1.1 What is data mining?

The past two decades has seen a dramatic increase in the amount of information or data being stored in electronic format. This accumulation of data has taken place at an explosive rate. It has been estimated that the amount of information in the world doubles every 20 months and the size and number of databases are increasing even faster. The increase in use of electronic data gathering devices such as point-of-sale or remote sensing devices has contributed to this explosion of available data. Figure 1.1 from the Red Brick company illustrates the data explosion.

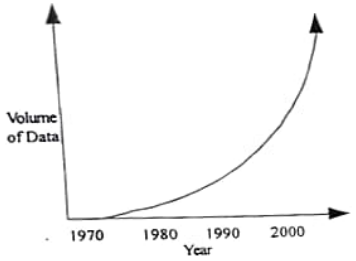


Figure 1.1 The Growing Base of Data

Data storage became easier as the availability of large amounts of computing power at low cost i.e. the cost of processing power and storage is falling, made data cheap. There was also the introduction of new machine learning methods for knowledge representation based on logic programming etc. in addition to traditional statistical analysis of data. The new methods tend to be computationally intensive hence a demand for more processing power.

Having concentrated so much attention on the accumulation of data the problem was what to do with this valuable resource? It was recognized that information is at the heart of business operations and that decision-makers could make use of the data stored to gain valuable insight into the business. Database Management systems gave access to the data stored but this was only a small part of what could be gained from the data. Traditional on-line transaction processing systems, OLTPs, are good at putting data into databases quickly, safely and efficiently but are not good at delivering meaningful analysis in return. Analyzing data can provide further knowledge about a business by going beyond the data explicitly stored to derive knowledge about the business. This is where Data Mining or Knowledge Discovery in Databases (KDD) has obvious benefits for any enterprise.

Basically data mining is concerned with the analysis of data and the use of software techniques for finding patterns and regularities in sets of data. It is the computer which is responsible for finding the patterns by identifying the underlying rules and features in the data. The idea is that it is possible to strike gold in unexpected places as the data mining software extracts patterns not previously discernable or so obvious that no-one has noticed them before.

Data mining analysis tends to work from the data up and the best techniques are those developed with an orientation towards large volumes of data, making use of as much of the collected data as possible to arrive at reliable conclusions and decisions. The analysis process starts with a set of data, uses a methodology to develop an optimal representation of the structure of the data during which time knowledge is acquired. Once knowledge has been acquired this can be extended to larger sets of data working on the assumption that the larger data set has a structure similar to the sample data. Again this is analogous to a mining operation where large amounts of low grade materials are sifted through in order to find something of value. The figure 1.2 summarises the some of the stages/processes identified in data mining.

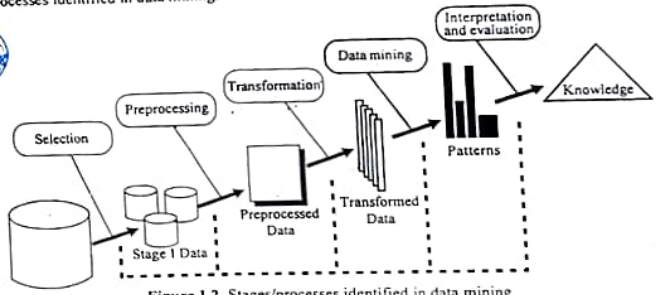


Figure 1.2 Stages/processes identified in data mining

The phases depicted start with the raw data and finish with the extracted knowledge which was acquired as a result of the following stages:

- **Selection** - selecting or segmenting the data according to some criteria e.g. all those people who own a car, in this way subsets of the data can be determined.
- **Preprocessing** - this is the data cleansing stage where certain information is removed which is deemed unnecessary and may slow down queries for example unnecessary to note the sex of a patient when studying pregnancy. Also the data is reconfigured to ensure a consistent format as there is a possibility of inconsistent formats because the data is drawn from several sources e.g. sex may recorded as f or m and also as 1 or 0.
- **Transformation** - the data is not merely transferred across but transformed in that overlays may added such as the demographic overlays commonly used in market research. The data is made useable and navigable.
- **Data mining** - this stage is concerned with the extraction of patterns from the data. A pattern can be defined as given a set of facts(data) F, a language L, and some measure of certainty C, a pattern is a statement S in L that describes relationships among a subset Fs of F with a certainty C such that S is simpler in some sense than the enumeration of all the facts in Fs.

- **Interpretation and evaluation** - the patterns identified by the system are interpreted into knowledge which can then be used to support human decision-making e.g. prediction and classification tasks, summarizing the contents of a database or explaining observed phenomena.

1.2 Data Mining Background

Data mining research has drawn on a number of other fields such as inductive learning, machine learning and statistics etc.

1.2.1 Inductive Learning

Induction is the inference of information from data and inductive learning is the model building process where the environment i.e. database is analyzed with a view to finding patterns. Similar objects are grouped in classes and rules formulated whereby it is possible to predict the class of unseen objects. This process of classification identifies classes such that each class has a unique pattern of values which forms the class description. The nature of the environment is dynamic hence the model must be adaptive i.e. should be able learn.

Generally it is only possible to use a small number of properties to characterize objects so we make abstractions in that objects which satisfy the same subset of properties are mapped to the same internal representation.

Inductive learning where the system infers knowledge itself from observing its environment has two main strategies:

- **supervised learning** - this is learning from examples where a teacher helps the system construct a model by defining classes and supplying examples of each class. The system has to find a description of each class i.e. the common properties in the examples. Once the description has been formulated the description and the class form a classification rule which can be used to predict the class of previously unseen objects. This is similar to discriminate analysis as in statistics.
- **unsupervised learning** - this is learning from observation and discovery. The data mine system is supplied with objects but no classes are defined so it has to observe the examples and recognize patterns (i.e. class description) by itself. This system results in a set of class descriptions, one for each class discovered in the environment. Again this similar to cluster analysis as in statistics.

Induction is therefore the extraction of patterns. The quality of the model produced by inductive learning methods is such that the model could be used to predict the outcome of future situations in other words not only for states encountered but rather for unseen states that could occur. The problem is that most environments have different states, i.e. changes within, and it is not always possible to verify a model by checking it for all possible situations.

1.2.2 Statistics

Statistics has a solid theoretical foundation but the results from statistics can be overwhelming and difficult to interpret as they require user guidance as to where and how to analyse the data. Data

mining however allows the expert's knowledge of the data and the advanced analysis techniques of the computer to work together.

Statistical analysis systems such as SAS and SPSS have been used by analysts to detect unusual patterns and explain patterns using statistical models such as linear models. Statistics have a role to play and data mining will not replace such analyses but rather they can act upon more directed analyses based on the results of data mining. For example statistical induction is something like the average rate of failure of machines.

1.2.3 Machine Learning

Machine learning is the automation of a learning process and learning is tantamount to the construction of rules based on observations of environmental states and transitions. This is a broad field which includes not only learning from examples, but also reinforcement learning, learning with teacher, etc. A learning algorithm takes the data set and its accompanying information as input and returns a statement e.g. a concept representing the results of learning as output. Machine learning examines previous examples and their outcomes and learns how to reproduce these and make generalizations about new cases.

Generally a machine learning system does not use single observations of its environment but an entire finite set called the training set at once. This set contains examples i.e. observations coded in some machine readable form. The training set is finite hence not all concepts can be learned exactly.

1.2.4 Differences between Data Mining and Machine Learning

Knowledge Discovery in Databases (KDD) or Data Mining, and the part of Machine Learning (ML) dealing with learning from examples overlap in the algorithms used and the problems addressed.

The main differences are:

- KDD is concerned with finding understandable knowledge, while ML is concerned with improving performance of an agent. So training a neural network to balance a pole is part of ML, but not of KDD. However, there are efforts to extract knowledge from neural networks which are very relevant for KDD.
- KDD is concerned with very large, real-world databases, while ML typically (but not always) looks at smaller data sets. So efficiency questions are much more important for KDD.
- ML is a broader field which includes not only learning from examples, but also reinforcement learning, learning with teacher, etc.

KDD is that part of ML which is concerned with finding understandable knowledge in large sets of real-world examples. When integrating machine learning techniques into database systems to implement KDD some of the databases require:

- More efficient learning algorithms because realistic databases are normally very large and noisy. It is usual that the database is often designed for purposes different from data mining and so properties or attributes that would simplify the learning task are not present nor can

they be requested from the real world. Databases are usually contaminated by errors so the data mining algorithm has to cope with noise whereas ML has laboratory type examples i.e. as near perfect as possible.

- More expressive representations for both data, e.g. tuples in relational databases, which represent instances of a problem domain, and knowledge, e.g. rules in a rule-based system, which can be used to solve users' problems in the domain, and the semantic information contained in the relational schema.

Practical KDD systems are expected to include three interconnected phases

- Translation of standard database information into a form suitable for use by learning facilities;
- Using machine learning techniques to produce knowledge bases from databases; and
- Interpreting the knowledge produced to solve users' problems and/or reduce data spaces. Data spaces being the number of examples.

1.3 Data Mining-On What Kind of Data?

Data mining should be applicable to any kind of information repository. This includes relational databases, data warehouses, transactional databases, advanced database systems, flat files, and the World Wide Web. Advanced database systems include object-oriented and object-relational databases, and specific application-oriented databases, such as spatial databases, time-series databases, text databases, and multimedia databases. The challenges and techniques of mining may differ for each of the repository systems.

1.3.1 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 pro-grams 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, which models the database as a set of entities and their relationships, is often constructed for relational databases.

Consider the following example.

Example1.1 : The *AllElectronics* Company is described by the following relation tables: *customer, item, employee,* and *branch.* Fragments of the tables described here are shown in Figure 1.3 For each relation, the attribute that represents the key or composite key component is underlined. The relation *customer* consists of a set of attributes, including a unique cus-tomer identity number *(cust_ID),* customer name, address, age, occupation, annual income, credit information, category, and so on. Similarly, each of the relations *item, employee,* and *branch* consists of a set of attributes, describing their properties.

