorical Data

• Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  – street < city < state < country
• Specification of a hierarchy for a set of values by explicit data grouping
  – \{Urbana, Champaign, Chicago\} < Illinois
• Specification of only a partial set of attributes
  – E.g., only street < city, not others
• Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  – E.g., for a set of attributes: \{street, city, state, country\}
Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set:

- The attribute with the most distinct values is placed at the lowest level of the hierarchy.
- Exceptions, e.g., weekday, month, quarter, year.

- **Country**: 15 distinct values
- **Province or State**: 365 distinct values
- **City**: 3567 distinct values
- **Street**: 674,339 distinct values
UNIT-II DATA PREPROCESSING

• Why preprocess the data?
• Data cleaning
• Data integration and transformation
• Data reduction
• Discretization and concept hierarchy generation
• Summary
Summary

• Data preparation or preprocessing is a big issue for both data warehousing and data mining

• Descriptive data summarization is need for quality data preprocessing

• Data preparation includes
  – Data cleaning and data integration
  – Data reduction and feature selection
  – Discretization

• A lot a methods have been developed but data preprocessing still an active area of research
UNIT – III  Association Rules

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Summary
What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining

- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
- Classification: associative classification
- Cluster analysis: frequent pattern-based clustering
- Data warehousing: iceberg cube and cube-gradient
- Semantic data compression: fascicles
- Broad applications
Basic Concepts: Frequent Patterns and Association Rules

- Itemset $X = \{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - **Support**, $s$, probability that a transaction contains $X \cup Y$
  - **Confidence**, $c$, conditional probability that a transaction having $X$ also contains $Y$

Let $\text{sup}_{\text{min}} = 50\%$, $\text{conf}_{\text{min}} = 50\%$

Freq. Pat.: \{A:3, B:3, D:4, E:3, AD:3\}

Association rules:
- $A \rightarrow D$ (60\%, 100\%)
- $D \rightarrow A$ (60\%, 75\%)
Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., \( \{a_1, \ldots, a_{100}\} \) contains \( \binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{0} = 2^{100} - 1 = 1.27 \times 10^{30} \) sub-patterns!

- Solution: Mine **closed patterns and max-patterns** instead

- An itemset \( X \) is **closed** if \( X \) is frequent and there exists no super-pattern \( Y \supseteq X \), with the same support as \( X \) (proposed by Pasquier, et al. @ ICDT'99)

- An itemset \( X \) is a **max-pattern** if \( X \) is frequent and there exists no frequent super-pattern \( Y \supseteq X \) (proposed by Bayardo @ SIGMOD'98)

- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules
Closed Patterns and Max-Patterns

- **Exercise.** \( \text{DB} = \{<a_1, \ldots, a_{100}>, <a_1, \ldots, a_{50}\} \)
  - \( \text{Min\_sup} = 1. \)
- What is the set of **closed itemset**?
  - \( <a_1, \ldots, a_{100}>: 1 \)
  - \( <a_1, \ldots, a_{50}>: 2 \)
- What is the set of **max-pattern**?
  - \( <a_1, \ldots, a_{100}>: 1 \)
- What is the set of **all patterns**?
  - !!
UNIT – III  Association Rules

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Summary
Scalable Methods for Mining Frequent Patterns

- The **downward closure** property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If \{beer, diaper, nuts\} is frequent, so is \{beer, diaper\}
  - i.e., every transaction having \{beer, diaper, nuts\} also contains \{beer, diaper\}

- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB‘94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD‘00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM‘02)
Apriori: A Candidate Generation-and-Test Approach

- **Apriori pruning principle**: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD'94)

- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
The Apriori Algorithm—An Example

Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

Sup_{min} = 2

1st scan

C_1

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{D}</td>
<td>1</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

L_1

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

2nd scan

C_2

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td>1</td>
</tr>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{A, E}</td>
<td>1</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

L_2

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
</tr>
<tr>
<td>{A, C}</td>
</tr>
<tr>
<td>{A, E}</td>
</tr>
<tr>
<td>{B, C}</td>
</tr>
<tr>
<td>{B, E}</td>
</tr>
<tr>
<td>{C, E}</td>
</tr>
</tbody>
</table>

3rd scan

C_3

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
</tr>
</tbody>
</table>

L_3

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

July 16, 2018
Data Mining: Concepts and Techniques
The Apriori Algorithm

- **Pseudo-code:**
  
  $C_k$: Candidate itemset of size $k$
  $L_k$: frequent itemset of size $k$

  \[
  L_1 = \{\text{frequent items}\};
  \]

  \[
  \text{for } (k = 1; L_k \neq \emptyset; k++) \text{ do begin}
  \]

  \[
  C_{k+1} = \text{candidates generated from } L_k;
  \]

  \[
  \text{for each transaction } t \text{ in database do}
  \]

  \[
  \text{increment the count of all candidates in } C_{k+1}
  \]

  \[
  \text{that are contained in } t
  \]

  \[
  L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}
  \]

  \[
  \text{end}
  \]

  \[
  \text{return } \bigcup_k L_k;
  \]
Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining $L_k$
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - $L_3=\{abc, abd, acd, ace, bcd\}$
  - Self-joining: $L_3 \times L_3$
    - $abcd$ from $abc$ and $abd$
    - $acde$ from $acd$ and $ace$
  - Pruning:
    - $acde$ is removed because $ade$ is not in $L_3$
    - $C_4=\{abcd\}$
How to Generate Candidates?

- Suppose the items in $L_{k-1}$ are listed in an order
- Step 1: self-joining $L_{k-1}$
  
  insert into $C_k$
  
  select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
  
  from $L_{k-1} p$, $L_{k-1} q$
  
  where $p.item_1=q.item_1, ..., p.item_{k-2}=q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$

- Step 2: pruning
  
  forall itemsets $c$ in $C_k$ do
  
  forall (k-1)-subsets $s$ of $c$ do
  
  if ($s$ is not in $L_{k-1}$) then delete $c$ from $C_k$
How to Count Supports of Candidates?

Why counting supports of candidates a problem?
- The total number of candidates can be very huge
- One transaction may contain many candidates

Method:
- Candidate itemsets are stored in a hash-tree
- Leaf node of hash-tree contains a list of itemsets and counts
- Interior node contains a hash table
- Subset function: finds all the candidates contained in a transaction
Example: Counting Supports of Candidates

Subset function

1, 4, 7
2, 5, 8
3, 6, 9

Transaction: 1 2 3 5 6

1 + 2 3 5 6
1 2 3 5 6
1 2 + 3 5 6
1 3 + 5 6
1 4 5
1 2 4
1 2 5
1 5 9
1 3 6 3 4 5
2 3 4 5 6 7
3 5 6 3 5 7 6 8 9
3 6 7 3 6 8

July 16, 2018
Efficient Implementation of Apriori in SQL

- Hard to get good performance out of pure SQL (SQL-92) based approaches alone
- Make use of object-relational extensions like UDFs, BLOBs, Table functions etc.
  - Get orders of magnitude improvement
Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates
Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. *An efficient algorithm for mining association in large databases.* In *VLDB'95*
DHP: Reduce the Number of Candidates

- A $k$-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries: \{ab, ad, ae\} \{bd, be, de\} ...
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of \{ab, ad, ae\} is below support threshold
Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check $abcd$ instead of $ab, ac, ..., etc.$
- Scan database again to find missed frequent patterns
- H. Toivonen. *Sampling large databases for association rules.* In *VLDB'96*
DIC: Reduce Number of Scans

- Once both A and D are determined frequent, the counting of AD begins.
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins.

Transactions

1 itemsets
2 itemsets
...

1-itemsets
2-items
3-items

Itemset lattice


DIC

Apriori

Data Mining: Concepts and Techniques
Bottleneck of Frequent-pattern Mining

- Multiple database scans are **costly**
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1i_2...i_{100}$
    - # of scans: 100
    - # of Candidates: $\binom{100}{1} + \binom{100}{2} + ... + \binom{10}{0} = 2^{100} - 1 = 1.27 \times 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?
Mining Frequent Patterns Without Candidate Generation

- Grow long patterns from short ones using local frequent items
  - \[\text{\texttt{abc}}\] is a frequent pattern
  - Get all transactions having \[\text{\texttt{abc}}\]: \(\text{DB} | \text{abc}\)
  - \(\text{\texttt{d}}\) is a local frequent item in \(\text{DB} | \text{abc}\) → abcd is a frequent pattern
Construct FP-tree from a Transaction Database

<table>
<thead>
<tr>
<th>TID</th>
<th>Items bought</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f, a, c, d, g, i, m, p}</td>
<td>{f, c, a, m, p}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c, f, l, m, o}</td>
<td>{f, c, a, b, m}</td>
</tr>
<tr>
<td>300</td>
<td>{b, f, h, j, o, w}</td>
<td>{f, b}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, k, s, p}</td>
<td>{c, b, p}</td>
</tr>
<tr>
<td>500</td>
<td>{a, f, c, e, l, p, m, n}</td>
<td>{f, c, a, m, p}</td>
</tr>
</tbody>
</table>

