DATA MINING

Introduction

Data mining refers to extracting or ―mining knowledge from large amounts of data.

Many other terms carry a similar or slightly different meaning to data mining, such as knowledge mining from data, knowledge extraction, data/pattern analysis, another popularly used term, Knowledge Discovery from Data, or KDD.

Essential step in the Process of knowledge discovery. **Knowledge discovery** as a process is depicted in Figure consists of an iterative sequence of the following steps:

Data cleaning: to remove noise and inconsistent data

Data integration: where multiple data sources may be combined

Data selection: where data relevant to the analysis task are retrieved from the database

Data transformation: where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance

Data mining: an essential process where intelligent methods are applied in order to extract data patterns

Pattern evaluation to identify the truly interesting patterns representing knowledge based on some interestingness measures;

Knowledge presentation where visualization and knowledge representation techniques are used to present the mined knowledge to the user

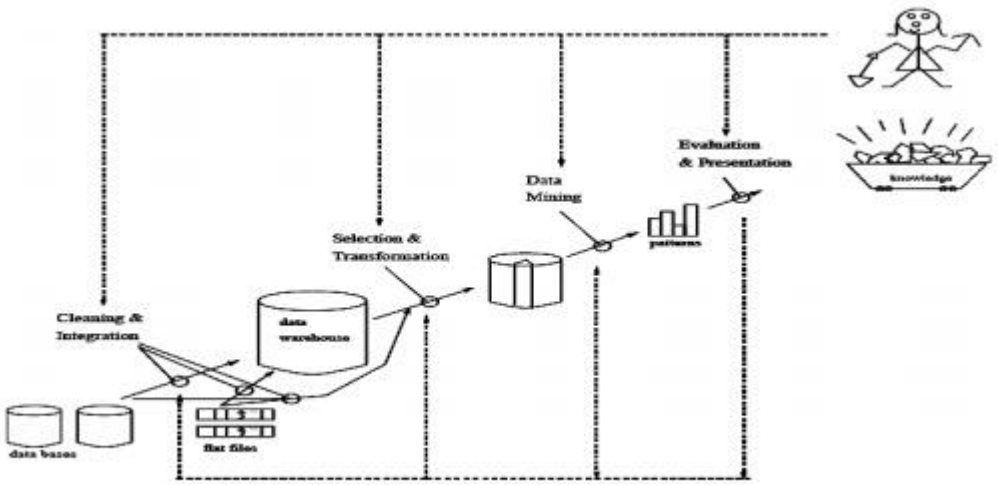


Figure: Data mining as a process of knowledge discovery.

Data Mining—On What Kind of Data? (Types of Data)

Relational Databases: A database system, also called a database management system (DBMS), consists of a collection of interrelated data, known as a database, and a set of software programs to manage and access the data.

.

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

Data Warehouses: A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site. Data warehouses are constructed via a process of data

cleaning, data integration, data transformation, data loading, and periodic data refreshing.

The data are stored to provide information from a *historical perspective* (such as from the past 5–10 years) and are typically *summarized*.

A data warehouse is usually modeled by a multidimensional database structure, where each dimension corresponds to an attribute or a set of attributes in the schema, and each cell stores the value of some aggregate measure, such as *count* or *sales amount*

The actual physical structure of a data warehouse may be a relational data store or a multidimensional data cube. A data cube provides a multidimensional view of data and allows the precomputation and fast accessing of summarized data

What is the difference between a data warehouse and a data mart

A data warehouse collects information about subjects that span an *entire organization*, and thus its scope is *enterprise-wide*.

A data mart, on the other hand, is a department subset of a data warehouse. It focuses on selected subjects, and thus its scope is *department-wide*. Data warehouse systems are well suited for on-line analytical processing, or OLAP. OLAP operations use background knowledge regarding the domain of the data being studied in order to allow the presentation of data at *different levels of abstraction*. Such operations accommodate different user viewpoints.

Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization,

Transactional Databases: Transactional database consists of a file where each record represents a transaction. A transaction typically includes a unique transaction identity number (*trans ID*) and a list of the items making up the transaction (such as items purchased in a store).

The transactional database may have additional tables associated with it, which contain other information regarding the sale, such as the date of the transaction, the customer ID number, the ID number of the salesperson and of the branch at which the sale occurred, and so on.

Advanced Data and Information Systems and Advanced Applications

The new database applications include handling spatial data (such as maps), engineering design data (such as the design of buildings, system components, or integrated circuits), hypertext and multimedia data (including text, image, video, and audio data), time-related data (such as historical records or stock exchange data), stream data (such as video surveillance and sensor data, where data flow in and out like streams), and the WorldWideWeb (a huge, widely distributed information repository made available by the Internet).