Customer

cust_ID	Name	Address	Age	Income	Credit_info	...
C1	Smith, Sandy	5463 E Hastings, Burnaby, BC V5A 4S9, Canada	21	$27000	1	...
...	...		...	...	...	...

Item

Item_ID	Name	Branc	Category	Type	Price	Place_made	Supplier	Cost
I3	High-res-TV	Toshiba	High Resolution	TV	$988.00	Japan	Niko X	$600.00
I8	'Multidisc-	Sanyo	Multidisc	CD Player	$369.00	Japan	Music Front	$120.00
...	CD Play	...	...	...	...	...	...	...

Employee

Empl_ID	Name	Category	Group	Salary	Commission
E55	Jones, Jane	Home Entertainment	Manager	$18,000	2%
...	...	...	...	...	...

Branch

Branch_ID	Name	Address
B1	City Square	369 Cambie., Vancouver, BC V5L 3A2, Canada
...	...	...

Purchases

Trans_ID	Cust_ID	Empl_ID	Data	Time	Method_paid	Amount
T100	C1	E55	09/21/98	15:45	Visa	$1357.00
...	...	...	...	...	...	...

Items_sold

Trans_ID	Item_ID	Qty.
T100	I3	1
T100	I8	2
...	...	...

Works_at

Empl_ID	Branch_ID
E55	B1
...	...

Figure 1.3 Fragments of relations from a relational database for ALLElectronics

Tables can also be used to represent the relationships between or among mul-tiple relation tables. For our example, these include *purchases* (customer pur-chases items, creating a sales transaction that is handled by an employee), *items_sold* (lists the items sold in a given transaction), and *works_at* (employee works at a branch of *AllElectronics).*

Relational data can be accessed by database queries written in a relational query language, such as SQL, or with the assistance of graphical user interfaces. In the latter, the user may employ a menu, for example, to specify attributes to be included in the query, and the constraints on these attributes. A given query is transformed into a set of relational operations, such as join, selection,

and projection, and is then optimized for efficient processing. A query allows retrieval of specified subsets of the data. Suppose that your job is to analyze the *AllElectronics* data. Through the use of relational queries, you can ask things like "Show me a list of all items that were sold in the last quarter." Relational languages also include aggregate functions such as sum, avg (average), count, max (maximum), and min (minimum). These allow you to ask things like "Show me the total sales of the last month, grouped by branch," or "How many sales transactions occurred in the month of December?" or "Which sales person had the highest amount of sales?".

When data mining is applied to relational databases, one can go further by *searching for trends or data patterns*. For example, data mining systems may analyze customer data to predict the credit risk of new customers based on their income, age, and previous credit information. Data mining systems may also detect deviations, such as items whose sales are far from those expected in comparison with the previous year. Such deviations can then be further investigated (e.g., has there been a change in packaging of such items, or a significant increase in price?). Relational databases are one of the most popularly available and rich information repositories, and thus they are a major data form in our study of data mining.

1.3.2 Data Warehouses

Suppose that *AllElectronics* is a successful international company, with branches around the world. Each branch has its own set of databases. The president of *AllElectronics* has asked you to provide an analysis of the company's sales per item type per branch for the third quarter. This is a difficult task, particularly since the relevant data are spread out over several databases, physically located at numerous sites.

If *AllElectronics* had a data warehouse, this task would be easy. A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and which usually resides at a single site. Data warehouses are constructed via a process of data cleaning, data transformation, data integration, data loading, and periodic data refreshing. Figure 1.4,1.5, 1.6,1.7 shows the data warehouse model, its structure, an example and data warehouse for *AllElectronics*.

Figure 1.4 A data warehouse model

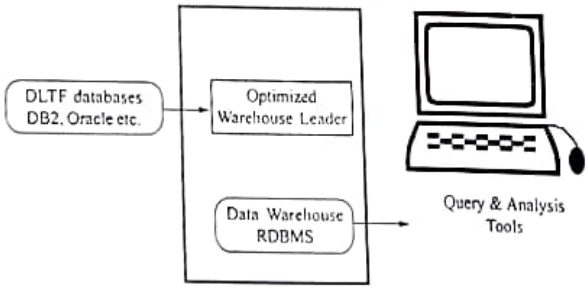


Figure 1.5 The structure of data inside the data warehouse

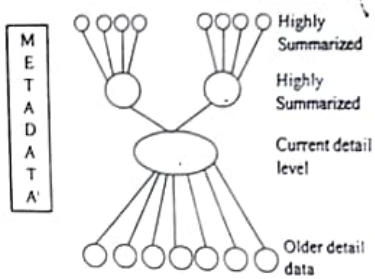


Figure 1.6 An example of levels of summarization of data inside the data warehouse

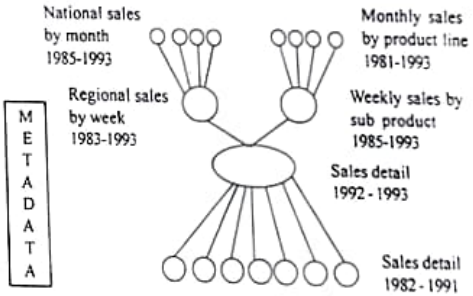
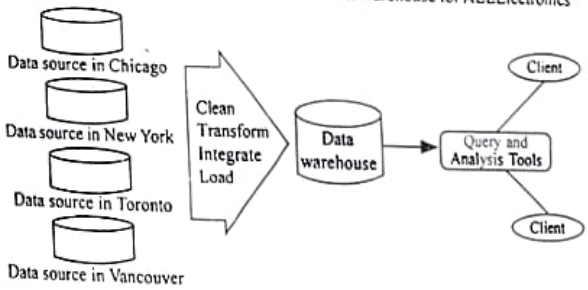
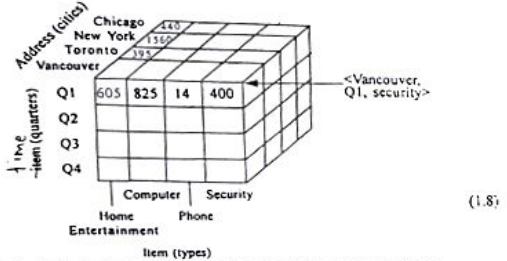


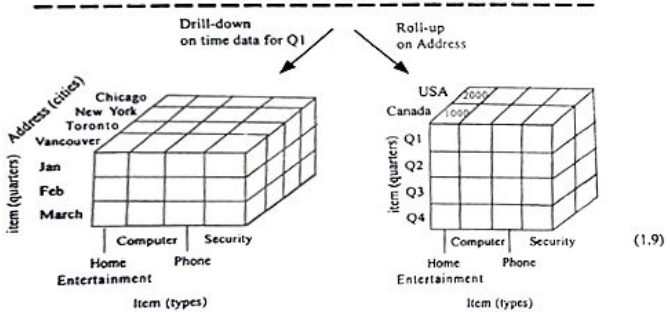
Figure 1.7 Typical architecture of a data warehouse for ALLElectronics



Example 1.2 : A data cube for summarized sales data of *AllElectronics* is presented in Figure 1.8 & 1.9. The cube has three *dimensions:* address (with *city* values *Chicago, New York, Toronto, Vancouver),* time (with *quarter* values Q1, Q2, Q3, Q4), and item (with item *type* values *home entertainment, computer, phone, security).* The aggregate value stored in each cell of the cube is *sales_amount* (in thousands). For example, the total sales for the first quarter, Q 1, for items relating to security systems in Vancouver is $400,000, as stored in cell *(Vancouver,* Q1, *security).* Additional cubes may be used to store aggregate sums over each dimension, corresponding to the aggregate values obtained using different SQL group-bys (e.g., the total sales amount per city and quarter, or per city and item, or per quarter and item, or per each individual dimension).



(1.8)



(1.9)

A Multidimensional data cube, commonly used for data warehousing, **Figure 1.8** Showing summarized data for ALLElectronics and Figure 1.9 Showing summarized data resulting from drill down and roll-up operations on the cube in 1.8

By providing multidimensional data views and the precomputation of summarized data, data warehouse systems are well suited for On-Line Analytical Processing, or OLAP. OLAP operations make use of background knowledge regarding the domain of the data being studied in order to allow the presentation of data at *different levels of abstraction.* Such operations accommodate different user viewpoints. Examples of OLAP operations include drill-down and roll-up, which allow the user to view the data at differing degrees of summarization, as illustrated in Figure 1.9. For instance, one may drill down on sales data summarized by *quarter* to see the data summarized by *month.* Similarly, one may roll up on sales data summarized by *city* to view the data summarized by *country.*

Although data warehouse tools help support data analysis, additional tools for data mining are required to allow more in-depth and automated analysis.

1.3.3 Transactional Databases

In general, a 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 sales person and of the branch at which the sale occurred, and so on.

Example 1.3 : Transactions can be stored in a table, with one record per transaction. A fragment of a transactional database for *AllElectronics* is shown in Figure 1.10. From the relational database point of view, the *sales* table in Figure 1.10 is a nested relation because the attribute *list of item_IDs* contains a set of *items.*

trans_ID	list of item_IDs
T100	11,13,18,116
...	...

Figure 1.10 Fragment of a transactional database for sales at ALLElectronics

As an analyst of the *AllElectronics* database, you may like to ask, "Show me all the items purchased by Sandy Smith" or "How many transactions include item number 13?." Answering such queries may require a scan of the entire transactional database.

Suppose you would like to dig deeper into the data by asking, "Which items sold well together?" This kind of *market basket data analysis* would enable you to bundle groups of items together as a strategy for maximizing sales. For example, given the knowledge that printers are commonly purchased together with computers, you could offer an expensive model of printers at a discount to customers buying selected computers, in the hopes of selling more of the expensive printers. A regular data retrieval system is not able to answer queries like the one above. However, data mining systems for transactional data can do so by identifying sets of items that are frequently sold together.

1.4 Data Mining - Advanced Database Systems and Advanced Database Applications

Relational database systems have been widely used in business applications. With the advances of database technology, various kinds of advanced database systems have emerged and are undergoing development to address the requirements of new database 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), and the World Wide Web (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 and multimedia data, and database schemas with complex structures and dynamic changes.