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

\text{F-list}=f-c-a-b-m-p
Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - For Connect-4 DB, compression ratio could be over 100
Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - ...
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency
Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item \( p \)
- Accumulate all of transformed prefix paths of item \( p \) to form \( p \)'s conditional pattern base

### Header Table

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>4</td>
</tr>
<tr>
<td>( c )</td>
<td>4</td>
</tr>
<tr>
<td>( a )</td>
<td>3</td>
</tr>
<tr>
<td>( b )</td>
<td>3</td>
</tr>
<tr>
<td>( m )</td>
<td>3</td>
</tr>
<tr>
<td>( p )</td>
<td>3</td>
</tr>
</tbody>
</table>

### Conditional pattern bases

<table>
<thead>
<tr>
<th>Item</th>
<th>Cond. Pattern Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c )</td>
<td>( f:3 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( fc:3 )</td>
</tr>
<tr>
<td>( b )</td>
<td>( fca:1, f:1, c:1 )</td>
</tr>
<tr>
<td>( m )</td>
<td>( fca:2, fcab:1 )</td>
</tr>
<tr>
<td>( p )</td>
<td>( fcam:2, cb:1 )</td>
</tr>
</tbody>
</table>
From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

**Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>m</td>
<td>3</td>
</tr>
<tr>
<td>p</td>
<td>3</td>
</tr>
</tbody>
</table>

**m-conditional pattern base:**

\[ fca:2, fcab:1 \]

All frequent patterns relate to \( m \)

- \{\}  
- \( f:3 \) \( fm, cm, am, fcm, fam, cam, fcam \)  
- \( c:3 \) \( fcam \)  
- \( a:3 \)
Recursion: Mining Each Conditional FP-tree

Cond. pattern base of $\neg am||$ (fc:3)

\[
\{\}
\quad f:3
\quad c:3
\quad am\text{-conditional FP-tree}
\]

Cond. pattern base of $\neg cm||$ (f:3)

\[
\{\}
\quad f:3
\quad cm\text{-conditional FP-tree}
\]

Cond. pattern base of $\neg cam||$ (f:3)

\[
\{\}
\quad f:3
\quad cam\text{-conditional FP-tree}
\]
A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts

\[
\begin{align*}
\{\} & \quad \{\} \\
\vdots & \quad r_1 = \quad \vdots \\
\quad \vdots & \quad \vdots \\
\end{align*}
\]

\[
\begin{align*}
\{\} & \quad \{\}
\end{align*}
\]

\[
\begin{align*}
\quad r_1 & \quad + \\
\quad \vdots & \quad \vdots \\
\end{align*}
\]

\[
\begin{align*}
\quad \vdots & \quad \vdots \\
\end{align*}
\]
Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition

- Method
  - For each frequent 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
Scaling FP-growth by DB Projection

- FP-tree cannot fit in memory?—DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. Partition projection techniques
  - Parallel projection is space costly
Partition-based Projection

- Parallel projection needs a lot of disk space
- Partition projection saves it

```
<table>
<thead>
<tr>
<th>Tran. DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcamp</td>
</tr>
<tr>
<td>fcabm</td>
</tr>
<tr>
<td>fb</td>
</tr>
<tr>
<td>cbp</td>
</tr>
<tr>
<td>fcamp</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcam</td>
</tr>
<tr>
<td>cb</td>
</tr>
<tr>
<td>fcam</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcab</td>
</tr>
<tr>
<td>fca</td>
</tr>
<tr>
<td>fca</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
</tr>
<tr>
<td>cb</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>am-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fc</td>
</tr>
<tr>
<td>fc</td>
</tr>
<tr>
<td>fc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cm-proj DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
</tr>
<tr>
<td>f</td>
</tr>
</tbody>
</table>
```

July 16, 2018
FP-Growth vs. Apriori: Scalability With the Support Threshold

Data set T25I20D10K
FP-Growth vs. Tree-Projection: Scalability with the Support Threshold

Data set T25I20D100K
Why Is FP-Growth the Winner?

- **Divide-and-conquer:**
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases

- **Other factors**
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching
Implications of the Methodology

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD'00)
- Mining sequential patterns
  - FreeSpan (KDD'00), PrefixSpan (ICDE'01)
- Constraint-based mining of frequent patterns
  - Convertible constraints (KDD'00, ICDE'01)
- Computing iceberg data cubes with complex measures
  - H-tree and H-cubing algorithm (SIGMOD'01)
MaxMiner: Mining Max-patterns

- 1\textsuperscript{st} scan: find frequent items
  - A, B, C, D, E
- 2\textsuperscript{nd} scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE, DE,

Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

- R. Bayardo. Efficiently mining long patterns from databases. In \textit{SIGMOD'98}
Mining Frequent Closed Patterns: CLOSET

- **Flist**: list of all frequent items in support ascending order
  - Flist: d-a-f-e-c
- Divide search space
  - Patterns having d
    - Patterns having d but no a, etc.
- Find frequent closed pattern recursively
  - Every transaction having d also has cfa → cfad is a frequent closed pattern
- J. Pei, J. Han & R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00.

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>20</td>
<td>a, b, e</td>
</tr>
<tr>
<td>30</td>
<td>c, e, f</td>
</tr>
<tr>
<td>40</td>
<td>a, c, d, f</td>
</tr>
<tr>
<td>50</td>
<td>c, e, f</td>
</tr>
</tbody>
</table>
CLOSET+: Mining Closed Itemsets by Pattern-Growth

- Itemset merging: if Y appears in every occurrence of X, then Y is merged with X
- Sub-itemset pruning: if Y \( \supset X \), and \( \text{sup}(X) = \text{sup}(Y) \), X and all of X's descendants in the set enumeration tree can be pruned
- Hybrid tree projection
  - Bottom-up physical tree-projection
  - Top-down pseudo tree-projection
- Item skipping: if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
- Efficient subset checking
CHARM: Mining by Exploring Vertical Data Format

- Vertical format: \( t(AB) = \{T_{11}, T_{25}, \ldots\} \)
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - \( t(X) = t(Y): X \) and \( Y \) always happen together
  - \( t(X) \subset t(Y): \) transaction having \( X \) always has \( Y \)
- Using \textit{diffset} to accelerate mining
  - Only keep track of differences of tids
  - \( t(X) = \{T_1, T_2, T_3\}, \ t(XY) = \{T_1, T_3\} \)
  - Diffset \( (XY, X) = \{T_2\} \)
- Eclat/MaxEclat (Zaki et al. @KDD’97), VIPER(P. Shenoy et al.@SIGMOD’00), CHARM (Zaki & Hsiao@SDM’02)
Further Improvements of Mining Methods

- AFOPT (Liu, et al. @ KDD’03)
  - A —push-right‖ method for mining condensed frequent pattern (CFP) tree
- Carpenter (Pan, et al. @ KDD’03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining
Visualization of Association Rules: Plane Graph
Visualization of Association Rules: Rule Graph
Visualization of Association Rules (SGI/MineSet 3.0)
UNIT – III Association Rules

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Summary
Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications
Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, ...
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - Invisible\textsuperscript{||} data mining
UNIT - IV Classification

- What is classification?
  What is prediction?

- Issues regarding classification and prediction

- Classification by decision tree induction

- Bayesian classification

- Lazy learners (or learning from your neighbors)

Summary
Classification vs. Prediction

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - 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 is 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
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known
Process (1): Model Construction

Training Data

Classification Algorithms

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

IF rank = ‘professor’ OR years > 6
THEN tenured = ‘yes’
Process (2): Using the Model in Prediction

### Unseen Data

(Jeff, Professor, 4)

Tenured?