These applications require efficient data structures and scalable methods for handling complex object structures; variable-length records; semi structured or unstructured data; text, spatiotemporal, and multimedia data; and database schemas with complex structures and dynamic changes.

Object-Relational Databases: Object-relational databases are constructed based on an object-relational data model. This model extends the relational model by providing a rich data type for handling complex objects and object orientation object-relational databases are becoming increasingly popular in industry and applications.

The object-relational data model inherits the essential concepts of object-oriented databases. Each object has associated with it the following:

A set of variables that describe the objects. These correspond to attributes in the entity relationship and relational models.

A set of messages that the object can use to communicate with other objects, or with the rest of the database system.

A set of methods, where each method holds the code to implement a message. Upon receiving a message, the method returns a value in response. For instance, the method for the message *get photo*(*employee*) will retrieve and return a photo of the given employee object.

Objects that share a common set of properties can be grouped into an object class. Each object is an instance of its class. Object classes can be organized into class/subclass hierarchies so that each class represents properties that are common to objects in that class

Temporal Databases, Sequence Databases, and Time-Series Databases

A temporal database typically stores relational data that include time-related attributes. These attributes may involve several timestamps, each having different semantics.

A sequence database stores sequences of ordered events, with or without a concrete notion of time. Examples include customer shopping sequences, Web click streams, and biological sequences. A time series database stores sequences of values or events obtained over repeated measurements of time (e.g., hourly, daily, weekly). Examples include data collected from the stock exchange, inventory control, and the observation of natural phenomena (like temperature and wind).

Spatial Databases and Spatiotemporal Databases

Spatial databases contain spatial-related information. Examples include geographic (map) databases, very large-scale integration (VLSI) or computed-aided design databases, and medical and satellite image databases.

Spatial data may be represented in raster format, consisting of *n*-dimensional bit maps or pixel maps. For example, a 2-D satellite image may be represented as raster data, where each pixel registers the rainfall in a given area. Maps can be represented in vector format, where roads, bridges, buildings, and lakes are represented as unions or overlays of basic geometric constructs, such as points, lines, polygons, and the partitions and networks formed by these components.

Text Databases and Multimedia Databases

Text databases are databases that contain word descriptions for objects. These word descriptions are usually not simple keywords but rather long sentences or paragraphs, such as product specifications, error or bug reports, warning messages, summary reports, notes, or other documents.

Text databases may be highly unstructured (such as some Web pages on the WorldWideWeb). Some text databases may be somewhat structured, that is, *semistructured* (such as e-mail messages and many HTML/XML Web pages), whereas others are relatively well structured (such as library catalogue databases). Text databases with highly regular structures typically can be implemented using relational database systems.

Multimedia databases store image, audio, and video data. They are used in applications such as picture content-based retrieval, voice-mail systems, video-on-demand systems, the World Wide Web, and speech-based user interfaces that recognize spoken commands. Multimedia databases must support large objects, because data objects such as video can require gigabytes of storage. Specialized storage and search techniques are also required. Because video and audio data require real-time retrieval at a steady and predetermined rate in order to avoid picture or sound gaps and system buffer overflows, such data are referred to as continuous-media data.

The World Wide Web

The World Wide Web and its associated distributed information services, such as Yahoo!, Google, America Online, and AltaVista, provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access. Users seeking information of interest traverse from one object via links to another. Such systems provide ample opportunities and challenges for data mining.

For example, understanding user access patterns will not only help improve system design (by providing efficient access between highly correlated objects), but also leads to better marketing decisions (e.g., by placing advertisements in frequently visited documents, or by providing better customer/user classification and behavior analysis). Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining).

Data Mining Functionalities—What Kinds of Patterns Can Be Mined?

Data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. data mining tasks can be classified into two categories: descriptive and predictive.

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

Concept/Class Description: Characterization and Discrimination

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

Data characterization, by summarizing the data of the class under study (often called the target class) in general terms,

Data discrimination, by comparison of the target class with one or a set of comparative classes (often called the contrasting classes), or (3) both data characterization and discrimination.

Data characterization is a summarization of the general characteristics or features of a target class of data. The data corresponding to the user-specified class are typically collected by a database query the output of data characterization can be presented in various forms. Examples include pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs.

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

Mining Frequent Patterns, Associations, and Correlations

Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including itemsets, subsequences, and substructures.

A *frequent itemset* typically refers to a set of items that frequently appear together in a transactional data set, such as Computer and Software. A frequently occurring subsequence, such as the pattern that customers tend to

purchase first a PC, followed by a digital camera, and then a memory card, is a (*frequent*) *sequential pattern*.