In response to these needs, advanced database systems and specific application oriented database systems have been developed. These include object-oriented and object-relational database systems, spatial database systems, temporal and time-series database systems, text and multimedia database systems, heteroge-neous and legacy database systems, and Web-based global information systems.

Object-Oriented Databases

Object-oriented databases are based on the object-oriented programming paradigm, where in general terms, each entity is considered as an object. Following the *AllElectronics* example, objects can be individual employees, customers, or items. Data and code relating to an object are *encapsulated* into a single unit. 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. For instance, an *employee* class can contain variables like *name, address,* and *birthdate.* Suppose that the class *sales_person* is a subclass of the class *employee.* A *sales_person* object would inherit all of the variables pertaining to its superclass of *employee.* In addition, it has all of the variables that pertain specifically to being a sales person (e.g., *commission).* Such a class inheritance feature benefits information sharing.

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. In addition, special constructs for relational query languages are included to manage the added data types. The object-relational model extends the basic relational data model by adding the power to handle complex data types, class hierarchies, and object inheritance as described above. Object-relational databases are becoming increasingly popular in industry and applications.

Data mining in object-oriented and object-relational systems share some similarities. In comparison with relational data mining, techniques need to be de-veloped for handling complex object structures, complex data types, class and subclass hierarchies, property inheritance, and methods and procedures.

Spatial Databases

A Spatial database contains spatial-related information. Such databases include geographic (map) databases, VLSI chip 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 of basic geometric constructs, such as points, lines, polygons, and the partitions and networks formed by these shapes.

Geographic databases have a number of applications, ranging from forestry and ecology planning, to providing public service information regarding the location of telephone and electric cables, pipes, and sewage systems. In addition, geographic databases are used in vehicle navigation and dispatching systems. An example of such a system for taxis would store a city map with information regarding one-way streets, suggested routes for moving from region A to region B during rush hour, the location of restaurants and hospitals, as well as the current location of each driver.

"*What kind of data mining can be performed on spatial databases?*" you may ask. Data mining may uncover patterns describing the characteristics of houses located near a specified kind of location, such as a park, for instance. Other patterns may describe the climate of mountainous areas located at various altitudes, or describe the change in trend of metropolitan poverty rates based on city distances from major highways. In addition, "spatial data cubes" may be constructed to organize data into multidimensional structures and hierarchies, on which OLAP operations (such as drill-down and roll-up) can be performed.

Temporal Databases and Time-Series Databases

Temporal databases and time-series databases both store time-related data. A temporal database usually stores relational data that include time-related attributes. These attributes may involve several timestamps, each having different semantics. A time-series database stores sequences of values that change with time, such as data collected regarding the stock exchange.

Data mining techniques can be used to find the characteristics of object evolution or the trend of changes for objects in the database. Such information can be useful in decision making and strategy planning. For instance, the mining of banking data may aid in the scheduling of bank tellers according to the volume of customer traffic. Stock exchange data can be mined to uncover trends that could help you plan investment strategies (e.g., when is the best time to purchase *AllElectronics* stock?). Such analyses typically require defining multiple granularities of time. For example, time may be decomposed according to fiscal years, academic years, or calendar years. Years may be further decomposed into quarters or months.

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 World Wide Web). Some text databases may be somewhat structured, that is, *semi structured* (such as e-mail messages and many HTML/XML Web pages), while others are relatively well structured (such as library databases). Text databases with highly regular structures typically can be implemented using relational database systems.

"*What can data mining on text databases uncover?*" Ultimately, it may uncover general descriptions of object classes, as well as keyword or content associations, and the clustering behavior of text objects. To do this, standard data mining methods need to be integrated with information retrieval techniques and the construction or use of hierarchies specifically for text data (such as dictionaries and thesauruses), as well as discipline-oriented term classification systems (such as in chemistry, medicine, law, or economics).

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, since data objects such as video can require gigabytes of storage. Specialized storage and search techniques are also required. Since 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.

Heterogeneous Databases and Legacy Databases

Objects in one component database may differ greatly from objects in other component databases, making it difficult to assimilate their semantics into the overall heterogeneous database. Many enterprises acquire legacy databases as a result of the long history of information technology development (including the application of different hard- wares and operating systems). A legacy database is groups of heterogeneous data-bases that combine different kinds of data systems, such as relational or object oriented databases, hierarchical databases, network databases, spreadsheets, mul-timedia databases, or file systems. The heterogeneous databases in a legacy data-base may be connected by intra-or inter-computer networks. Information exchange across such databases is difficult since one need to work out precise transformation rules from one representation to another, considering diverse semantics.

Consider, for example, the problem in exchanging information regarding student academic performance among different schools. Each school may have its own computer system and use its own curriculum and grading system. One university may adopt a quarter system, offer three courses on database systems, and assign grades from A + to F, while another may adopt a semester system, offer two courses on databases, and assign grades from 1 to 10.

It is very difficult to work out precise course-to-grade transformation rules between the two universities, making information exchange difficult. Data mining techniques may provide an interesting solution to the information exchange problem by transforming the given data into higher, more generalized, conceptual levels (such as *fair, good,* or *excellent* for student grades), from which information exchange can then more easily be performed.

The World Wide Web

The World Wide Web and its associated distributed information services, such as America Online, Yahoo!, AltaVista, and Prodigy, provide rich, world-wide, 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 mining path traversal patterns.

Although Web pages may appear fancy and informative to human readers, they can be highly unstructured and lack a predefined schema, type, or pattern. Thus it is difficult for computers to understand the semantic meaning of diverse Web pages and structure them in an organized way for systematic information retrieval and data mining. Web services that provide keyword-based searches without understanding the context behind particular Web pages can only offer limited help to users.

For example, a Web search based on a single keyword may return hundreds of Web page pointers containing the keyword, but most of the pointers will be unrelated to what the user wants to find. Can data mining provide more help here than Web search services? Can data mining help us learn about the distribution of information on the Web in general, Web page characteristics, and associations among different Web pages? Can it help find authoritative Web pages on any specific topic? Can it produce a good classification of pages on the Internet? These questions pose additional challenging issues for advanced data mining.

1.5 Data Mining Functions

Data mining methods may be classified by the function they perform or according to the class of application they can be used in. Some of the main techniques used in data mining are described in this section.

Classification

Data mine tools have to infer a model from the database, and in the case of supervised learning this requires the user to define one or more classes. The database contains one or more

attributes that denote the class of a tuple and these are known as predicted attributes whereas the remaining attributes are called predicting attributes. A combination of values for the predicted attributes defines a class.

When learning classification rules the system has to find the rules that predict the class from the predicting attributes so firstly the user has to define conditions for each class, the data mine system then constructs descriptions for the classes. Basically the system should given a case or tuple with certain known attribute values be able to predict what class this case belongs to.

Once classes are defined the system should infer rules that govern the classification therefore the system should be able to find the description of each class. The descriptions should only refer to the predicting attributes of the training set so that the positive examples should satisfy the description and none of the negative. A rule said to be correct if its description covers all the positive examples and none of the negative examples of a class.

A rule is generally presented as, if the left hand side (LHS) then the right hand side (RHS), so that in all instances where LHS is true then RHS is also true, are very probable. The categories of rules are:

- exact rule - permits no exceptions so each object of LHS must be an element of RHS
- strong rule - allows some exceptions, but the exceptions have a given limit
- probabilistic rule - relates the conditional probability P(RHS|LHS) to the probability P(RHS)

Other types of rules are classification rules where LHS is a sufficient condition to classify objects as belonging to the concept referred to in the RHS.

Associations

Given a collection of items and a set of records, each of which contain some number of items from the given collection, an association function is an operation against this set of records which return affinities or patterns that exist among the collection of items. These patterns can be expressed by rules such as "72% of all the records that contain items A, B and C also contain items D and E". The specific percentage of occurrences (in this case 72) is called the confidence factor of the rule. Also, in this rule, A,B and C are said to be on an opposite side of the rule to D and E. Associations can involve any number of items on either side of the rule.

A typical application, identified by IBM, that can be built using an association function is Market Basket Analysis. This is where a retailer run an association operator over the point of sales transaction log, which contains among other information, transaction identifiers and product identifiers. The set of products identifiers listed under the same transaction identifier constitutes a record. The output of the association function is, in this case, a list of product affinities. Thus, by invoking an association function, the market basket analysis application can determine affinities such as "20% of the time that a specific brand toaster is sold, customers also buy a set of kitchen gloves and matching cover sets".

Another example of the use of associations is the analysis of the claim forms submitted by patients to a medical insurance company. Every claim form contains a set of medical procedures that were performed on a given patient during one visit. By defining the set of items to be the collection

of all medical procedures that can be performed on a patient and the records to correspond to each claim form, the application can find, using the association function, relationships among medical procedures that are often performed together.

Sequential/Temporal patterns

Sequential/temporal pattern functions analyze a collection of records over a period of time for example to identify trends. Where the identity of a customer who made a purchase is known an analysis can be made of the collection of related records of the same structure (i.e. consisting of a number of items drawn from a given collection of items). The records are related by the identity of the customer who did the repeated purchases. Such a situation is typical of a direct mail application where for example a catalogue merchant has the information, for each customer, of the sets of products that the customer buys in every purchase order. A sequential pattern function will analyze such collections of related records and will detect frequently occurring patterns of products bought over time. A sequential pattern operator could also be used to discover for example the set of purchases that frequently precedes the purchase of a microwave oven.

Sequential pattern mining functions are quite powerful and can be used to detect the set of customers associated with some frequent buying patterns. Use of these functions on for example a set of insurance claims can lead to the identification of frequently occurring sequences of medical procedures applied to patients which can help identify good medical practices as well as to potentially detect some medical insurance fraud.

Clustering/Segmentation

Clustering and segmentation are the processes of creating a partition so that all the members of each set of the partition are similar according to some metric. A cluster is a set of objects grouped together because of their similarity or proximity. Objects are often decomposed into an exhaustive and/or mutually exclusive set of clusters.