Yes

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Merlisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

July 16, 2018

Data Mining: Concepts and Techniques
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
UNIT - IV Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Lazy learners (or learning from your neighbors)
- Summary
Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data
Issues: Evaluating Classification Methods

- **Accuracy**
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- **Other measures**, e.g., goodness of rules, such as decision tree size or compactness of classification rules
UNIT – 4 Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Lazy learners (or learning from your neighbors)
- Summary
### Decision Tree Induction: Training Dataset

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

This follows an example of Quinlan’s ID3 (Playing Tennis)
Output: A Decision Tree for \textit{buys_computer}
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
Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let $p_i$ be the probability that an arbitrary tuple in $D$ belongs to class $C_i$, estimated by $|C_{i,D}|/|D|$.
- **Expected information** (entropy) needed to classify a tuple in $D$:
  \[
  Info(D) = - \sum_{i=1}^{m} p_i \log_2(p_i)
  \]
- Information needed (after using $A$ to split $D$ into $v$ partitions) to classify $D$:
  \[
  Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)
  \]
- **Information gained** by branching on attribute $A$
  \[
  Gain(A) = Info(D) - Info_A(D)
  \]
Attribute Selection: Information Gain

- Class P: buys_computer = yes
- Class N: buys_computer = no

\[ \text{Info}(D) = I(9, 5) = - \frac{9}{14} \log_2 \left( \frac{9}{14} \right) - \frac{5}{14} \log_2 \left( \frac{5}{14} \right) = 0.940 \]

<table>
<thead>
<tr>
<th>age</th>
<th>( p_i )</th>
<th>( n_i )</th>
<th>( I(p_i, n_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \leq 30 )</td>
<td>2</td>
<td>3</td>
<td>0.971</td>
</tr>
<tr>
<td>31...40</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( &gt; 40 )</td>
<td>3</td>
<td>2</td>
<td>0.971</td>
</tr>
</tbody>
</table>

\[ \frac{5}{14} I(2,3) \] means \( \text{age} \leq 30 \) has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

\[ \text{Gain}(age) = \text{Info}(D) - \text{Info}_{age}(D) = 0.246 \]

Similarly,

\[ \text{Gain}(income) = 0.029 \]

\[ \text{Gain}(student) = 0.151 \]

\[ \text{Gain}(credit \_ rating) = 0.048 \]
Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - \((a_i + a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)
    - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying \(A \leq \text{split-point}\), and
  - D2 is the set of tuples in D satisfying \(A > \text{split-point}\)
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

\[
SplitInfo_A(D) = - \sum_{j=1}^{v} \left| \frac{D_j}{D} \right| \times \log_2 \left( \frac{|D_j|}{|D|} \right)
\]

- GainRatio(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}
- Ex. \hspace{1cm} \text{SplitInfo}_A(D) = \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 0.926

- gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute
Gini index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where $p_j$ is the relative frequency of class $j$ in $D$

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)
Gini index (CART, IBM IntelligentMiner)

- Ex. D has 9 tuples in buys_computer = yes and 5 in no

\[
gini(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459
\]

- Suppose the attribute income partitions D into 10 in \(D_1\): \{low, medium\} and 4 in \(D_2\)

\[
gini_{\text{income} \in \{\text{low}, \text{medium}\}}(D) = \frac{10}{14}gini(D) + \frac{4}{14}gini(D)
\]

\[
= \frac{10}{14}(1 - \left( \frac{6}{10} \right)^2 - \left( \frac{4}{10} \right)^2) + \frac{4}{14}(1 - \left( \frac{1}{4} \right)^2 - \left( \frac{3}{4} \right)^2)
\]

\[
= 0.450
\]

but \(gini_{\{\text{medium,high}\}}\) is 0.30 and thus the best since it is the lowest

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes
Comparing Attribute Selection Measures

The three measures, in general, return good results but

- Information gain:
  - biased towards multivalued attributes

- Gain ratio:
  - tends to prefer unbalanced splits in which one partition is much smaller than the others

- Gini index:
  - biased to multivalued attributes
  - has difficulty when # of classes is large
  - tends to favor tests that result in equal-sized partitions and purity in both partitions
Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to $\chi^2$ distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others
Overfitting and Tree Pruning

- **Overfitting:** An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - 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‖
Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication
Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
Scalable Decision Tree Induction Methods

- **SLIQ** (EDBT’96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory

- **SPRINT** (VLDB’96 — J. Shafer et al.)
  - Constructs an attribute list data structure

- **PUBLIC** (VLDB’98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier

- **RainForest** (VLDB’98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

- **BOAT** (PODS’99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples
Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: **AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute $X$)
  - Projection of training dataset onto the attribute $X$ and class label where counts of individual class label are aggregated
- **AVC-group** (of a node $n$)
  - Set of AVC-sets of all predictor attributes at the node $n$
**Rainforest: Training Set and Its AVC Sets**

### Training Examples

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buy_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

### AVC-set on Age

| Age    | Buy_Computer | AVC-set on income
|--------|--------------|------------------|
| <=30   | no           | high
| <=30   | no           | high
| 31..40 | no           | medium
| >40    | yes          | low
| >40    | yes          | low

### AVC-set on Student

<table>
<thead>
<tr>
<th>student</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>yes</td>
<td>1</td>
</tr>
<tr>
<td>no</td>
<td>3</td>
</tr>
<tr>
<td>no</td>
<td>4</td>
</tr>
</tbody>
</table>

### AVC-set on credit_rating

<table>
<thead>
<tr>
<th>credit_rating</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>fair</td>
<td>6</td>
</tr>
<tr>
<td>excellent</td>
<td>3</td>
</tr>
</tbody>
</table>

---

*July 10, 2018*
Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
- Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level
BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory.
- Each subset is used to create a tree, resulting in several trees.
- These trees are examined and used to construct a new tree $T'$.
  - It turns out that $T'$ is very close to the tree that would be generated using the whole data set together.
- Adv: requires only two scans of DB, an incremental alg.
Presentation of Classification Results

Classification attribute: product
- Environmental Line
- GO Sport Line
- Outdoor Products
Visualization of a Decision Tree in SGI/MineSet 3.0
Interactive Visual Mining by Perception-Based Classification (PBC)
UNIT - IV Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Lazy learners (or learning from your neighbors)
- Summary
Bayesian Classification: Why?

- **A statistical classifier**: performs *probabilistic prediction*, i.e., predicts class membership probabilities.
- **Foundation**: Based on Bayes’ Theorem.
- **Performance**: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers.
- **Incremental**: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data.
- **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: Basics

- Let \( X \) be a data sample (―evidence‖): class label is unknown
- Let \( H \) be a *hypothesis* that \( X \) belongs to class \( C \)
- Classification is to determine \( P(H|X) \), the probability that the hypothesis holds given the observed data sample \( X \)
- \( P(H) \) *(prior probability)*, the initial probability
  - E.g., \( X \) will buy computer, regardless of age, income, ...
- \( P(X) \): probability that sample data is observed
- \( P(X|H) \) *(posteriori probability)*, the probability of observing the sample \( X \), given that the hypothesis holds
  - E.g., Given that \( X \) will buy computer, the prob. that \( X \) is 31..40, medium income
Bayesian Theorem

- Given training data $X$, posteriori probability of a hypothesis $H$, $P(H|X)$, follows the Bayes theorem

\[
P(H | X) = \frac{P(X | H) P(H)}{P(X)}
\]

- Informally, this can be written as
  
  posteriori = likelihood x prior/evidence

- Predicts $X$ belongs to $C_2$ iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the $k$ classes

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost
Towards Naïve Bayesian Classifier

- Let \( D \) be a training set of tuples and their associated class labels, and each tuple is represented by an \( n \)-D attribute vector \( \mathbf{X} = (x_1, x_2, \ldots, x_n) \)
- Suppose there are \( m \) classes \( C_1, C_2, \ldots, C_m \).
- Classification is to derive the maximum posteriori, i.e., the maximal \( P(C_i|X) \)
- This can be derived from Bayes’ theorem

\[
P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}
\]

- Since \( P(X) \) is constant for all classes, only

\[
P(C_i|X) = P(X|C_i)P(C_i)
\]

needs to be maximized
Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):
  \[
P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times ... \times P(x_n | C_i)
\]

- This greatly reduces the computation cost: Only counts the class distribution

- If \(A_k\) is categorical, \(P(x_k | C_i)\) is the # of tuples in \(C_i\) having value \(x_k\) for \(A_k\) divided by \(|C_{i,D}|\) ( # of tuples of \(C_i\) in \(D\))

- If \(A_k\) is continous-valued, \(P(x_k | C_i)\) is usually computed based on Gaussian distribution with a mean \(\mu\) and standard deviation \(\sigma\)

\[
g(x, \mu, \sigma) = \frac{1}{\sqrt{2 \pi} \sigma} e^{-\frac{(x-\mu)^2}{2 \sigma^2}}
\]

and \(P(x_k | C_i)\) is

\[
P(X | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})
\]
Naïve Bayesian Classifier: Training Dataset

Class:
C1: buys_computer = ‘yes’
C2: buys_computer = ‘no’

Data sample
X = (age <=30, Income = medium, Student = yes
Credit_rating = Fair)

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit</th>
<th>rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Naive Bayesian Classifier: An Example

- **P(C_i):**
  
  \[
P(\text{buys_computer} = \text{yes}) = \frac{9}{14} = 0.643
  \]
  
  \[
P(\text{buys_computer} = \text{no}) = \frac{5}{14} = 0.357
  \]

- **Compute P(X|C_i) for each class**

  \[
P(\text{age} = \leq 30 | \text{buys_computer} = \text{yes}) = \frac{2}{9} = 0.222
  \]
  
  \[
P(\text{age} = \leq 30 | \text{buys_computer} = \text{no}) = \frac{3}{5} = 0.6
  \]
  
  \[
P(\text{income} = \text{medium} | \text{buys_computer} = \text{yes}) = \frac{4}{9} = 0.444
  \]
  
  \[
P(\text{income} = \text{medium} | \text{buys_computer} = \text{no}) = \frac{2}{5} = 0.4
  \]
  
  \[
P(\text{student} = \text{yes} | \text{buys_computer} = \text{yes}) = \frac{6}{9} = 0.667
  \]
  
  \[
P(\text{student} = \text{yes} | \text{buys_computer} = \text{no}) = \frac{1}{5} = 0.2
  \]
  
  \[
P(\text{credit_rating} = \text{fair} | \text{buys_computer} = \text{yes}) = \frac{6}{9} = 0.667
  \]
  
  \[
P(\text{credit_rating} = \text{fair} | \text{buys_computer} = \text{no}) = \frac{2}{5} = 0.4
  \]