Example: Association analysis. Suppose, as a marketing manager of *AllElectronics*, you would like to determine which items are frequently purchased together within the same transactions. An example of such a rule, mined from the *AllElectronics* transactional database, is

***buys*(*X*;—*computer*‖) *buys*(*X*; —*software*‖) [*support* = 1%, *confidence* = 50%]**

where *X* is a variable representing a customer. A confidence, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she will buy software as well. A 1% support means that 1% of all of the transactions under analysis showed that computer and software were purchased together. This association rule involves a single attribute or predicate (i.e., *buys*) that repeats. Association rules that contain a single predicate are referred to as single-dimensional association rules. Dropping the predicate notation, the above rule can be written simply as —*compute software* [1%, 50%]‖.

Classification and Prediction

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

The derived model may be represented in various forms, such as ***classification (IF-THEN) rules***, *decision trees*, *mathematical formulae*, or *neural networks*

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

A neural network, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. There are many other methods for constructing classification models, such as naïve

Bayesian classification , support vector machines, and *k*-nearest neighbor classification. Whereas classification predicts categorical (discrete, unordered) labels, prediction models Continuous-valued functions. That is, it is used to predict missing or unavailable *numerical data values* rather than class labels. Although the term *prediction* may refer to both numeric prediction and class label prediction

Cluster Analysis

Classification and prediction analyze class-labeled data objects, where as **clustering** analyzes data objects without consulting a known class label.

Outlier Analysis

A database may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or exceptions. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as outlier mining.

Evolution Analysis

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

Interestingness Of Patterns

A data mining system has the potential to generate thousands or even millions of patterns, or rules. then *"are all of the patterns interesting?"* Typically not—only a small fraction of the patterns potentially generated would actually be of interest to any given user.

This raises some serious questions for data mining. You may wonder, *"**What makes a pattern interesting? Can a data mining system generate all of the interesting patterns? Can a data mining system generate only interesting patterns?"***

To answer the first question, a pattern is interesting if it is

1)*easily understood* by humans,

(2)*valid* on new or test data with some degree of *certainty*,

potentially *useful*, an *novel*.

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

Several objective measures of pattern interestingness exist. These are based on the structure of discovered patterns and the statistics underlying them. An objective measure for association rules of the form $X \Rightarrow Y$ is rule support, representing the percentage of transactions from a transaction database that the given rule satisfies.

This is taken to be the probability $P(X \cup Y)$, where XUY indicates that a transaction contains both X and Y, that is, the union of itemsets X and Y. Another objective measure for association rules is confidence, which assesses the degree of certainty of the detected association. This is taken to be the conditional probability $P(Y \mid X)$, that is, the probability that a transaction containing X also contains Y. More formally, support and confidence are defined as

$$\textbf{\textit{support}}(X \Rightarrow Y) = P(XUY) \quad \textbf{\textit{confidence}}(X \Rightarrow Y) = P(Y \mid X)$$

In general, each interestingness measure is associated with a threshold, which may be controlled by the user. For example, rules that do not satisfy a

confidence threshold of, say, 50% can be considered uninteresting. Rules below the threshold threshold likely reflect noise, exceptions, or minority cases and are probably of less value.

The second question—***—Can a data mining system generate all of the interesting patterns****?*‖—refers to the completeness of a data mining algorithm. It is often unrealistic and inefficient for data mining systems to generate all of the possible patterns. Instead, user-provided constraints and interestingness measures should be used to focus the search.

Finally,

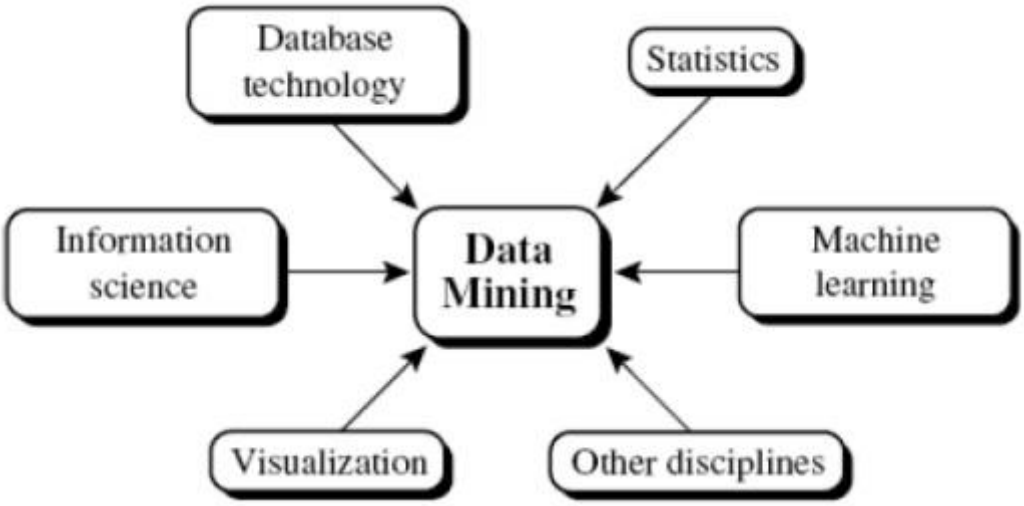
The third question—*"****Can a data mining system generate only interesting atterns****?"*—is an optimization problem in data mining. It is highly desirable for data mining systems to generate only interesting patterns. This would be much more efficient for users and data mining systems, because neither would have to search through the patterns generated in order to identify the truly interesting ones. Progress has been made in this direction; however, such optimization remains a challenging issue in data mining.