Clustering according to similarity is a very powerful technique, the key to it being to translate some intuitive measure of similarity into a quantitative measure. When learning is unsupervised then the system has to discover its own classes i.e. the system clusters the data in the database. The system has to discover subsets of related objects in the training set and then it has to find descriptions that describe each of these subsets.

There are a number of approaches for forming clusters. One approach is to form rules which dictate membership in the same group based on the level of similarity between members. Another approach is to build set functions that measure some property of partitions as functions of some parameter of the partition.

1.6 Classification of Data Mining Systems

Data mining is an interdisciplinary field, the confluence of a set of disciplines (as shown in Figure 1.11), 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.

Because of the diversity of disciplines contributing to data mining, data mining research is expected to generate a large variety of data mining systems. Therefore, it is necessary

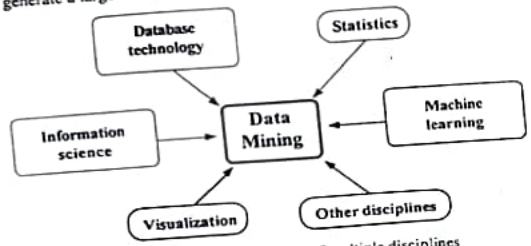


Figure 1.11 Data mining as a confluence of multiple disciplines

to provide a clear classification of data mining systems. Such a classification may help potential users distinguish data mining systems and identify those that best match their needs. 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 themselves 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.

For instance, if classifying according to data models, we may have a relational, transactional, object-oriented, object-relational, or data warehouse mining system. If classifying according to the special types of data handled, we may have a spatial, time-series, text, or multimedia data mining system, or a World Wide Web mining system.

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, classification, clustering, outlier analysis, and evolution analysis. A comprehensive data mining system usually provides multiple and/or integrated data mining functionalities. Moreover, data mining systems can be distinguished based on the granularity or levels of abstraction of the knowledge mined, including generalized knowledge (at a high level of abstraction), primitive-level knowledge (at a raw data level), or knowledge at multiple levels (considering several levels of abstraction). An

advanced data mining system should facilitate the discovery of knowledge at multiple levels of abstraction.

1.7 Data mining problems/issues

Data mining systems rely on databases to supply the raw data for input and this raises problems in that databases tend be dynamic, incomplete, noisy, and large. Other problems arise as a result of the adequacy and relevance of the information stored.

Limited Information

A database is often designed for purposes different from data mining and sometimes the properties or attributes that would simplify the learning task are not present nor can they be requested from the real world. Inconclusive data causes problems because if some attributes essential to knowledge about the application domain are not present in the data it may be impossible to discover significant knowledge about a given domain. For example cannot diagnose malaria from a patient database if that database does not contain the patients red blood cell count.

Noise and missing values

Databases are usually contaminated by errors so it cannot be assumed that the data they contain is entirely correct. Attributes which rely on subjective or measurement judgments can give rise to errors such that some examples may even be misclassified. Error in either the values of attributes or class information are known as noise. Obviously where possible it is desirable to eliminate noise from the classification information as this affects the overall accuracy of the generated rules.

Missing data can be treated by discovery systems in a number of ways such as;

- simply disregard missing values
- omit the corresponding records
- infer missing values from known values
- treat missing data as a special value to be included additionally in the attribute domain
- or average over the missing values using Bayesian techniques.

Noisy data in the sense of being imprecise is characteristic of all data collection and typically fit a regular statistical distribution such as Gaussian while wrong values are data entry errors. Statistical methods can treat problems of noisy data, and separate different types of noise.

Uncertainty

Uncertainty refers to the severity of the error and the degree of noise in the data. Data precision is an important consideration in a discovery system.

Size, updates, and irrelevant fields

Databases tend to be large and dynamic in that their contents are ever changing as information is added modified or removed. The problem with this from the data mining perspective is how to ensure that the rules are up-to-date and consistent with the most current information. Also the

iii) the mean, ... spread of

learning system has to be time-sensitive as some data values vary over time and the discovery system is affected by the 'timeliness' of the data.

Another issue is the relevance or irrelevance of the fields in the database to the current focus of discovery for example post codes are fundamental to any studies trying to establish a geographical connection to an item of interest such as the sales of a product.

Summary

Database technology has evolved from primitive file processing to the development database management systems with query and transaction processing. Further progress has led to the increasing demand for efficient and effective data analysis and understanding tools. This need is a result of the result of the explosive growth in data collected from applications including business and management, government administration, science and engineering, and environmental control.

Data Mining is the task of discovering interesting patterns from large amount of data where the data can be stored in databases, data warehouses, or other information repositories. It is a young interdisciplinary field, drawing from areas such as database systems, data warehouse statistics, machine learning, data visualization, information retrieval, and high-performance computing. Other areas include neural networks, pattern recognition, spatial data analysis, image databases, signal processing, and many application fields, such as business, economics, and bioinformatics. A Knowledge discovery process includes data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation, and knowledge presentation.

Data patterns can be mined from any different kinds of database, such as relaticnal databases, data warehouses, and transactional, object-relational, and object-oriented databases. Interesting data patterns can also be extracted from other kinds of information repositories, including spatial, time-related, text, multimedia, and legacy databases, and the World Wide Web.

A data warehouse is a repository for a long-term storage of data from multiple sources, organized so as to facilitate management decision making. The data are stored under a unified schema and are typically summarized. Data warehouse systems provide some data analysis capabilities, collectivity referred to as OLAP (On-line analylitical Processing).

Data mining functionalities include the discovery of concept/class descriptions, association, classification, prediction, clustering, trend analysis, deviation analysis, and similarity analysis. Characterization and discrimination are forms of data summarization.

A pattern represents knowledge if it is easily understood by humans; valid on test data with some degree of certainty; and potentially useful, novel, or validates a hunch about which the user was curious. Measures of patterns interestingness, either objective or subjective, can he used to guide the discovery process.

Data mining systems can be classified according to the kinds of databases mined, the kinds of knowledge mined, techniques used or the applications adapted.

Efficient and effective data mining in large databases poses numerous requirements and great challenges to researches and developers. The issues involved include data mining methodology, user interaction, performance and scalability, and the processing of a large variety of data types. Other issues include the exploration of data mining applications and their social impacts.

Exercises

1. What is data mining? In your answer, address the following :
   (a) Is it another type?
   (b) Is it a simple transformation of technology developed from databases, statistics, and machine learning?
   (c) Explain how evolution of database technology led to data mining.
   (d) Describe the steps involved in data mining when viewed as a process of knowledge discovery.
2. Present an example where data mining is crucial to success of a business. What data mining functions does this business need? Can they be performed alternatively by data query processing or simple statistical analysis?
3. Suppose your task as a software engineer at Big-University is to design a data mining system to examine their university course database, which contains the following information: the name, address, and status (e.g., undergraduate or graduate) of each student, the courses taken, and their cumulative grade point average (GPA). Describe the architecture you would choose. What is the purpose of each component of this architecture.
4. How is data warehouse different from a database? How are they similar?
5. Briefly describe the following advanced database systems and applications: object-oriented databases, spatial databases, text database, multimedia databases, the World Wide Web.
6. Define each of the following data mining functionalities: association, classification, clustering. Give examples of each data mining functionality, using a real-life database that you are familiar with.
7. What is the difference between discrimination and association? Between and clustering? Between classification and prediction? For each of these pairs of tasks, how they are similar?
8. Describe three challenges to data mining regarding data mining methodology and user interaction issues.
9. Describe two challenges to data mining regarding performance issues.

Objective

1) **Many people treat data mining as synonym for another popularly used term**
   a) **Knowledge Discovery in databases**
   b) knowledge inventory in databases
   c) Knowledge acceptance in databases
   d) knowledge disposal in databases.

2) A database is a collection of
   a) Related data
   b) Interrelated data
   c) Irrelevant data
   d) Distributed data

3) A Relational database is a collection of
   a) tables
   b) events
   c) attributes
   d) values

4) A _____ is a repository of information collected from multiple squares stored under a unified schema, and which usually resides at a single site.
   a) Data mining
   b) Database
   c) Data warehouse
   d) legacy databases

5) Which of the following databases is used to store image, audio, and video data?
   a) Heterogeneous databases
   b) Temporal databases
   c) Legacy databases
   d) Multimedia databases

6) Which of the following issues relate to the diversity of database type?
   a) Handling noisy or incomplete data
   b) Incorporation of background knowledge
   c) Handling of relational and complex types of data
   d) Efficiency and scalability of data mining algorithms

7) Which of the following is not major issue in data mining?
   a) Mining methodology and user interaction issues
   b) Performance issues
   c) Issues relating to the diversity of database types
   d) Issues relating to the Measurement

8) Processing _____ queries in operational databases would substantially degrade the performance of operational tasks.
   a) On-Line Transaction Processing
   b) On-Line Electronic Processing
   c) On-Line Data Processing
   d) On-Line Analytical Processing

9) An _____System typically adopts either a star or snow flake model and subject oriented database design.
   a) On-Line Transaction Processing
   b) On-Line Electronic Processing
   c) On-Line Analytical Processing
   d) On-Line Data Processing

10) The access patterns of an _____system consist mainly of short, atomic transactions.
   a) On-Line Analytical Processing
   b) On-Line Transaction Processing
   c) On-Line Electronic Processing
   d) On-Line Data Processing

11) Which of the following approach requires complex information filtering and integration processes and competes for resources with processing at local sources?
   a) Update-driven approach
   b) Integrate-driven approach
   c) Query-driven approach
   d) Data-driven approach

12) Mining different kinds of knowledge in databases is an issue in
   a) Performance issue
   b) Mining methodology and user interaction issues
   c) Diversity of database types issues
   d) time complexity

13) Pattern evolution is an issue related to
   a) Mining methodology and user interaction issues
   b) Performance issues
   c) Issues relating to the diversity of database types
   d) Issues relating to the Measurement

14) A DWH is a subject oriented, integrated, time- variant, and _____ collection of data in support of management's decision-making process.
   a) Nonvolatile
   b) Volatile
   c) Disintegrated
   d) Object- oriented