- **X = (age <= 30, income = medium, student = yes, credit_rating = fair)**

  \[
P(X | C_i) : P(X | \text{buys_computer} = \text{yes}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044
  \]
  
  \[
P(X | \text{buys_computer} = \text{no}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019
  \]

  \[
P(X | C_i) \times P(C_i) : P(X | \text{buys_computer} = \text{yes}) \times P(\text{buys_computer} = \text{yes}) = 0.028
  \]
  
  \[
P(X | \text{buys_computer} = \text{no}) \times P(\text{buys_computer} = \text{no}) = 0.007
  \]

- **Therefore, X belongs to class (“buys_computer = yes”)**
Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

\[
P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i)
\]

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    - \( \text{Prob(income = low)} = 1/1003 \)
    - \( \text{Prob(income = medium)} = 991/1003 \)
    - \( \text{Prob(income = high)} = 11/1003 \)
  - The —corrected‖ prob. estimates are close to their —uncorrected‖ counterparts
Naïve Bayesian Classifier: Comments

- **Advantages**
  - Easy to implement
  - Good results obtained in most of the cases

- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

- **How to deal with these dependencies?**
  - Bayesian Belief Networks
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

Family History  Smoker  LungCancer  Emphysema

PositiveXRay  Dyspnea

The conditional probability table (CPT) for variable LungCancer:

<table>
<thead>
<tr>
<th></th>
<th>(FH, S)</th>
<th>(FH, ~S)</th>
<th>(~FH, S)</th>
<th>(~FH, ~S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>~LC</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>

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:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(y_i))$$
Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: gradient descent (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
  - Ref. D. Heckerman: Bayesian networks for data mining
UNIT - IV Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Lazy learners (or learning from your neighbors)
- Summary
UNIT - IV Classification

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Lazy learners (or learning from your neighbors)
- Summary
Lazy vs. Eager Learning

- **Lazy vs. eager learning**
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify

- **Lazy**: less time in training but more time in predicting

- **Accuracy**
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing (—lazy evaluation‖) until a new instance must be classified

- Typical approaches
  - $k$-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference
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, \( \text{dist}(\mathbf{X}_1, \mathbf{X}_2) \)
- 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 \( x_q \)
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples
Discussion on the $k$-NN Algorithm

- $k$-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

$$w = \frac{1}{d(x_q, x_i)^2}$$
Case-Based Reasoning (CBR)

- **CBR**: Uses a database of problem solutions to solve new problems
- Store *symbolic description* (tuples or cases)—not points in a Euclidean space
- **Applications**: Customer-service (product-related diagnosis), legal ruling
- **Methodology**
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- **Challenges**
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases
Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.

Effective and scalable methods have been developed for decision trees induction, Naive Bayesian classification, Bayesian belief network, rule-based classifier, Backpropagation, Support Vector Machine (SVM), associative classification, nearest neighbor classifiers, and case-based reasoning, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches.
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms
Clustering: Rich Applications and Multidisciplinary Efforts

- Pattern Recognition
- Spatial Data Analysis
  - Create thematic maps in GIS by clustering feature spaces
  - Detect spatial clusters or for other spatial mining tasks
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns
Examples of Clustering Applications

- **Marketing**: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.
- **Land use**: Identification of areas of similar land use in an earth observation database.
- **Insurance**: Identifying groups of motor insurance policy holders with a high average claim cost.
- **City-planning**: Identifying groups of houses according to their house type, value, and geographical location.
- **Earthquake studies**: Observed earth quake epicenters should be clustered along continent faults.
Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters with:
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.
Measure the Quality of Clustering

- **Dissimilarity/Similarity metric**: Similarity is expressed in terms of a distance function, typically metric: \(d(i, j)\)
- There is a separate —quality‖ function that measures the —goodness‖ of a cluster.
- The definitions of **distance functions** are usually very different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables.
- Weights should be associated with different variables based on applications and data semantics.
- It is hard to define —similar enough‖ or —good enough‖
  - the answer is typically highly subjective.
Requirements of Clustering in Data Mining

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

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
Data Structures

- **Data matrix**
  - (two modes)

- **Dissimilarity matrix**
  - (one mode)
Type of data in clustering analysis

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

- Standardize data
  - Calculate the mean absolute deviation:
    $$s_f = \frac{1}{n} (|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|)$$
  
  where 
  $$m_f = \frac{1}{n} (x_{1f} + x_{2f} + \ldots + x_{nf}).$$

- Calculate the standardized measurement (z-score)
  $$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Using mean absolute deviation is more robust than using standard deviation
Similarity and Dissimilarity Between Objects

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

- Some popular ones include: *Minkowski distance*:

  \[ d(i, j) = \sqrt[q]{\sum_{i=1}^{p} |x_{i} - x_{j}|^q} = \sqrt[q]{|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \ldots + |x_{i_p} - x_{j_p}|^q} \]

  where \( i = (x_{i_1}, x_{i_2}, \ldots, x_{i_p}) \) and \( j = (x_{j_1}, x_{j_2}, \ldots, x_{j_p}) \) are two \( p \)-dimensional data objects, and \( q \) is a positive integer.

- If \( q = 1 \), \( d \) is Manhattan distance:

  \[ d(i, j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}| \]
Similarity and Dissimilarity Between Objects (Cont.)

- If \( q = 2 \), \( d \) is Euclidean distance:

\[
d(i, j) = \sqrt{\left| x_i - x_j \right|^2 + \left| x_{i2} - x_{j2} \right|^2 + \ldots + \left| x_{ip} - x_{jp} \right|^2}
\]

- Properties
  - \( d(i, j) \geq 0 \)
  - \( d(i, i) = 0 \)
  - \( d(i, j) = d(j, i) \)
  - \( d(i, j) \leq d(i, k) + d(k, j) \)

- Also, one can use weighted distance, parametric Pearson product moment correlation, or other dissimilarity measures
## Binary Variables

- **A contingency table for binary data**

<table>
<thead>
<tr>
<th>Object $i$</th>
<th>Object $j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a$</td>
</tr>
<tr>
<td>0</td>
<td>$c$</td>
</tr>
<tr>
<td>sum</td>
<td>$a + c$</td>
</tr>
<tr>
<td>1</td>
<td>$b$</td>
</tr>
<tr>
<td>0</td>
<td>$d$</td>
</tr>
<tr>
<td>sum</td>
<td>$b + d$</td>
</tr>
<tr>
<td>sum</td>
<td>$a + b$</td>
</tr>
<tr>
<td>sum</td>
<td>$c + d$</td>
</tr>
<tr>
<td>sum</td>
<td>$p$</td>
</tr>
</tbody>
</table>

- **Distance measure for symmetric binary variables:**

$$d(i, j) = \frac{b + c}{a + b + c + d}$$

- **Distance measure for asymmetric binary variables:**

$$d(i, j) = \frac{b + c}{a + b + c}$$

- **Jaccard coefficient** (*similarity measure for asymmetric binary variables*):

$$sim_{Jaccard}(i, j) = \frac{a}{a + b + c}$$
Dissimilarity between Binary Variables

- **Example**

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Fever</th>
<th>Cough</th>
<th>Test-1</th>
<th>Test-2</th>
<th>Test-3</th>
<th>Test-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>M</td>
<td>Y</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Mary</td>
<td>F</td>
<td>Y</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>P</td>
<td>N</td>
</tr>
<tr>
<td>Jim</td>
<td>M</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

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

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

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

\[
d(jim, mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75
\]
Nominal Variables

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

- Method 1: Simple matching
  - \( m \): # of matches, \( p \): total # of variables
    \[
    d(i, j) = \frac{p - m}{p}
    \]

- Method 2: use a large number of binary variables
  - creating a new binary variable for each of the \( M \) nominal states
Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace $x_{if}$ by their rank
  - map the range of each variable onto $[0, 1]$ by replacing $i$-th object in the $f$-th variable by
    \[
    z_{if} = \frac{r_{if} - 1}{M_f - 1}
    \]
  - compute the dissimilarity using methods for interval-scaled variables
Ratio-Scaled Variables

- **Ratio-scaled variable**: a positive measurement on a nonlinear scale, approximately at exponential scale, such as \( Ae^{Bt} \) or \( Ae^{-Bt} \)

- **Methods:**
  - treat them like interval-scaled variables—*not a good choice!* (why?—the scale can be distorted)
  - apply logarithmic transformation
    \[
    y_{if} = \log(x_{if})
    \]
  - treat them as continuous ordinal data treat their rank as interval-scaled
Variables of Mixed Types