Classification of Data Mining Systems

Data mining is an interdisciplinary field, the confluence of a set of disciplines, including database systems, statistics, machine learning, visualization, and information science.

Moreover, depending on the data mining approach used, techniques from other disciplines may be applied, such as neural networks, fuzzy and/or rough set theory, knowledge representation, inductive logic programming, or high-performance computing. Depending on the kinds of data to be mined or on the given data mining application, the data mining system may also integrate techniques from spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, computer graphics, Web technology, economics, business, bioinformatics, or psychology.

Data mining systems can be categorized according to various criteria, as follows:



Classification according to the *kinds of databases* mined: A data mining system can be classified according to the kinds of databases mined. Database systems can be classified according to different criteria (such as data models, or the types of data or applications involved), each of which may require its own data mining technique. Data mining systems can therefore be classified accordingly.

Classification according to the *kinds of knowledge* mined: Data mining systems can be categorized according to the kinds of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis. A comprehensive data mining system usually provides multiple and/or integrated data mining functionalities.

Classification according to the *kinds of techniques* utilized: Data mining systems can be categorized according to the underlying data mining techniques employed. These techniques can be described according to the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems, query-driven systems) or the methods of data analysis employed (e.g., database-oriented or data warehouse– oriented techniques, machine learning, statistics, visualization, pattern recognition, neural

networks, and so on). A sophisticated data mining system will often adopt multiple data mining techniques or work out an effective, integrated technique that combines the merits of a few individual approaches.

Classification according to the *applications adapted*: Data mining systems can also be categorized according to the applications they adapt. For example, data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on. Different applications often require the integration of application-specific methods. Therefore, a generic, all-purpose data mining system may not fit domain-specific mining tasks.

Integration Of A Data Mining System With A Database Or Data Warehouse System

DB and DW systems, possible integration schemes include *no coupling*, *loose coupling*, *semitight coupling*, and *tight coupling*. We examine each of these schemes, as follows:

1. **No coupling:** *No coupling* means that a DM system will not utilize any function of a DB or DW system. It may fetch data from a particular source (such as a file system), process data using some data mining algorithms, and then store the mining results in another file.

2.Loose coupling: *Loose coupling* means that a DM system will use some facilities of a DB or DW system, fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data Warehouse. Loose coupling is better than no coupling because it can fetch any portion of data stored in databases or data warehouses by using query processing, indexing, and other system facilities.

However, many loosely coupled mining systems are main memory-based. Because mining does not explore data structures and query optimization methods provided by DB or DW systems, it is difficult for loose coupling to achieve high scalability and good performance with large data sets.

3.Semitight coupling: *Semi tight coupling* means that besides linking a DM system to a DB/DW system, efficient implementations of a few essential data mining primitives (identified by the analysis of frequently encountered data mining functions) can be provided in the DB/DW system. These primitives can include sorting, indexing, aggregation, histogram analysis, multi way join, and precomputation of some essential statistical measures, such as sum, count, max, min ,standard deviation,

4. Tight coupling: *Tight coupling* means that a DM system is smoothly integrated into the DB/DW system. The data mining subsystem is treated as one functional component of information system. Data mining queries and functions are optimized based on mining query analysis, data structures, indexing schemes, and query processing methods of a DB or DW system.

Data mining Task primitives.

A data mining query is defined in terms of the following primitives

Task-relevant data: This is the database portion to be investigated. For example, suppose that you are a manager of All Electronics in charge of sales in the United States and Canada. In particular, you would like to study the buying trends of customers in Canada. Rather than mining on the entire database. These are referred to as relevant attributes

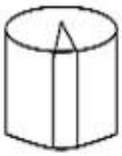
The kinds of knowledge to be mined: This specifies the data mining functions to be performed, such as characterization, discrimination, association, classification, clustering, or evolution analysis. For instance, if studying the buying habits of customers in Canada, you may choose to mine associations between customer profiles and the items that these customers like to buy

Background knowledge: Users can specify background knowledge, or knowledge about the domain to be mined. This knowledge is useful for guiding the knowledge discovery process, and for evaluating the patterns found. There are several kinds of background knowledge.