15) An ___system focuses mainly on the current data with in an enterprise or department, without referring to historical data or data in different organizations.
   a) On-Line Analytical Processing
   b) On-Line Data Processing
   c) On-Line Electronic Processing
   d) On-Line Transaction Processing

16) **The basic characteristic of On-line Analytical Processing is**
   a) **Informational processing**
   b) Operational processing
   c) Data processing
   d) Data cleaning

17) **Which of the following cuboid that holds the highest level of summarization?**
   a) Cuboid
   b) Base cuboid
   c) Non-base cuboid
   d) Apex cuboid

18) **_____ is a visualization operation that rotates the data axes in view in order to provide an alternative presentation of the data**
   a) Rollup
   b) Drill down
   c) **Pivot**
   d) Slice & dice

19) **_____ tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.**
   a) Fact
   b) Summarized
   c) **Dimension**
   d) Relational

20) **_____ executes queries involving more than one fact table**
   a) Drill-through
   b) **Drill-across**
   c) Drill-down
   d) Rotate

21) **A _____ allows data to be modeled and viewed in multiple dimensions.**
   a) Meta data
   b) **Data cube**
   c) Database
   d) Fact table

22) **The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model image kept in ____ form**
   a) Standard
   b) De-normalized
   c) **Normalized**
   d) Multi dimensional

23) **Which of the following is constructed where the enterprise warehouse is the sole custodian of all warehouse data. Which is then distributed to the various dependent data marts.**
   a) Enterprise DWH
   b) Two- tier DWH
   c) **Multi-tier DWH**
   d) Virtual warehouse

24) **The _____ view includes fact tables and dimension tables.**
   a) **DWH**
   b) Top-down
   c) Data source
   d) Business Query

25) **Which of the following is a Hybrid OLAP server?**
   a) MS SQL server 1.0
   b) MS SQL 5.0
   c) **MS SQL server 7.0**
   d) MS SQL server 3.0

26) **ETL stands for**
   a) Evaluate, Transport and Link
   b) **Extract Transfer and Load**
   c) Error, Tracking and Load
   d) Extract, Transient and Load

27) **To architect the DWH, the major driving factor to support is**
   a) An inability to cope with requirements evolution
   b) Not populating the warehouse
   c) **Day- to- day management of the warehouse**
   d) Supporting Online Transaction processing

28) **A _____ contains a subset of corporate-wide data that is of value to a specific group of users.**
   a) Enterprise warehouse
   b) Virtual warehouse
   c) Data warehouse
   d) **Data mart**

29) **A _____ is a set of views over operational databases**
   a) Enterprise warehouse
   b) **Virtual warehouse**
   c) Data warehouse
   d) Data mart

30) **What kind of the intermediate servers that stand in between a relational back-end server and client front-end tools?**
   a) Hybrid OLAP servers
   b) Multidimensional OLAP server
   c) **Relational OLAP servers**
   d) Specialized SQL servers

31) **Choose the _____ that will populate each fact table record**
   a) **Measures**
   b) Dimensions
   c) Grain
   d) Business Process

32) **Meta data repository contains**
   a) **Operational meta data**
   b) Data irrelevant to system performance
   c) The mapping from the DWH to the operational environment
   d) Summarized data

33) **A set of attributes in a relation schema that forms a primary key for another relation schema is called a _____**
   a) Primary key
   b) **Foreign key**
   c) Secondary key
   d) Composite key

34) **Which of the following typically gathers data from multiple, heterogeneous, and external sources?**
   a) Data cleaning
   b) Load
   c) Refresh
   d) **Data extraction**

35) **OLAM is particularly important for the following reason**
   a) How quality of data in DWH
   b) Data processing
   c) OLTP-based exploratory data analysis
   d) **Online selection of data mining functions**

36) **Data warehouse application is _____**
   a) Data Processing
   b) Transaction Processing
   c) Data cube
   d) **Data mining**

References

U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996.

J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM, 39:58-64, 1996.

G. Piatetsky-Shapiro, U. Fayyad, and P. Smith. From data mining to knowledge discovery: An overview. In U.M. Fayyad, et al. (eds.), Advances in Knowledge Discovery and Data Mining, 1-35. AAAI/MIT Press, 1996.

G. Piatetsky-Shapiro and W. J. Frawley. Knowledge Discovery in Databases. AAAI/MIT Press, 1991.

Agarwal S., Agarwal R., Deshpande P.M ., and Gupta A. On the computation of multidimensional aggregates.

Virmani A. Second Generation Data Mining: Concepts and implementation;Ph.D Thesis, Rutgers University, April 1998.

Barbara D.(ed) Special Issue on mining of Large Datasets; IEEE Data Engineering Bulletin,21(1),1998.

Nestorov S., and Tsur S. Integrating data mining with relational DBMS: A tightly coupled approach, www-db.stanford.

Heckerman D. Bayesian networks for data mining. Data Mining and knowledge Discovery,1997.

Agarwal R, Gupta A., and Sarawagi S. modeling multidimensional databases ,ICDE 1997.

DATA PREPROCESSING

At Present databases are highly susceptible to noisy missing and inconsistent data due to their typically huge size, often several gigabytes or more. *"How can the data be preprocessed in order to help improve the quality of the data and, consequently, of the mining results?"* you may wonder. *"How can the data be preprocessed so as to improve the efficiency and ease of the mining process?"*

There are a number of data preprocessing techniques.

- *Data cleaning* can be applied to remove noise and correct inconsistencies in the data. *Data integration* merges data from multiple sources into a coherent data store, such as a data warehouse or a data cube.
- *Data transformations*, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.
- *Data reduction* can reduce the data size by aggregating, eliminating redundant features, or clustering, for instance. These data processing techniques, when applied prior to mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

We concentrate methods *for* data preprocessing. These meth-ods are organized into the following categories: data cleaning, data integration and transformation, and data reduction. The use of hierarchies for data discretization, an alternative *form* of data reduction. Concept hierarchies can be further used to promote mining at multiple levels of abstraction. We concentrate how concept hierarchies can be generated automatically from the given data.

1.8 Why Preprocess the Data?

Imagine that you are a manager at *AllElectronics* and have been charged with analyzing the company's data with respect to the sales at your branch. You immediately set out to perform this task. You carefully inspect the company's database and data warehouse, identifying and selecting the attributes or dimensions he included in your analysis, such as *item*, *price*, and *units_sold*. Notice that several of the attributes for various tuples have no recorded value.

For your analysis, you would like to include information as to whether each item purchased was advertised as on sale, yet you discover that this information has not been recorded. Furthermore, users of your database system have reported errors, unusual values, and inconsistencies in the data recorded for some transactions. To analyze by data mining techniques are incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data), noisy (containing errors, or *outlier* values that deviate from the expected), and inconsistent (e.g., containing discrepancies in the department codes used to categorize items).

Incomplete, noisy, and inconsistent data are commonplace properties of large real-world databases and data warehouses. Incomplete data can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Other data may not be included simply because it was not considered important at the time of entry.

Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions. Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the history or modifications to the data may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.

There are many possible reasons for noisy data (having incorrect attribute values). The data collection instruments used may be faulty. There may have been inhuman or computer errors occurring at data entry. Errors in data transmission can also occur. There may be technology limitations, such as limited buffer size for coordinating synchronized data transfer and consumption. Incorrect data may also result from inconsistencies in naming conventions or data codes used. Duplicate tuples also require data cleaning.

Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying of removing outliers, and resolving inconsistencies. Dirty data can cause confusion for the mining procedure, resulting in unreliable output. Although most mining routines have some procedures for dealing with incomplete or noisy data, they are not always robust. Instead, they may concentrate on avoiding overfitting the data to the function being modeled.

At AllElectronics, suppose that you would like to include data from multiple sources in your analysis. This would involve integrating multiple databases, data cubes, or files, that is, data integration. Yet some attributes representing a given concept may have different names in different databases, causing inconsistencies and redundancies. For example, the attribute for customer identification may be referred to as customerid in one data store, and cust_id in another. Naming inconsistencies may also occur for attribute values.

For example, the same first name could be registered as "Sachin" in one database, but "Ram" in another, and "Krish." in the third. Furthermore, you suspect that some attributes may be inferred from others (e.g., annual revenue). Having a large amount of redundant data may slow down or confuse the knowledge discovery process. Clearly, in addition to data cleaning, steps must be taken to help avoid redundancies during data integration. Typically, data cleaning and data integration are performed as a preprocessing step when preparing the data for a data warehouse. Additional data cleaning may be performed to detect and remove redundancies that may have resulted from data integration.

Getting back to your data, you have decided, say, that you would like to use a distance-based mining algorithm for your analysis, such as neural networks, nearest-neighbor classifiers, or clustering.) Such methods provide better results if the data to be analyzed have been *normalized*, that is, scaled to a specific range such as [0.0, 1.0].

Your customer data, for example, contain the attributes *age* and *annual salary*. The *annual salary* attribute can take many more values than *age*. Therefore, if the attributes are left unnormalized, then distance measurements taken on *annual salary* will generally outweigh distance measurements taken on *age*. Furthermore, it would be useful for your analysis to obtain aggregate information as to the sales per customer region-something that is not part of any pre computed data cube in your data warehouse. You soon realize that data transformation operations, such as normalization and aggregation, are additional data preprocessing procedures that world contribute towards the success of the mining process.

"The data set I have selected for analysis is huge-it is sure to slow down the mining process. Is there any way I can reduce the size of my data set, without jeopardizing the data mining results?" Data reduction obtains a reduced representation of the data set that is much smaller in volume, yet produces the same (or almost the same) analytical results. There are a number of strategies for data reduction. These include *data aggregation* (e.g., building a data cube), *dimension reduction* (e.g., removing irrelevant attributes through correlation analysis), *data compression* (e.g., using encoding schemes such as minimum length encoding or wavelets), data compression *reduction* (e.g., "replacing" the data by alternative, smaller representations such as clusters or parametric models).

Data can also be "reduced" by *generalization*, where low-level concepts, such as *city* for customer location, are replaced with higher-level concepts, such as *region* or *province_or_state*. A concept hierarchy is used to organize the concepts into varying levels of abstraction. Since concept hierarchies are so useful in mining at multiple levels of abstraction, we devote a separate section to the automatic generation of this important data structure.