- A database may contain all the six types of variables
  - symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio
- One may use a weighted formula to combine their effects
  \[
  d(i, j) = \frac{\sum_{f}^{p} \delta^{(f)} d_{ij}^{(f)}}{\sum_{f}^{p} \delta^{(f)}}
  \]
- \( f \) is binary or nominal:
  \( d_{ij}^{(f)} = 0 \) if \( x_{if} = x_{jf} \), or \( d_{ij}^{(f)} = 1 \) otherwise
- \( f \) is interval-based: use the normalized distance
- \( f \) is ordinal or ratio-scaled
  - compute ranks \( r_{if} \) and
  - and treat \( z_{if} \) as interval-scaled
  \[
  z_{if} = \frac{r_{if} - 1}{M_{f} - 1}
  \]
Vector Objects

- Vector objects: keywords in documents, gene features in micro-arrays, etc.
- Broad applications: information retrieval, biologic taxonomy, etc.
- Cosine measure
  \[ s(\vec{X}, \vec{Y}) = \frac{\vec{X}^t \cdot \vec{Y}}{|\vec{X}| |\vec{Y}|}, \]
  \( \vec{X}^t \) is a transposition of vector \( \vec{X} \), \(|\vec{X}|\) is the Euclidean normal of vector \( \vec{X} \),
- A variant: Tanimoto coefficient
  \[ s(\vec{X}, \vec{Y}) = \frac{\vec{X}^t \cdot \vec{Y}}{\vec{X}^t \cdot \vec{X} + \vec{Y}^t \cdot \vec{Y} - \vec{X}^t \cdot \vec{Y}}, \]
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
Major Clustering Approaches (I)

- **Partitioning approach:**
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS

- **Hierarchical approach:**
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, ROCK, CAMELEON

- **Density-based approach:**
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
Major Clustering Approaches (II)

- **Grid-based approach:**
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

- **Model-based:**
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB

- **Frequent pattern-based:**
  - Based on the analysis of frequent patterns
  - Typical methods: pCluster

- **User-guided or constraint-based:**
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
Typical Alternatives to Calculate the Distance between Clusters

- **Single link**: smallest distance between an element in one cluster and an element in the other, i.e., \( \text{dis}(K_i, K_j) = \min(t_{ip}, t_{jq}) \)
- **Complete link**: largest distance between an element in one cluster and an element in the other, i.e., \( \text{dis}(K_i, K_j) = \max(t_{ip}, t_{jq}) \)
- **Average**: avg distance between an element in one cluster and an element in the other, i.e., \( \text{dis}(K_i, K_j) = \text{avg}(t_{ip}, t_{jq}) \)
- **Centroid**: distance between the centroids of two clusters, i.e., \( \text{dis}(K_i, K_j) = \text{dis}(C_i, C_j) \)
- **Medoid**: distance between the medoids of two clusters, i.e., \( \text{dis}(K_i, K_j) = \text{dis}(M_i, M_j) \)
  - Medoid: one chosen, centrally located object in the cluster
Centroid, Radius and Diameter of a Cluster
(for numerical data sets)

- **Centroid:** the “middle‖ of a cluster

\[
C_m = \frac{\sum_{i=1}^{N} t_{ip}}{N}
\]

- **Radius:** square root of average distance from any point of the cluster to its centroid

\[
R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{ip} - c^m)^2}{N}}
\]

- **Diameter:** square root of average mean squared distance between all pairs of points in the cluster

\[
D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}
\]
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
Partitioning Algorithms: Basic Concept

- **Partitioning method**: Construct a partition of a database $D$ of $n$ objects into a set of $k$ clusters, s.t., min sum of squared distance

$$\sum_{m=1}^{k} \sum_{t_{mi} \in K_m} (C_m - t_{mi})^2$$

- Given a $k$, find a partition of $k$ clusters that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: $k$-means and $k$-medoids algorithms
  - $k$-means (MacQueen’67): Each cluster is represented by the center of the cluster
  - $k$-medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw’87): Each cluster is represented by one of the objects in the cluster
The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when no more new assignment
The *K*-Means Clustering Method

- Example

```
K=2
Arbitrarily choose K object as initial cluster center
```

Assign each objects to most similar center

```
Assign each objects to most similar center
```

Update the cluster means

```
Update the cluster means
```

reassign

```
reassign
```

reassign

```
reassign
```

```
July 16, 2018
Data Mining: Concepts and Techniques
270
```
Comments on the **K-Means** Method

- **Strength**: Relatively efficient: $O(tkn)$, where $n$ is # objects, $k$ is # clusters, and $t$ is # iterations. Normally, $k, t << n$.
  - Comparing: PAM: $O(k(n-k)^2 )$, CLARA: $O(ks^2 + k(n-k))$

- **Comment**: Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: deterministic annealing and genetic algorithms

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

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - Replacing means of clusters with *modes*
  - Using new dissimilarity measures to deal with categorical objects
  - Using a *frequency*-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method
What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers!
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster.
The *K-Medoids* Clustering Method

- Find *representative* objects, called *medoids*, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)
A Typical K-Medoids Algorithm (PAM)

1. **Arbitrary choose** k object as initial medoids

2. **Assign each remaining** object to nearest medoids

3. **Randomly select** a nonmedoid object, O_{random}

4. **Swapping O and O_{random}** if quality is improved.

**Do loop**

**Until no change**

Total Cost = 20

Data Mining: Concepts and Techniques

July 16, 2018
PAM (Partitioning Around Medoids) (1987)

- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  - Select \( k \) representative objects arbitrarily
  - For each pair of non-selected object \( h \) and selected object \( i \), calculate the total swapping cost \( TC_{ih} \)
  - For each pair of \( i \) and \( h \),
    - If \( TC_{ih} < 0 \), \( i \) is replaced by \( h \)
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change
PAM Clustering: Total swapping cost \( TC_{ih} = \sum_j C_{jih} \)

\[ C_{jih} = d(j, h) - d(j, i) \]

\[ C_{jih} = 0 \]

\[ C_{jih} = d(j, t) - d(j, i) \]
What Is the Problem with PAM?

- Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean.

- Pam works efficiently for small data sets but does not scale well for large data sets.
  - $O(k(n-k)^2)$ for each iteration
    - where $n$ is # of data, $k$ is # of clusters

  ➔ Sampling based method, CLARA(Clustering LARge Applications)
CLARA (Clustering Large Applications) (1990)

- **CLARA** (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as S+
  - It draws *multiple samples* of the data set, applies PAM on each sample, and gives the best clustering as the output
- **Strength**: deals with larger data sets than PAM
- **Weakness**:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased
CLARANS (—Randomized|| CLARA) (1994)

- CLARANS (A Clustering Algorithm based on Randomized Search) (Ng and Han’94)
- CLARANS draws sample of neighbors dynamically
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of \( k \) medoids
- If the local optimum is found, CLARANS starts with new randomly selected node in search for a new local optimum
- It is more efficient and scalable than both PAM and CLARA
- Focusing techniques and spatial access structures may further improve its performance (Ester et al.’95)
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
Hierarchical Clustering

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

$$
\begin{align*}
\text{Step 0} & \quad \text{Step 1} & \quad \text{Step 2} & \quad \text{Step 3} & \quad \text{Step 4} \\

\text{Step 4} & \quad \text{Step 3} & \quad \text{Step 2} & \quad \text{Step 1} & \quad \text{Step 0}
\end{align*}
$$

agglomerative
(AGNES)

divisive
(DIANA)
AGNES (Agglomerative Nesting)

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

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

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

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

- Major weakness of agglomerative clustering methods
  - do not scale well: time complexity of at least $O(n^2)$, where $n$ is the number of total objects
  - can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - ROCK (1999): clustering categorical data by neighbor and link analysis
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling
BIRCH (1996)

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies (Zhang, Ramakrishnan & Livny, SIGMOD’96)
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- Scales linearly: finds a good clustering with a single scan and improves the quality with a few additional scans
- Weakness: handles only numeric data, and sensitive to the order of the data record.
Clustering Feature Vector in BIRCH

Clustering Feature: $CF = (N, LS, SS)$

$N$: Number of data points

$LS$: $\sum_{i=1}^{N} X_i$

$SS$: $\sum_{i=1}^{N} X_i^2$

$CF = (5, (16, 30), (54, 190))$

(3, 4)
(2, 6)
(4, 5)
(4, 7)
(3, 8)
CF-Tree in BIRCH

- **Clustering feature:**
  - summary of the statistics for a given subcluster: the 0-th, 1st and 2nd moments of the subcluster from the statistical point of view.
  - registers crucial measurements for computing cluster and utilizes storage efficiently

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A nonleaf node in a tree has descendants or —children‖
  - The nonleaf nodes store sums of the CFs of their children

- A CF tree has two parameters
  - Branching factor: specify the maximum number of children.
  - threshold: max diameter of sub-clusters stored at the leaf nodes
The CF Tree Structure