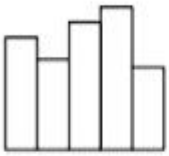
Interestingness measures: These functions are used to separate uninteresting patterns from knowledge. They may be used to guide the mining process, or after discovery, to evaluate the discovered patterns. Different kinds of knowledge may have different interestingness measures.

Presentation and visualization of discovered patterns: This refers to the form in which discovered patterns are to be displayed. Users can choose from different forms for knowledge presentation, such as rules, tables, charts, graphs, decision trees, and cubes.

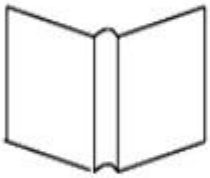
Figure : Primitives for specifying a data mining task.



Task-relevant data
- database or data warehouse name
- database tables or data warehouse cubes
- conditions for data selection
- relevant attributes or dimensions
- data grouping criteria

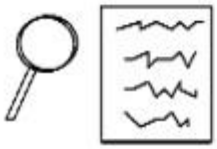


Knowledge type to be mined
- characterization
- discrimination
- association
- classification/prediction
- clustering



Background knowledge
- concept hierarchies
- user beliefs about relationships in the data

Pattern interestingness measurements
- simplicity
- certainty (e.g., confidence)
- utility (e.g., support)
- novelty



Visualization of discovered patterns
- rules, tables, reports, charts, graphs, decisison trees, and cubes
- drill-down and roll-up

Major Issues In Data Mining

1. Mining methodology and user-interaction issues. These react the kinds of knowledge mined, the ability to mine knowledge at multiple granularities, the use of domain knowledge, ad-hoc mining, and knowledge visualization.

Mining different kinds of knowledge in databases.

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

Interactive mining of knowledge at multiple levels of abstraction.

Since it is difficult to know exactly what can be discovered within a database, the data mining process should be interactive. For databases containing a huge amount of data, appropriate sampling technique can first be applied to facilitate interactive data exploration. Interactive mining allows users to focus the search for patterns, providing and refining data mining requests based on returned results. Specifically, knowledge should be mined by drilling-down, rolling-up, and pivoting through the data space and knowledge space interactively, similar to what OLAP can do on data cubes. In this way, the user can interact with the data mining system to view data and discovered patterns at multiple granularities and from different angles.

Incorporation of background knowledge.

Background knowledge, or information regarding the domain under study, may be used to guide the discovery process and allow discovered patterns to

be expressed in concise terms and at different levels of abstraction. Domain knowledge related to databases, such as integrity constraints and deduction rules, can help focus and speed up a data mining process, or judge the interestingness of discovered patterns.

Data mining query languages and ad-hoc data mining.

Relational query languages (such as SQL) allow users to pose ad-hoc queries for data retrieval. In a similar vein, high-level data mining query languages need to be developed to allow users to describe ad-hoc data mining tasks by facilitating the speci_cation of the relevant sets of data for analysis, the domain knowledge, the kinds of knowledge to be mined, and the conditions and interestingness constraints to be enforced on the discovered patterns. Such a language should be integrated with a database or data warehouse query language, and optimized for e_cient and exible data mining.

Presentation and visualization of data mining results.

Discovered knowledge should be expressed in high-level languages, visual representations, or other expressive forms so that the knowledge can be easily understood and directly usable by humans. This is especially crucial if the data mining system is to be interactive. This requires the system to adopt expressive knowledge representation techniques, such as trees, tables, rules, graphs, charts, crosstabs, matrices, or curves.

Handling outlier or incomplete data.

The data stored in a database may reect outliers | noise, exceptional cases, or incomplete data objects. These objects may confuse the analysis process, causing over_tting of the data to the knowledge modelconstructed. As a result, the accuracy of the discovered patterns can be poor. Data cleaning methods and data analysis methods which can handle outliers are required. While most methods discard outlier data, such data may be of interest in itself such as in fraud detection for Finding unusual usage of tele-communication services or credit cards. This form of data analysis is known as outlier mining.

Pattern evaluation: the interestingness problem.

A data mining system can uncover thousands of patterns. Many of the patterns discovered may be uninteresting to the given user, representing common knowledge or lacking novelty. Several challenges remain regarding the development of techniques to assess the interestingness of discovered patterns, particularly with regard to subjective measures which estimate the value of patterns with respect to a given user class, based on user beliefs or expectations. The use of interestingness measures to guide the discovery process and reduce the search space is another active area of research.

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

Efficiency and scalability of data mining algorithms.

To effectively extract information from a huge amount of data in databases, data mining algorithms must be efficient and scalable. That is, the running time of a data mining algorithm must be predictable and acceptable in large databases. Algorithms with exponential or even medium-order polynomial complexity will not be of practical use. From a database perspective on knowledge discovery, efficiency and scalability are key issues in the implementation of data mining systems. Many of the issues discussed above under mining methodology and user-interaction must also consider efficiency and scalability.