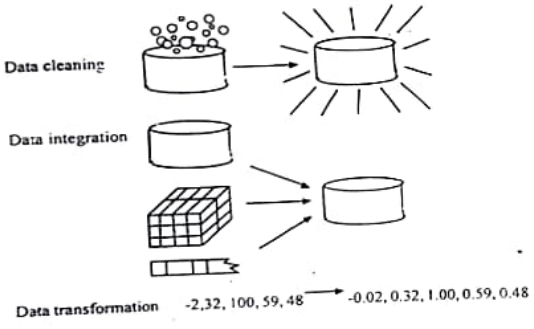


Figure 1.12 Forms of data preprocessing

Figure 1.12 summarizes the data preprocessing steps described here. Note that the above categorization is not mutually exclusive. For example, the removal of redundant data may be seen as a form of data cleaning, as well as data reduction.

In summary real-world data tend to be dirty, incomplete, and inconsistent. Data preprocessing techniques can improve the quality of the data, there by helping to improve the accuracy and efficiency of the subsequent mining process. Data preprocessing is an important step in the knowledge discovery process, since quality decisions must be based on Quality data. Detecting data anomalies, rectifying them early, and reducing the data to be analyzed for data making can lead to payoffs for decision making.

1.9 Data Cleaning

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

1.9.1 Missing Values

Imagine that you need to analyze *AllElectronics* sales and customer data. You note that many tuples have no recorded value for several attributes, such as customer *income*. To fill in the missing values for this attribute we use the following methods:

1. **Ignore the tuple:** 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"* or $-\infty$. If missing values are replaced by, say, *"Unknown,"* then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common-that of *"Unknown."* Hence, although this method is simple, it is not recommended.
4. **Use the attribute mean to fill in the missing value:** For example, suppose that the average income of *AllElectronics* 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 as the given tuple:** For example, if classifying customers according to *credit_risk*, replace the missing value with that 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 deter-mined with regression, inference-based tools using a Bayesian formalism, or decision tree induction. For example, using the other customer attributes in year data set, you may construct a decision tree to predict the missing values for *income*.

Methods 3 to 6 bias the data. The filled in value may not be correct. By considering the values of the other attributes in its estimation of the missing value for *income*, there is a greater chance that the relationships between *Income* and the other attributes are preserved.

1.9.2 Noisy Data

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

1. **Binning:** Binning methods smooth a sorted data value by consulting its "neighborhood," that is, the values around it. The sorted values are distributed into a number of "buckets," or *bins*. Because binning methods consult the neighborhood of values, they perform *local* smoothing. Figure 1.12.1 illustrates some binning techniques. In this example, the data for *price* are first sorted and then partitioned into *equidepth* bins of depth 3 (i.e., each bin contains three values). In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.

For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9. Similarly, smoothing by bin medians can be employed, in which each bin value is replaced by the bin median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing. Alternatively, bins may be equiwidth, where the interval range of values in each bin is constant.

Sorted data for price (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34

Partition into (equidepth) bins:
Bin 1 : 4, 8, 15
Bin 2 : 21, 21, 24
Bin 3 : 25, 28, 34

Smoothing by bin means:

Bin 1 : 9, 9, 9
Bin 2 : 22, 22, 22
Bin 3 : 29, 29, 29

Smoothing by bin boundaries

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

Figure 1.12.1 Binding Methods for data smoothing.

2. **Clustering:** Outliers may be detected by clustering, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers (Figure 1.13).

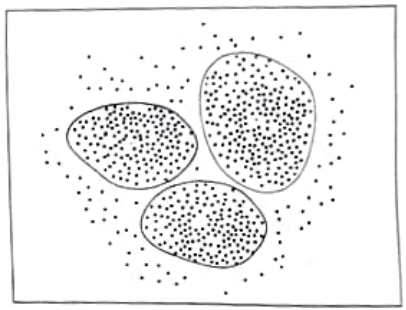


Figure 1.13 Outliers may be detected by clustering analysis

3. **Combined computer and human inspection:** Outliers may be identified through a combination of computer and human inspection. In one applica-tion, for example, an information-theoretic measure was used to help identify outlier patterns in a handwritten character database for classification. The measure's value reflected the "surprise" content of the predicted character label with respect to the known label. Outlier patterns may be informative (e.g., identifying useful data exceptions, such as different versions of the char-acters "0" or "7") or "garbage" (e.g., mislabeled characters).

Patterns whose surprise content is above a threshold are output to a list. A human can then sort through the patterns in the list to identify the actual garbage ones. This is much faster than having to manually search through the entire database. The garbage patterns can then be excluded *from* use in subsequent data mining.

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

Many methods for data smoothing are also methods for data reduction involving discretization. For example, the binning techniques described above reduce the number of distinct values per attribute. This acts as a form of data reduction for logic based data mining methods, such as decision tree induction, which repeatedly make value comparisons on sorted data. Concept hierarchies are a form of data discretization that can also be used for data smoothing. A concept hierarchy for *price*,... for example, may map real *price* values into *inexpensive*, *moderately_priced*, and *expensive*, there by reducing the number of data values to be handled by the mining process. Some methods of classification, such as neural networks, have built in data smoothing mechanisms.

1.9.3 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, values 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 contradicting the functional constraints. There may also be inconsistencies due to data integration, where a given at-tribute can have different names in different databases. Redundancies may also exist.

1.10 Data Integration and Transformation

Data mining often requires data integration-the merging of data from multiple data stores. The data may also need to be transformed into forms appropriate for mining. This section describes both data integration and data transformation.

1.10.1 Data Integration

It is likely that your data analysis task will involve *data integration*, which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files.

There are a number of issues to consider during data integration. *Schema integration* can be tricky. How can equivalent real-world entities from multiple data sources be matched up? This is referred to as the **entity identification problem**. For example, how can the data analyst or the computer be sure that *customer_id* in one database and *cust_number* in another refer to the same entity? Databases and data warehouses typically have metadata-that is, data about the data. Such metadata can be used to help avoid errors in schema integration.

Redundancy is another important issue. An attribute may be redundant if it can be "derived" from another table, such as *annual revenue*. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set. Some redundancies can be detected by **correlation analysis**. For example, given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. The correlation between attributes A and B can be measured by

$$r_{A,B} = \frac{\sum (A - \bar{A})(B - \bar{B})}{(n-1)\sigma_A \sigma_B} \qquad \text{(1.1)}$$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective mean values of A and B, and σ_A and σ_B are the respective standard deviations of A and B. If the resulting value of equation (1.1) is greater than 0, then A and B are positively correlated, meaning that the values of A increase as the values of B increase. The higher the value, the more each attribute implies the other. Hence, a high value may indicate that A (or B) may be removed as a redundancy. If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, where the values of one attribute increases the values of the other attribute decrease. This means that each attribute discourages the other. Equation (1.1) may detect a correlation between the *customer_id* and *cust_number* attributes described above.

In addition to detecting redundancies between attributes, duplication should also be detected at the tup'e level (e.g., where there are two or more identical tuples for a given unique data entry case). A third important issue in data integration is the *detection and resolution of data value conflicts*. For example, for the same real-world entity, attribute values from different sources may differ.

The mean of A is $\bar{A} = \frac{\sum A}{n}$

The standard deviation of A is

$$\sigma A = \sqrt{\frac{\sum (A - \bar{A})^2}{n-1}}$$

This may be due to differences in representation, scaling, or encoding. For instance, a *weight* attribute may be stored in metric un in one system and British imperial units in another. The *price* of different hob may involve not only different currencies but also different services (such as free breakfas:) and taxes. Such semantic heterogeneity of data poses great challenges, in data integration. Careful integration of the data from multiple sources can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent mining process.

1.10.2 Data Transformation

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

- **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. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.
- **Generalization of the data**, where low-level or "primitive" (raw) data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like *street*, can be generalized to higher level concepts, like *city* or *country*. Similarly, values for numeric attributes, like *age*, may be mapped to higher-level concepts, like *young, middle-aged*, and *senior*.
- **Normalization**, where the attribute data are scaled so as to fall within a small specified range, such as -1.0 to 1.0, 'or 0.0 to 1.0.

- **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.

Smoothing is a form of data cleaning. Aggregation and generalization also serve as forms of data reduction, respectively. we therefore discuss normalization and attribute construction.

An attribute is normalized by scaling its values so that they fall within a small specified range, such as 0.0 to 1.0. Normalization is particularly useful for classification algorithms involving neural networks, or distance measurements such as nearest neighbor classification and clustering. For distance-based methods, normalization helps prevent attributes with initially large ranges (e.g., *income*) from outweighing attributes with initially smaller ranges (e.g., binary attributes). There are many methods for data normalization. We study three: *min-max normalization*, *z-score normalization*, and *normalization by decimal scaling*.

Min-max normalization performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute A. Min-max normalization maps a value v of A to v' in the range $[new\_min_A, new\_max_A]$ by computing

$$v' = (v - min_A)/(max_A - min_A)(new\_max_A - new\_min_A) + new\_min_A \quad \text{..... (1.2)}$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out of bounds" error if a future input case for normalization falls outside of the original data range for A.

Example 1.4

Suppose that the minimum and maximum values for the attribute *income* are $12,000 and $98,000, respectively. We would like to map *income* to the range [0.0, 1.0]. By min-max normalization, a value of $73,600 for *income* is transformed to (73,600-12,0000/98,000-12,000)(1.0 - 0) + 0 = 0.716.

In **z-score normalization** (or *zero-mean normalization*), the values for an attribute A are normalized based on the mean and standard deviation of A. A value v of A is normalized to v' by computing

$$V' = \frac{V - \bar{A}}{\sigma A} \quad \text{..... (1.3)}$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.

Example 1.5

Suppose that the mean and standard deviation of the values for the attribute *income* are $54,000 and $16,000, respectively. With z-score normalization, a value of $73,600 for *income* is transformed to (73,600-54,000/16,000) = 1.225.