<table>
<thead>
<tr>
<th></th>
<th>CF_1</th>
<th>CF_2</th>
<th>CF_3</th>
<th></th>
<th>CF_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>child_1</td>
<td>child_2</td>
<td>child_3</td>
<td></td>
<td></td>
<td>child_6</td>
</tr>
</tbody>
</table>

Non-leaf node

<table>
<thead>
<tr>
<th></th>
<th>CF_1</th>
<th>CF_2</th>
<th>CF_3</th>
<th></th>
<th>CF_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>child_1</td>
<td>child_2</td>
<td>child_3</td>
<td></td>
<td></td>
<td>child_5</td>
</tr>
</tbody>
</table>

Leaf node

<table>
<thead>
<tr>
<th>prev</th>
<th>CF_1</th>
<th>CF_2</th>
<th></th>
<th>CF_6</th>
<th>next</th>
</tr>
</thead>
</table>

Leaf node

prev | CF_1 | CF_2 |     | CF_4 | next

B = 7
L = 6

The CF Tree Structure
Clustering Categorical Data: The ROCK Algorithm

- ROCK: RObust Clustering using linKs
  - S. Guha, R. Rastogi & K. Shim, ICDE‘99
- Major ideas
  - Use links to measure similarity/proximity
  - Not distance-based
  - Computational complexity: \( O(n^2 + nm_m m_a + n^2 \log n) \)
- Algorithm: sampling-based clustering
  - Draw random sample
  - Cluster with links
  - Label data in disk
- Experiments
  - Congressional voting, mushroom data
Similarity Measure in ROCK

- Traditional measures for categorical data may not work well, e.g., Jaccard coefficient
- Example: Two groups (clusters) of transactions
  - \( C_1. <a, b, c, d, e>\): \{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}
  - \( C_2. <a, b, f, g>\): \{a, b, f\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}
- Jaccard co-efficient may lead to wrong clustering result
  - \( C_1\): 0.2 (\{a, b, c\}, \{b, d, e\}) to 0.5 (\{a, b, c\}, \{a, b, d\})
  - \( C_1 \& C_2\): could be as high as 0.5 (\{a, b, c\}, \{a, b, f\})
- Jaccard co-efficient-based similarity function:
  \[
  S \text{im} \left( T_1, T_2 \right) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}
  \]
- Ex. Let \( T_1 = \{a, b, c\}, T_2 = \{c, d, e\} \)
  \[
  S \text{im} \left( T_1, T_2 \right) = \frac{|\{ c \}|}{|\{ a, b, c, d, e \}|} = \frac{1}{5} = 0.2
  \]
Link Measure in ROCK

- Links: # of common neighbors
  - $C_1 <a, b, c, d, e>: \{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}$
  - $C_2 <a, b, f, g>: \{a, b, f\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}$
  - Let $T_1 = \{a, b, c\}, T_2 = \{c, d, e\}, T_3 = \{a, b, f\}$
    - link($T_1$, $T_2$) = 4, since they have 4 common neighbors
      - $\{a, c, d\}, \{a, c, e\}, \{b, c, d\}, \{b, c, e\}$
    - link($T_1$, $T_3$) = 3, since they have 3 common neighbors
      - $\{a, b, d\}, \{a, b, e\}, \{a, b, g\}$
  - Thus link is a better measure than Jaccard coefficient

- CHAMELEON: by G. Karypis, E.H. Han, and V. Kumar*99
- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the interconnectivity and closeness (proximity) between two clusters are high relative to the internal interconnectivity of the clusters and closeness of items within the clusters
  - Cure ignores information about interconnectivity of the objects, Rock ignores information about the closeness of two clusters
- A two-phase algorithm
  1. Use a graph partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
  2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters
Overall Framework of CHAMELEON

Construct Sparse Graph

Partition the Graph

Merge Partition

Data Set

Final Clusters

July 16, 2018

Data Mining: Concepts and Techniques
CHAMELEON (Clustering Complex Objects)
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
What Is Outlier Discovery?

- What are outliers?
  - The set of objects are considerably dissimilar from the remainder of the data
  - Example: Sports: Michael Jordon, Wayne Gretzky, ...
- Problem: Define and find outliers in large data sets
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Customer segmentation
  - Medical analysis
Assume a model underlying distribution that generates data set (e.g. normal distribution)

- Use discordancy tests depending on
  - data distribution
  - distribution parameter (e.g., mean, variance)
  - number of expected outliers

Drawbacks
- most tests are for single attribute
- In many cases, data distribution may not be known
Outlier Discovery: Distance-Based Approach

- Introduced to counter the main limitations imposed by statistical methods
  - We need multi-dimensional analysis without knowing data distribution
- Distance-based outlier: A $DB(p, D)$-outlier is an object $O$ in a dataset $T$ such that at least a fraction $p$ of the objects in $T$ lies at a distance greater than $D$ from $O$
- Algorithms for mining distance-based outliers
  - Index-based algorithm
  - Nested-loop algorithm
  - Cell-based algorithm
Density-Based Local Outlier Detection

- Distance-based outlier detection is based on global distance distribution
- It encounters difficulties to identify outliers if data is not uniformly distributed
- Ex. $C_1$ contains 400 loosely distributed points, $C_2$ has 100 tightly condensed points, 2 outlier points $o_1$, $o_2$
- Distance-based method cannot identify $o_2$ as an outlier
- Need the concept of local outlier

- Local outlier factor (LOF)
  - Assume outlier is not crisp
  - Each point has a LOF
Outlier Discovery: Deviation-Based Approach

- Identifies outliers by examining the main characteristics of objects in a group
- Objects that —deviate— from this description are considered outliers
- Sequential exception technique
  - simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects
- OLAP data cube technique
  - uses data cubes to identify regions of anomalies in large multidimensional data
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
UNIT- V Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Outlier Analysis
7. Summary
Cluster analysis groups objects based on their similarity and has wide applications.

Measure of similarity can be computed for various types of data.

Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods.

Outlier detection and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches.

There are still lots of research issues on cluster analysis.
Problems and Challenges

- Considerable progress has been made in scalable clustering methods
  - Partitioning: k-means, k-medoids, CLARANS
  - Hierarchical: BIRCH, ROCK, CHAMELEON
- Current clustering techniques do not address all the requirements adequately, still an active area of research
Mining Complex Types of Data

Mining time-series and sequence data
Mining the World-Wide Web
Mining spatial databases
Mining multimedia databases
Summary
Mining Time-Series and Sequence Data

Time-series database
Consists of sequences of values or events changing with time
Data is recorded at regular intervals
Characteristic time-series components
  Trend, cycle, seasonal, irregular

Applications
  Financial: stock price, inflation
  Biomedical: blood pressure
  Meteorological: precipitation
Mining Time-Series and Sequence Data

Time-series plot

Price History - International Business Machines Corp... (3/17/97 - 3/20/98)

- Chart
- Period
 Compare with:
   - Add
   - IBM
   - MSFT
   - INTC

Export Data
Options
Print
Mining Time-Series and Sequence Data: Trend analysis

A time series can be illustrated as a time-series graph which describes a point moving with the passage of time

Categories of Time-Series Movements

- Long-term or trend movements (trend curve)
- Cyclic movements or cycle variations, e.g., business cycles
- Seasonal movements or seasonal variations
  - i.e., almost identical patterns that a time series appears to follow during corresponding months of successive years.
- Irregular or random movements
Estimation of Trend Curve

The freehand method
   Fit the curve by looking at the graph
   Costly and barely reliable for large-scaled data mining

The least-square method
   Find the curve minimizing the sum of the squares of the deviation of points on the curve from the corresponding data points

The moving-average method
   Eliminate cyclic, seasonal and irregular patterns
   Sensitive to outliers
Discovery of Trend in Time-Series

Estimation of irregular variations

By adjusting the data for trend, seasonal and cyclic variations. With the systematic analysis of the trend, cyclic, seasonal, and irregular components, it is possible to make long- or short-term predictions with reasonable quality.
Similarity Search in Time-Series Analysis

Normal database query finds exact match
Similarity search finds data sequences that differ only slightly from the given query sequence
Two categories of similarity queries
  Whole matching: find a sequence that is similar to the query sequence
  Subsequence matching: find all pairs of similar sequences

Typical Applications
  Financial market
  Market basket data analysis
  Scientific databases
  Medical diagnosis
Many techniques for signal analysis require the data to be in the frequency domain. Usually data-independent transformations are used. The transformation matrix is determined a priori. E.g., discrete Fourier transform (DFT), discrete wavelet transform (DWT). The distance between two signals in the time domain is the same as their Euclidean distance in the frequency domain. DFT does a good job of concentrating energy in the first few coefficients. If we keep only first a few coefficients in DFT, we can compute the lower bounds of the actual distance.
Multidimensional Indexing

Multidimensional index
   Constructed for efficient accessing using the first few Fourier coefficients
Use the index to retrieve the sequences that are at most a certain small distance away from the query sequence
Perform post-processing by computing the actual distance between sequences in the time domain and discard any false matches
Subsequence Matching