Parallel, distributed, and incremental updating algorithms.

The huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods are factors motivating the development of parallel and distributed data mining algorithms. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged. Moreover, the high cost of some data mining processes promotes the need

for incremental data mining algorithms which incorporate database updates without having to mine the entire data again \from scratch". Such algorithms perform knowledge modification incrementally to amend and strengthen what was previously discovered.

3. Issues relating to the diversity of database types.

Handling of relational and complex types of data.

There are many kinds of data stored in databases and data warehouses. Since relational databases and data warehouses are widely used, the development of efficient and effective data mining systems for such data is important. However, other databases may contain complex data objects, hypertext and multimedia data, spatial data, temporal data, or transaction data. It is unrealistic to expect one system to mine all kinds of data due to the diversity of data types and different goals of data mining. Specific data mining systems should be constructed for mining specific kinds of data. Therefore, one may expect to have different data mining systems for different kinds of data.

Mining information from heterogeneous databases and global information systems.

Local and wide-area computer networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases. The discovery of knowledge from di_erent sources of structured, semi-structured, or unstructured data with diverse data semantics poses great challenges to data mining. Data mining may help disclose high-level data regularities in multiple heterogeneous databases that are unlikely to be discovered by simple query systems and may improve information exchange and interoperability in heterogeneous databases.

Data Preprocessing

1 . Data Cleaning.

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

(i). Missing values

1. Ignore the tuple: This is usually done when the class label is missing (assuming the mining task involves classification or description). This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably.

2. Fill in the missing value manually: In general, this approach is time-consuming and may not be feasible given a large data set with many missing values.

3. Use a global constant to fill in the missing value: Replace all missing attribute values by the same constant, such as a label like ―Unknown". If missing values are replaced by, say, ―Unknown", then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common - that of ―Unknown". Hence, although this method is simple, it is not recommended.

4. Use the attribute mean to fill in the missing value: For example, suppose that the average income of All Electronics customers is $28,000. Use this value to replace the missing value for income.

5. Use the attribute mean for all samples belonging to the same class as the given tuple: For example, if classifying customers according to credit risk, replace the missing value with the average income value for customers in the same credit risk category as that of the given tuple.

6. Use the most probable value to fill in the missing value: This may be determined with inference-based tools using a Bayesian formalism or decision tree induction. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for income.

(ii). Noisy data

Noise is a random error or variance in a measured variable.

1. Binning methods:

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

In this example, the data for price are first sorted and partitioned into equi-depth bins (of depth 3). In smoothing by bin means, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9. Similarly, smoothing by bin medians can be employed, in which each bin value is replaced by the bin median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

(i).Sorted data for price (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34 (ii).Partition into (equi-width) bins:

Bin 1: 4, 8, 15

Bin 2: 21, 21, 24

Bin 3: 25, 28, 34

(iii).Smoothing by bin means:

Bin 1: 9, 9, 9,

Bin 2: 22, 22, 22

Bin 3: 29, 29, 29

(iv).Smoothing by bin boundaries:

Bin 1: 4, 4, 15

Bin 2: 21, 21, 24

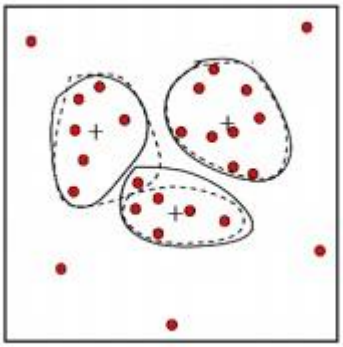
Bin 3: 25, 25, 34

2. Clustering:

Outliers may be detected by clustering, where similar values are organized into groups or

―clusters‖. Intuitively, values which fall outside of the set of clusters may be considered outliers.

Figure: Outliers may be detected by clustering analysis.



3. Regression: Data can be smoothed by fitting the data to a function, such as with regression.

Linear regression involves finding the ―best" line to fit two variables, so that one variable can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two variables are involved and the data are fit to a multidimensional surface.

(iii). Inconsistent data

There may be inconsistencies in the data recorded for some transactions. Some data inconsistencies may be corrected manually using external references. For example, errors made at data entry may be corrected by performing a paper trace. This may be coupled with routines designed to help correct the inconsistent use of codes. Knowledge engineering tools may also be used to detect the violation of known data constraints. For example, known functional dependencies between attributes can be used to find values contradicting the functional constraints.

Data Integration:

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components in data management. There are some problems to be considered during data integration.