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

$$v' = (V/10^j) \quad \text{..... (1.4)}$$

where j is the smallest integer such that Max(| v'|) < 1.

Example 1.6

Suppose that the recorded values of A range from -986 to 917. The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide value by 1000 (i.e., j = 3) so that -986 normalizes to -0.986.

Normalization can change the original data a bit, especially the latter two methods shown above. It is also necessary to save the normalization parameters (such as the mean and standard deviation if using z-score normalization) so that future data can be normalized in a uniform manner.

In attribute construction, new attributes are constructed from the given attributes and added in order to help improve the accuracy and understanding of structure in high-dimensional data. For example, we may wish to add the attribute *area* based on the attributes *height* and *width*. Attribute construction can help alleviate the *fragmentation problem* when decision tree algorithms are used for classification, where an attribute is repeatedly tested along a path in the derived decision tree.) Examples of operators for attribute construction include **and** for binary attributes and **product** for nominal attributes. By combining attributes, attribute construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

1.11 Data Reduction

We have selected data from the *AllElectronics* data warehouse for analysis. The data set is huge. Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible.

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

Strategies for data reduction include the following:

1. **Data cube aggregation**, where aggregation operations are applied to the data in the construction of a data cube.
2. **Dimension reduction**, where irrelevant, weakly relevant, or redundant at-tributes or dimensions may be detected and removed.
3. **Data compression**, where encoding mechanisms are used to reduce the data.

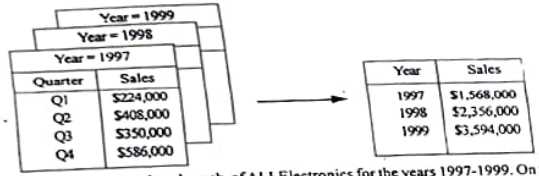


Figure 1.14 Sales data for a given branch of ALLElectronics for the years 1997-1999. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales

4. **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.
5. **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.

The computational time spent on data reduction should not outweigh or "erase" the time saved by mining on a reduced data set size.

1.11.1 Data Cube Aggregation

We have collected the data for our analysis. These data consist of the *AllElectronics* sales per quarter, for the years 1997 to 1999. If you are, interested in the annual sales (total per year), rather than the total per quarter. Thus the data can be *aggregated* so that the resulting data summarize the total sales per year instead of per quarter. This aggregation is illustrated in Figure 1.14. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.

Data cubes store multidimensional aggregated information. For example, Figure 1.15 shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch.

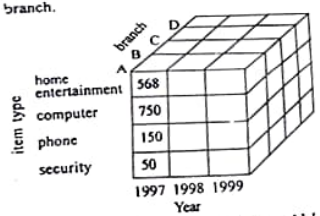


Figure 1.15 A data Cube for sales at ALLElectronics

Each cell holds an aggregate data value, corresponding to the data point in multidimensional space. Concept hierarchies may exist for each attribute, allowing the analysis of data at multiple levels of abstraction. For example, a hierarchy for *branch* could allow branches to be grouped into regions, based on their address. Data cubes provide fast access to precomputed, summarized data, thereby benefiting on-line analytical processing as well as data mining.

The cube created at the lowest level of abstraction is referred to as the *base cuboid*. A cube for the highest level of abstraction is the *apex cuboid*. For the sales data of Figure 1.15, the apex cuboid would give one total-the total *sales* for all three years, for all item types, and for all branches. Data cubes created for varying levels of abstraction are often referred to as *cuboids*, so that a data cube may instead refer to a *lattice of cuboids*. Each higher level of abstraction further reduces the resulting data size.

The base cuboid should correspond to an individual entity of interest, such as *sales* or *customer*. In other words, the lowest level should be usable, or useful for the analysis. Since data cubes provide fast accessing to precomputed, summarized data, they should be used when possible to reply to queries regarding aggregated information. When replying to such OLAP queries or data mining requests, the *smallest* available cuboid relevant to the given task should be used.

1.11.2 Dimensionality Reduction

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task, or redundant. For example, if the task is to classify customers as to whether or not they are likely to purchase a popular new CD at *AllElectronics* when notified of a sale, attributes such as the customer's telephone number are likely to be irrelevant, unlike attributes such as *age* or *music_taste*. Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time-consuming task, especially when the behavior of the data is not well known (hence, a reason behind its analysis!). Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the mining algorithm employed. This can result in discovered patterns of poor quality. In addition, the added volume of irrelevant or redundant attributes can slow down the mining process.

Dimensionality reduction reduces the data set size by removing such attributes (or dimensions) from it. Typically, methods of attribute subset selection are applied. The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Mining on a reduced set of attributes has an additional benefit. It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.

"How can we find a 'good' subset of the original attributes?" For *d attributes*, there are 2^d possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as *d* and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically *greedy* in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.

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

Basic heuristic methods of attribute subset selection include the following techniques:

1. **Stepwise 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.
2. **Stepwise backward elimination:** The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
3. **Combination of forward selection and backward elimination:** The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.

The stopping criteria for methods 1 to 3 may vary. The procedure may employ a threshold on the measure used to determine when to stop the attribute selection process.

Forward selection	Backward elimination	Decision tree induction
Initial attribute set : {A1, A2, A3, A4, A5, A6}	Initial attribute set : {A1, A2, A3, A4, A5, A6}	Initial attribute set : {A1, A2, A3, A4, A5, A6}
Intial reduced set : { }	→ {A1, A3, A4, A5, A6}	
→ {A1}	→ {A1, A4, A5, A6}	
→ {A1, A4}	→ Reduced attribute set : {A1, A4, A6}	
→ Reduced attribute set : {A1, A4, A6}		

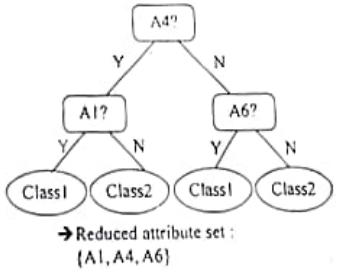


Figure 1.16 Greedy methods for attribute subset selection.

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

When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.

If the mining task is classification, and the mining algorithm itself is used to determine the attribute subset, then this is called a wrapper approach; otherwise, it is a **filter approach**. In general, the **wrapper approach** leads to greater accuracy since it optimizes the evaluation measure of the algorithm while removing attributes. However, it requires much more computation than a filter approach.

1.11.3 Data Compression

In *data compression*, data encoding or transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be *reconstructed* from the compressed data without any loss of information, the data compression technique used is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data compression technique is called lossy. There are several well-tuned algorithms for string compression. Although they are typically lossless, they allow only limited manipulation of the data. We instead focus on two popular and effective methods of lossy data compression: *wavelet transforms* and *principal components analysis*.

Wavelet Transforms

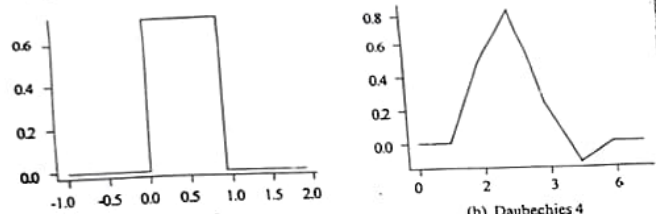
The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector D, transforms it to a numerically different vector, D', of wavelet coefficients. The two vectors are of the same length.

"How can this technique be useful for data reduction if the wavelet transformed data are of the same length as the original data?" The use-fulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients. For example, all wavelet coefficients larger than some user-specified threshold can be retained. The remaining coefficients are set to 0. The resulting data representation is therefore very sparse, so that operations that can take advantage of data sparsity are computationally very fast if performed in wavelet space. The technique also works to remove noise without smoothing out the main features of the data, making it effective for data cleaning as well. Given a set of coefficients, an approximation of the original data can be constructed by applying the *inverse* of the DWT used.

The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines. In general, however, the DWT achieves better lossy compression. That is, if the same number of coefficients is retained for a DWT and a DFT of a

Walchand College LIBRARY
48548 / USH / JYO

data vector, the DWT version will provide a more accurate approximation of the original data. Hence for an equivalent approximation, the DWT requires less space than the DFT. Unlike the DFT,



(a) Haar 2 (b) Daubechies 4

Figure 1.17 Examples of wavelet families. The number next to a wavelet name is the number of vanishing moments of the wavelet. This is a set of mathematical relationships that the coefficients must satisfy and is related to the number of coefficients.

Wavelets are quite localized in space, contributing to the conservation of local detail. There is only one DFT, yet there are several families of DWTs. Figure 1.17 shows some wavelet families. Popular wavelet transforms include the Haar_2, Daubechies_4, and Daubechies_6 transforms. The general procedure for applying a discrete wavelet transform uses a hierarchical *pyramid algorithm* that halves the data at each iteration, resulting in fast computational speed. The method is as follows:

1. The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros, as necessary.
2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.
3. The two functions are applied to pairs of the input data, resulting in two sets of data of length $L/2$. In general, these represent a smoothed or low frequency version of the input data, and the high frequency content of it, respectively.
4. The two functions are recursively applied to the sets of data obtained in the previous loop, until the resulting data sets obtained are of length 2.
5. A selection of values from the data sets obtained in the above iterations are designated the wavelet coefficients of the transformed data.

Equivalently, a matrix multiplication can be applied to the input data in order to obtain the wavelet coefficients, where the matrix used depends on the given tors and are mutually orthogonal, so that the matrix inverse is just its transpose. Although we do not have room to discuss it here, the

property allows the re-construction of the data from the smooth and smooth-difference data sets. By factoring the matrix used in to a product of a few sparse matrices, the resulting *"fast DWT"* algorithm has a complexity of $O(n)$ for an input vector of length n.

Wavelet transforms can be applied to multidimensional data, such as a data cube. This is done by first applying the transform to the first dimension, then to the second, and so on. The computational complexity involved is linear with respect to the number of cells in the cube. Wavelet transforms give good results on sparse or skewed data and on data with ordered attributes. Lossy compression by wavelets is reportedly better than JPEG compression, the current commercial standard. Wavelet transforms have many real-world applications, including the compression of fingerprint images, computer vision, analysis of time-series data, and data cleaning.