Break each sequence into a set of pieces of window with length $w$
Extract the features of the subsequence inside the window
Map each sequence to a “trail” in the feature space
Divide the trail of each sequence into “subtrails” and represent each of them with minimum bounding rectangle
Use a multipiece assembly algorithm to search for longer sequence matches
Enhanced similarity search methods

Allow for gaps within a sequence or differences in offsets or amplitudes
Normalize sequences with amplitude scaling and offset translation
Two subsequences are considered similar if one lies within an envelope of $\varepsilon$ width around the other, ignoring outliers
Two sequences are said to be similar if they have enough non-overlapping time-ordered pairs of similar subsequences
Parameters specified by a user or expert: sliding window size, width of an envelope for similarity, maximum gap, and matching fraction
Similar time series analysis
Steps for Performing a Similarity Search

Atomic matching
   Find all pairs of gap-free windows of a small length that are similar

Window stitching
   Stitch similar windows to form pairs of large similar subsequences allowing gaps between atomic matches

Subsequence Ordering
   Linearly order the subsequence matches to determine whether enough similar pieces exist
Similar time series analysis

VanEck International Fund

Fidelity Selective Precious Metal and Mineral Fund

Two similar mutual funds in the different fund group
Query Languages for Time Sequences

Time-sequence query language
Should be able to specify sophisticated queries like
   Find all of the sequences that are similar to some sequence in class $A$, but not similar to any sequence in class $B$
Should be able to support various kinds of queries: range queries, all-pair queries, and nearest neighbor queries

Shape definition language
Allows users to define and query the overall shape of time sequences
Uses human readable series of sequence transitions or macros
Ignores the specific details
   E.g., the pattern up, Up, UP can be used to describe increasing degrees of rising slopes
   Macros: spike, valley, etc.
Sequential Pattern Mining

Mining of frequently occurring patterns related to time or other sequences
Sequential pattern mining usually concentrate on symbolic patterns
Examples
   Renting “Terminator I”, then “Terminator II”, then “Terminator III” in that order
   Collection of ordered events within an interval
Applications
   Targeted marketing
   Customer retention
   Weather prediction
## Mining Sequences (cont.)

<table>
<thead>
<tr>
<th>CustId</th>
<th>Video sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{(C), (H)}</td>
</tr>
<tr>
<td>2</td>
<td>{(AB), (C), (DFG)}</td>
</tr>
<tr>
<td>3</td>
<td>{(CEG)}</td>
</tr>
<tr>
<td>4</td>
<td>{(C), (DG), (H)}</td>
</tr>
<tr>
<td>5</td>
<td>{(H)}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Large Itemsets</th>
<th>MappedID</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C)</td>
<td>1</td>
</tr>
<tr>
<td>(D)</td>
<td>2</td>
</tr>
<tr>
<td>(G)</td>
<td>3</td>
</tr>
<tr>
<td>(DG)</td>
<td>4</td>
</tr>
<tr>
<td>(H)</td>
<td>5</td>
</tr>
</tbody>
</table>

Sequential patterns with support > 0.25:
- {(C), (H)}
- {(C), (DG)}
Sequential pattern mining: Cases and Parameters

Duration of a time sequence $T$
Sequential pattern mining can then be confined to the data within a specified duration
Ex. Subsequence corresponding to the year of 1999
Ex. Partitioned sequences, such as every year, or every week after stock crashes, or every two weeks before and after a volcano eruption

Event folding window $w$
If $w = T$, time-insensitive frequent patterns are found
If $w = 0$ (no event sequence folding), sequential patterns are found where each event occurs at a distinct time instant
If $0 < w < T$, sequences occurring within the same period $w$ are folded in the analysis
Sequential pattern mining: Cases and Parameters

Time interval, \( int \), between events in the discovered pattern

\( int = 0 \): no interval gap is allowed, i.e., only strictly consecutive sequences are found

Ex. “Find frequent patterns occurring in consecutive weeks”

\( min_int \leq int \leq max_int \): find patterns that are separated by at least \( min_int \) but at most \( max_int \)

Ex. “If a person rents movie A, it is likely she will rent movie B within 30 days” \((int \leq 30)\)

\( int = c \neq 0 \): find patterns carrying an exact interval

Ex. “Every time when Dow Jones drops more than 5%, what will happen exactly two days later?” \((int = 2)\)
Episodes and Sequential Pattern Mining

Methods

Other methods for specifying the kinds of patterns

Serial episodes: \( A \rightarrow B \)
Parallel episodes: \( A \& B \)
Regular expressions: \( (A \mid B)C^*(D \rightarrow E) \)

Methods for sequential pattern mining

Variations of Apriori-like algorithms, e.g., GSP
Database projection-based pattern growth

Similar to the frequent pattern growth without candidate generation
Periodicity Analysis

Periodicity is everywhere: tides, seasons, daily power consumption, etc.

**Full periodicity**
Every point in time contributes (precisely or approximately) to the periodicity

**Partial periodicity**: A more general notion
Only some segments contribute to the periodicity
  
  Jim reads NY Times 7:00-7:30 am every week day

**Cyclic association rules**
Associations which form cycles

**Methods**
Full periodicity: FFT, other statistical analysis methods
Partial and cyclic periodicity: Variations of Apriori-like mining methods
Mining Complex Types of Data

Mining time-series and sequence data

Mining the World-Wide Web

Mining spatial databases

Mining multimedia databases

Summary
Mining the World-Wide Web

The WWW is huge, widely distributed, global information service center for
Information services: news, advertisements, consumer information, financial management, education, government, e-commerce, etc.
Hyper-link information
Access and usage information
WWW provides rich sources for data mining
Challenges
Too huge for effective data warehousing and data mining
Too complex and heterogeneous: no standards and structure
Mining the World-Wide Web

Growing and changing very rapidly

![Internet Timeline](image)

Broad diversity of user communities

Only a small portion of the information on the Web is truly relevant or useful

99% of the Web information is useless to 99% of Web users

How can we find high-quality Web pages on a specified topic?
Index-based: search the Web, index Web pages, and build and store huge keyword-based indices
Help locate sets of Web pages containing certain keywords
Deficiencies
A topic of any breadth may easily contain hundreds of thousands of documents
Many documents that are highly relevant to a topic may not contain keywords defining them
Web Mining: A more challenging task

Searches for
- Web access patterns
- Web structures
- Regularity and dynamics of Web contents

Problems
- The “abundance” problem
- Limited coverage of the Web: hidden Web sources, majority of data in DBMS
- Limited query interface based on keyword-oriented search
- Limited customization to individual users
Web Mining Taxonomy

Web Mining

Web Content Mining
- Web Page Content Mining

Web Structure Mining
- Search Result Mining

Web Usage Mining
- General Access Pattern Tracking
- Customized Usage Tracking
Web Mining

Web Structure Mining

Web Usage Mining

Search Result Mining

General Access Pattern Tracking

Customized Usage Tracking

Web Page Content Mining

Web Page Summarization

WebLog (Lakshmanan et.al. 1996), WebOQL (Mendelzon et.al. 1998) …:
Web Structuring query languages; Can identify information within given web pages

• Ahoy! (Etzioni et.al. 1997): Uses heuristics to distinguish personal home pages from other web pages

• ShopBot (Etzioni et.al. 1997): Looks for product prices within web pages
Web Mining

Web Content Mining

Web Structure Mining

Web Usage Mining

Search Result Mining

Search Engine Result Summarization

- Clustering Search Result (Leouski and Croft, 1996, Zamir and Etzioni, 1997):
  Categorizes documents using phrases in titles and snippets
Mining the World-Wide Web

Web Mining

- Web Content Mining
- Search Result Mining
- Web Page Content Mining

Web Usage Mining

- General Access Pattern Tracking
- Customized Usage Tracking

Web Structure Mining

Using Links
- PageRank (Brin et al., 1998)
- CLEVER (Chakrabarti et al., 1998)
Use interconnections between web pages to give weight to pages.

Using Generalization
Uses a multi-level database representation of the Web. Counters (popularity) and link lists are used for capturing structure.
Web Mining

Web Content Mining
- Web Page Content Mining
- Search Result Mining

Web Structure Mining

Web Usage Mining

General Access Pattern Tracking
- **Web Log Mining** (Zaïane, Xin and Han, 1998)
  Uses KDD techniques to understand general access patterns and trends. Can shed light on better structure and grouping of resource providers.