- Schema integration: Integrates metadata(a set of data that describes other data) from different sources.
- Entity identification problem: Identifying entities from multiple databases. For example, the system or the use should know student _id of one database and student_name of another database belongs to the same entity.
- Detecting and resolving data value concepts: The data taken from different databases while merging may differ. Like the attribute values from one database may differ from another database. For example, the date format may differ like “MM/DD/YYYY” or “DD/MM/YYYY”.

2. Data Transformation.

In data transformation, the data are transformed or consolidated into forms appropriate for mining. Data transformation can involve the following:

i)Normalization, where the attribute data are scaled so as to fall within a small specified range, such as -1.0 to 1.0, or 0 to 1.0.

There are three main methods for data normalization : **min-max normalization, z-score**,**normalization, and normalization by decimal scaling.**

ii)Smoothing, which works to remove the noise from data? Such techniques include binning, clustering, and regression.

Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.

Generalization of the data, where low level or 'primitive' (raw) data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher level concepts, like city or county.

3. Data reduction.

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.

Strategies for data reduction include the following.

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

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

Dimension reduction, where encoding mechanisms are used to reduce the data set size.

Numerosity reduction, where the data are replaced or estimated by alternative, smaller data representations such as parametric models (which need

store only the model parameters instead of the actual data), or nonparametric methods such as clustering, sampling, and the use of histograms.

Discretization and concept hierarchy generation, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction, and are a powerful tool for data mining.

Data Cube Aggregation

The lowest level of a data cube 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 wit 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:

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:

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

Forward Selection

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

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

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

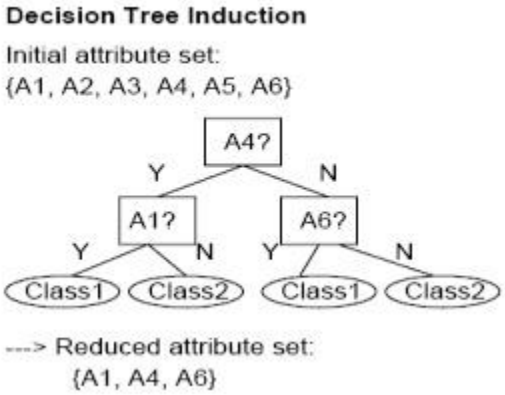
Backward Elimination

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

Combination forward selection and backward elimination: The step-wise forward selection and backward elimination methods can be combined, where at each step one selects the best attribute and removes the

Decision tree induction: Decision tree algorithms, such as ID3 and C4.5, were originally intended for classifcation. Decision tree induction constructs a flow-chart-like structure where each internal (non-leaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the

best" attribute to partition the data into individual classes.



Dimensionality Reduction

In data compression, data encoding or transformations are applied so as to obtain a reduced or ‖compressed" representation of the original data. If the original data can be reconstructed from the compressed data without any loss of information, the data compression technique used is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data compression technique is called lossy. The two popular and effective methods of lossy data compression**: wavelet transforms, and principal components analysis.**

Wavelet transforms

The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector D, transforms it to a numerically different vector, D0, of wavelet coefficients. The two vectors are of the same length.

The DWT is closely related to the discrete Fourier transform (DFT), a signal processing technique involving sines and cosines. In general, however, the DWT achieves better lossy compression.

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

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

Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference.

The two functions are applied to pairs of the input data, resulting in two sets of data of length L=2. In general, these respectively represent a smoothed version of the input data, and the high-frequency content of it.

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

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

Principal components analysis

Principal components analysis (PCA) searches for c k-dimensional orthogonal vectors that can best be used to represent the data, where c << N. The original data is thus projected onto a much smaller space, resulting in data compression. PCA can be used as a form of dimensionality reduction. The initial data can then be projected onto this smaller set.

The basic procedure is as follows.

The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.

PCA computes N orthonormal vectors which provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the principal components. The input data are a linear combination of the principal components.

The principal components are sorted in order of decreasing ―significance" or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance.

since the components are sorted according to decreasing order of ―significance", the size of the data can be reduced by eliminating the weaker components, i.e., those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

Numerosity reduction Regression and log-linear models

Regression and log-linear models can be used to approximate the given data. In linear regression, the data are modeled to fit a straight line. For example, a random variable, Y (called a response variable), can be modeled as a linear function of another random variable, X (called a predictor variable), with the equation where the variance of Y is assumed to be constant. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line.

Multiple regression is an extension of linear regression allowing a response variable Y to be modeled as a linear function of a multidimensional feature vector.

Log-linear models approximate discrete multidimensional probability distributions. The method can be used to estimate the probability of each cell in a base cuboid for a set of discretized attributes, based on the smaller cuboids making up the data cube lattice

Histograms

A histogram for an attribute A partitions the data distribution of A into disjoint subsets, or buckets. The buckets are displayed on a horizontal axis, while the height (and area) of a bucket typically reects the average frequency of the values represented by the bucket.

Equi-width: In an equi-width histogram, the width of each bucket range is constant (such as the width of $10 for the buckets in Figure 3.8).

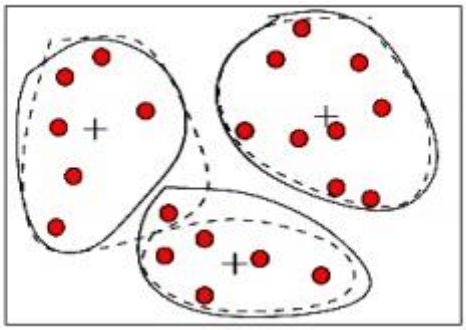
Equi-depth (or equi-height): In an equi-depth histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (that is, each bucket contains roughly the same number of contiguous data samples).

V-Optimal: If we consider all of the possible histograms for a given number of buckets, the V-optimal histogram is the one with the least variance. Histogram variance is a weighted sum of the original values that each bucket represents, where bucket weight is equal to the number of values in the bucket.

MaxDiff: In a MaxDiff histogram, we consider the difference between each pair of adjacent values. A bucket boundary is established between each pair for pairs having the Beta largest differences, where Beta-1 is user-specified.

Clustering

Clustering techniques consider data tuples as objects. They partition the objects into groups or clusters, so that objects within a cluster are ―similar" to one another and ―dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how ―close" the objects are in space, based on a distance function. The ―quality" of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster. Centroid distance is an alternative measure of cluster quality, and is defined as the average distance of each cluster object from the cluster centroid.



Sampling

Sampling can be used as a data reduction technique since it allows a large data set to be represented by a much smaller random sample (or subset) of the data.

Data Discretization and Concept hierarchy Generation

Data discretization techniques can be used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values. Replacing numerous values of a continuous attribute by a small number of interval labels thereby reduces and simplifies the original data. This leads to a concise, easy-to-use, knowledge-level representation of mining results.

Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up). If the discretization process uses class information, then we say it is supervised discretization. Otherwise, it is unsupervised. If the process starts by first finding one or a few points (called split points or cut points) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called top-down discretization or splitting. This contrasts with bottom-up discretization or merging, which starts by considering all of the continuous values as potential split-points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals. Discretization can be performed recursively on an attribute to provide a hierarchical or multi resolution partitioning of the attribute values, known as a concept hierarchy. Concept hierarchies are useful for mining at multiple levels of abstraction.

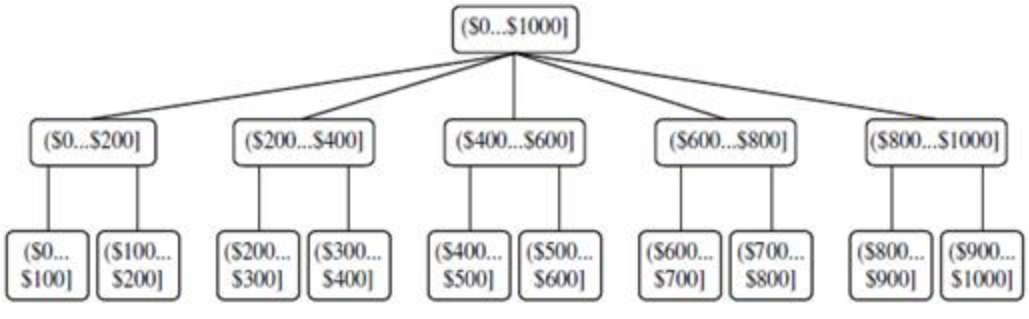


Figure 2.22 A concept hierarchy for the attribute *price*, where an interval ($X ... $Y] denotes the range from $X (exclusive) to $Y (inclusive).

A concept hierarchy for a given numerical attribute defines a discretization of the attribute. Concept hierarchies can be used to reduce the data by collecting

and replacing low-level concepts (such as numerical values for the attribute age) with higher-level concepts (such as youth, middle-aged, or senior). Although detail is lost by such data generalization, the generalized data may be more meaningful and easier to interpret. This contributes to a consistent representation of data mining results among multiple mining tasks, which is a common requirement. In addition, mining on a reduced data set requires fewer input/output operations and is more efficient than mining on a larger, un-generalized data set. Because of these benefits, discretization techniques and concept hierarchies are typically applied before data mining as a preprocessing step, rather than during mining. An example of a concept hierarchy for the attribute price is given in Figure 2.22.More than one concept hierarchy can be defined for the same attribute in order to accommodate the needs of various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, several discretization methods can be used to automatically generate or dynamically refine concept hierarchies for numerical attributes. Furthermore, many hierarchies for categorical attributes are implicit within the database schema and can be automatically defined at the schema definition level.