Customized Usage Tracking
Mining the World-Wide Web

Web Mining

- Web Content Mining
  - Web Page Content Mining
  - Search Result Mining

- Web Structure Mining
  - General Access Pattern Tracking

- Web Usage Mining
  - Customized Usage Tracking
    - Adaptive Sites (Perkowitz and Etzioni, 1997)
      Analyzes access patterns of each user at a time. Web site restructures itself automatically by learning from user access patterns.
Mining the Web's Link Structures

Finding authoritative Web pages
Retrieving pages that are not only relevant, but also of high quality, or authoritative on the topic

Hyperlinks can infer the notion of authority
The Web consists not only of pages, but also of hyperlinks pointing from one page to another
These hyperlinks contain an enormous amount of latent human annotation
A hyperlink pointing to another Web page, this can be considered as the author's endorsement of the other page
Problems with the Web linkage structure
   Not every hyperlink represents an endorsement
     Other purposes are for navigation or for paid advertisements
     If the majority of hyperlinks are for endorsement, the collective opinion will still dominate
   One authority will seldom have its Web page point to its rival authorities in the same field

Hub
   Set of Web pages that provides collections of links to authorities
Automatic Classification of Web Documents

Assign a class label to each document from a set of predefined topic categories
Based on a set of examples of preclassified documents
Example
  Use Yahoo!'s taxonomy and its associated documents as training and test sets
  Derive a Web document classification scheme
  Use the scheme classify new Web documents by assigning categories from the same taxonomy
Keyword-based document classification methods
Statistical models
Multilayered Web Information Base

Layer$_0$: the Web itself
Layer$_1$: the Web page descriptor layer
  Contains descriptive information for pages on the Web
  An abstraction of Layer$_0$: substantially smaller but still rich enough to preserve most of the interesting, general information
  Organized into dozens of semistructured classes
  \( \textit{document, person, organization, ads, directory, sales, software, game, stocks, library\_catalog, geographic\_data, scientific\_data, etc.} \)

Layer$_2$ and up: various Web directory services constructed on top of Layer$_1$
  provide multidimensional, application-specific services
Multiple Layered Web Architecture

Layer$_n$
...
Layer$_1$
Layer$_0$
Mining the World-Wide Web

Layer-0: Primitive data

Layer-1: dozen database relations representing types of objects (metadata)

document, organization, person, software, game, map, image, ...

• **document**(file_addr, authors, title, publication, publication_date, abstract, language, table_of_contents, category_description, keywords, index, multimedia_attached, num_pages, format, first_paragraphs, size_doc, timestamp, access_frequency, links_out, ..)

• **person**(last_name, first_name, home_page_addr, position, picture_attached, phone, e-mail, office_address, education, research_interests, publications, size_of_home_page, timestamp, access_frequency, ..)

• **image**(image_addr, author, title, publication_date, category_description, keywords, size, width, height, duration, format, parent_pages, colour_histogram, Colour_layout, Texture_layout, Movement_vector, localisation_vector, timestamp, access_frequency, ..)
Layer-2: simplification of layer-1

- **doc_brief** (file_addr, authors, title, publication, publication_date, abstract, language, category_description, key_words, major_index, num_pages, format, size_doc, access_frequency, links_out)

- **person_brief** (last_name, first_name, publications, affiliation, e-mail, research_interests, size_home_page, access_frequency)

Layer-3: generalization of layer-2

- **cs_doc** (file_addr, authors, title, publication, publication_date, abstract, language, category_description, keywords, num_pages, form, size_doc, links_out)

- **doc_summary** (affiliation, field, publication_year, count, first_author_list, file_addr_list)

- **doc_author_brief** (file_addr, authors, affiliation, title, publication, pub_date, category_description, keywords, num_pages, format, size_doc, links_out)

- **person_summary** (affiliation, research_interest, year, num_publications, count)
XML and Web Mining

XML can help to extract the correct descriptors
Standardization would greatly facilitate information extraction

Potential problem:
XML can help solve heterogeneity for vertical applications, but the freedom to define tags can make horizontal applications on the Web more heterogeneous.
Benefits of Multi-Layer Meta-Web

Benefits:
- Multi-dimensional Web info summary analysis
- Approximate and intelligent query answering
- Web high-level query answering (WebSQL, WebML)
- Web content and structure mining
- Observing the dynamics/evolution of the Web

Is it realistic to construct such a meta-Web?
- Benefits even if it is partially constructed
- Benefits may justify the cost of tool development, standardization and partial restructuring
Web Usage Mining

Mining Web log records to discover user access patterns of Web pages

Applications
- Target potential customers for electronic commerce
- Enhance the quality and delivery of Internet information services to the end user
- Improve Web server system performance
- Identify potential prime advertisement locations

Web logs provide rich information about Web dynamics
- Typical Web log entry includes the URL requested, the IP address from which the request originated, and a timestamp
Techniques for Web usage mining

Construct multidimensional view on the Weblog database
   Perform multidimensional OLAP analysis to find the top $N$ users, top $N$ accessed Web pages, most frequently accessed time periods, etc.

Perform data mining on Weblog records
   Find association patterns, sequential patterns, and trends of Web accessing
   May need additional information, e.g., user browsing sequences of the Web pages in the Web server buffer

Conduct studies to
   Analyze system performance, improve system design by Web caching, Web page prefetching, and Web page swapping
Mining the World-Wide Web

Design of a Web Log Miner

Web log is filtered to generate a relational database
A data cube is generated form database
OLAP is used to drill-down and roll-up in the cube
OLAM is used for mining interesting knowledge
Mining Complex Types of Data

Mining time-series and sequence data
Mining the World-Wide Web
Mining spatial databases
Mining multimedia databases
Summary
Spatial Association Analysis

Spatial association rule: \( A \Rightarrow B [s\%, c\%] \)

A and B are sets of spatial or non-spatial predicates
- Topological relations: intersects, overlaps, disjoint, etc.
- Spatial orientations: left_of, west_of, under, etc.
- Distance information: close_to, within_distance, etc.

\( s\% \) is the support and \( c\% \) is the confidence of the rule

Examples
1) \( \text{is}_a(x, \text{large}_\text{town}) \land \text{intersect}(x, \text{highway}) \rightarrow \text{adjacent}_\text{to}(x, \text{water}) \) [7\%, 85\%]
2) What kinds of objects are typically located close to golf courses?
Progressive Refinement Mining of Spatial Association Rules

Hierarchy of spatial relationship:
- \textit{g\_close\_to}: near\_by, touch, intersect, contain, etc.

First search for rough relationship and then refine it

Two-step mining of spatial association:
- Step 1: Rough spatial computation (as a filter)
  - Using MBR or R-tree for rough estimation
- Step 2: Detailed spatial algorithm (as refinement)
  - Apply only to those objects which have passed the rough spatial association test (no less than \textit{min\_support})
Spatial Classification

Analyze spatial objects to derive classification schemes, such as decision trees, in relevance to certain spatial properties (district, highway, river, etc.)

- Classifying medium-size families according to income, region, and infant mortality rates
- Mining for volcanoes on Venus

Employ most of the methods in classification

- Decision-tree classification, Naïve-Bayesian classifier + boosting, neural network, genetic programming, etc.
- Association-based multi-dimensional classification - Example: classifying house value based on proximity to lakes, highways, mountains, etc.
Spatial Trend Analysis

Function
Detect changes and trends along a spatial dimension
Study the trend of non-spatial or spatial data changing with space

Application examples
Observe the trend of changes of the climate or vegetation with increasing distance from an ocean
Crime rate or unemployment rate change with regard to city geo-distribution
Spatial Cluster Analysis

Mining clusters—k-means, k-medoids, hierarchical, density-based, etc.
Analysis of distinct features of the clusters

[Map with pie charts showing data distribution across the US]
Constraint-Based Clustering: Planning ATM Locations

Spatial data with obstacles

Clustering *without* taking obstacles into consideration
Mining Complex Types of Data

Mining time-series and sequence data
Mining the World-Wide Web
Mining spatial databases
Mining multimedia databases

Summary
Similarity Search in Multimedia Data

Description-based retrieval systems

Build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation

Labor-intensive if performed manually

Results are typically of poor quality if automated

Content-based retrieval systems

Support retrieval based on the image content, such as color histogram, texture, shape, objects, and wavelet transforms
Queries in Content-Based Retrieval Systems

Image sample-based queries

Find all of the images that are similar to the given image sample

Compare the feature vector (signature) extracted from the sample with the feature vectors of images that have already been extracted and indexed in the image database

Image feature specification queries

Specify or sketch image features like color, texture, or shape, which are translated into a feature vector

Match the feature vector with the feature vectors of the images in the database
Refining or combining searches

Search for “blue sky”
(top layout grid is blue)

Search for “blue sky and green meadows”
(top layout grid is blue and bottom is green)

Search for “airplane in blue sky”
(top layout grid is blue and keyword = “airplane”)
Mining Multimedia Databases

The Data Cube and the Sub-Space Measurements

- Format of image
- Duration
- Colors
- Textures
- Keywords
- Size
- Width
- Height
- Internet domain of image
- Internet domain of parent pages
- Image popularity

- Group By Colour
- Measurement
- Cross Tab
- By Format
- By Colour
- By Colour & Size
- By Format & Colour
- By Format & Size
- Sum

- Dimensions

- RED
- WHITE
- BLUE

- JPEG
- GIF

- Small
- Medium
- Large
- Very Large
Mining Complex Types of Data

Mining time-series and sequence data
Mining the World-Wide Web
Mining spatial databases
Mining multimedia databases