Principal Components Analysis

In this subsection we provide an intuitive introduction to principal components analysis as a method of data compression.

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

PCA can be used as a form of dimensionality reduction. However, unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

The basic procedure is as follows

1. 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.
2. PCA computes c orthonormal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the *principal components*. The input data are a linear combination of the principal components.
3. The principal components are sorted in order of decreasing "significance" or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, Figure 1.18 shows the first two principal components, $Y1$ and $Y2$, for the given set of data originally mapped to the axes $X1$ and $X2$. This information helps identify groups or patterns within the data.

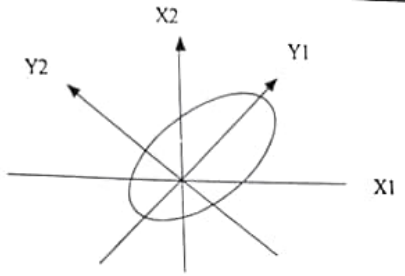


Figure 1.18 Principal components analysis. Y1 and Y2 are the first two Principal components for the given data

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

PCA is computationally inexpensive, can be applied to ordered and unordered attributes, and can handle sparse data and skewed data. Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions. For example, a 3-D data cube for sales with the dimensions *item_type, branch,* and *year* must first be reduced to a 2- D cube, such as with the dimensions *item_type* and *branch* x *year*. In comparison with wavelet transforms for data compression, PCA tends to be better at handling sparse data, while wavelet transforms are more suitable for data of high dimensionality.

1.11.4 Numerosity Reduction

"Can we reduce the data volume by choosing alternative, 'smaller' forms of data representation?" Techniques of *numerosity reduction* can indeed be applied for this purpose. These techniques may be parametric or nonparametric. For *para-metric methods,* a model is used to estimate the data, so that typically only the data parameters need be stored, instead of the actual data. (Outliers may also be stored.) Log-linear models, which estimate discrete multidimensional probability distributions, are an example. *Nonparametric methods* for storing reduced representations of the data include histograms, clustering, and sampling.

Let's have a look at each of the numerosity reduction techniques mentioned above.

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

$$Y = \alpha + \beta X, \qquad (1.5)$$

where the variance of *Y* is assumed to be constant. The coefficients α and β (called *regression coefficients)* specify the *Y*-intercept and slope of the line, respectively. 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. This allows higher-order data cubes to be constructed from lower-order ones. Log-linear models are therefore also useful for data compression (since the smaller-order cuboids together typically occupy less space than the base cuboid) and data smoothing (since cell estimates in the smaller-order cuboids are less subject to sampling variations than cell estimates in the base cuboid).

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

Histograms

Histograms use binning to approximate data distributions and are a popular form of data reduction. A histogram for an attribute *A* partitions the data distribution of *A* into disjoint subsets, or *buckets.* The buckets are displayed on a horizontal axis, while the height (and area) of a bucket typically reflects the average frequency of the values represented by the bucket. If each bucket represents only a single attribute-value/frequency pair, the buckets are called *singleton buckets.* Often, buckets instead represent continuous ranges for the given attribute.

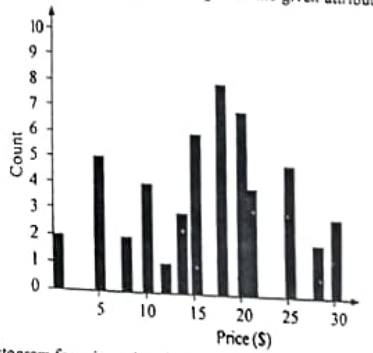


Figure 1.19 A Histogram for price using singleton buckets-each bucket represents one price-value/ frequency pair.

Example 1.7
The following data are a list of prices of commonly sold items at *AllElectronics* (rounded to the nearest dollar). The numbers have been sorted: 1, 1,5,5,5,5,5,8,8,10,10,10,10,12,14,14,14,15,15,15,15, 15,15,18,18,18,18,18,18,18,18,20,20,20,20,20,20,20,21,21,21,21,25,25,25,25,25,28,28,30,30,30.

Figure 1.19 shows a histogram for the data using singleton buckets. To further reduce the data, it is common to have each bucket denote a continuous range of values for the given attribute. In Figure 1.20, each bucket represents a different $10 range for *price*.

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

- **Equiwidth:** In an equiwidth histogram, the width of each bucket range is uniform (such as the width of $10 for the buckets in Figure 3.10).
- **Equidepth** (or equiheight): In an equidepth histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (that is, each bucketcontairs 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 MaxDiffhistogram, we consider the difference between each pair of adjacent values. A bucket boundary is established between each pair for pairs having the $\beta - 1$ largest differences, where β is user-specified.

V-Optimal and MaxDiffhistograms tend to be the most accurate and practical. Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data. The histograms described above for single attributes can be extended for multiple attributes. *Multidimensional histograms* can capture dependencies between attributes. Such histograms have been found effective in approximating data with up to five attributes. More studies are needed regarding the effectiveness of multidimensional histograms for very high dimensions. Singleton buckets are useful for storing outliers with high frequency.

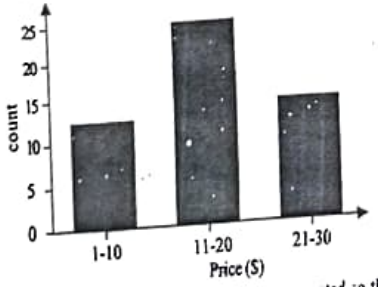


Figure 1.20 An equiwidth histogram for price, a values are aggregated so that each bucket has a uniform width of $10

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

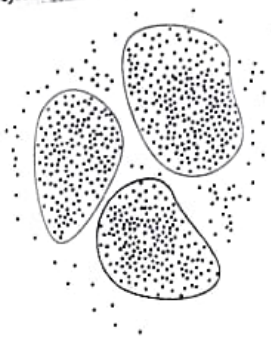


Figure 1.21 A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster centroid is marked with "+".

of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the "average object," or average point in space for the cluster). Figure 1.21 shows a 2-D plot of customer data with respect to customer locations in a city, where the centroid of each cluster is shown with a "+". Three data clusters are visible.

In data reduction, the cluster representations of the data are used to replace the actual data. The effectiveness of this technique depends on the nature of the data. It is much more effective for data that can be organized into distinct clusters than for smeared data.

In database systems, multidimensional index trees are primarily used for providing fast data access. They can also be used for hierarchical data reduction, providing a multiresolution clustering of the data. This can be used to provide approximate answers to queries. An index tree recursively partitions the multidi-mensional space for a given set of data objects, with the root node representing the entire space. Such trees are typically balanced, consisting of internal and leaf nodes. Each parent node contains keys and pointers to child nodes that, collectively, represent the space represented by the parent node. Each leaf node contains pointers to the data tuples they represent (or to the actual tuples).

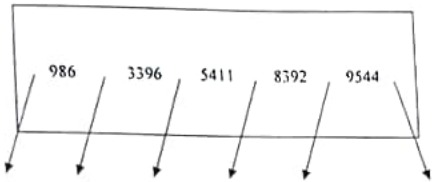


Figure 1.22 The root of a B+ - tree for a given set of data.

An index tree can therefore store aggregate and detail data at varying levels of resolution or abstraction. It provides a hierarchy of clusterings of the data set, where each cluster has a label that holds for the data contained in the cluster. If we consider each child of a parent node as a bucket, then an index tree can be considered as a *hierarchical histogram*. For example, consider the root of a B+ -tree as shown in Figure 1.22, with pointers to the data keys 986,3396,5411,8392, and 9544. Suppose that the tree contains 10,000 tuples with keys ranging from 1 to 9999.

The data in the tree can be approximated by an equidepth histogram of six buckets for the key ranges 1 to 985, 986 to 3395, 3396 to 5410, 5411 to 8391, 8392 to 9543, and 9544 to 9999. Each bucket contains roughly 10,000/6 items. Similarly, each bucket is subdivided into smaller buckets, allowing for aggregate data at a finer-detailed level. The use of multidimensional index trees as a form of data reduction relies on an ordering of the attribute values in each dimension. Multidimensional index trees include R-trees, quad-trees; and their variations. They are well suited for handling both sparse and skewed data.

Sampling

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

- **Simple random sample without replacement (SRSWOR) of size *n*:** This is created by drawing *n* of the N tuples from D ($n < N$), where the probability of drawing any tuple in D is 1/ N, that is, all tuples are equally likely.
- **Simple random sample with replacement (SRSWR) of size *n*:** This is similar to SRSWOR, except that each time a tuple is drawn from D, it is recorded and then *replaced*. That is; after a tuple is drawn, it is placed back in D so that it may be drawn again.
- **Cluster sample:** If the tuples in D are grouped into M mutually disjoint "clusters" then an SRS of *m* clusters can be obtained, where $m < M$. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples.
- **Stratified sample:** If *D* is divided into mutually disjoint parts called *strata*, a stratified sample of D is generated by obtaining an SRS at each stratum. This helps to ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

These samples are illustrated in Figure 1.23. They represent the most commonly used forms of sampling for data reduction.

An advantage of sampling for data reduction is that the cost of obtaining a sample *is proportional to the size of the sample, n,* as opposed to N, the data set size. Hence, sampling complexity is potentially *sublinear* to the size of the data. Other data reduction techniques can require at least one complete pass through D. For a fixed sample size, sampling complexity increases only linearly as the number of data dimensions, *d*, increases, while techniques using histograms, for example, increase exponentially in *d*.

When applied to data reduction, sampling is most commonly used to estimate the answer to an aggregate query. It is possible (using the central limit theorem) to, determine a sufficient sample size for estimating a given function within a specified degree of error. This sample size *n*, may be extremely small in comparison to N. Sampling is a natural choice for the progressive refinement of a reduced data set. Such a set can be further refined by simply increasing the sample size.

1.12 Discretization and Concept Hierarchy Generation

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. Reducing the number of values for an attribute is especially beneficial if decision-tree-based methods of classification mining are to be applied to the preprocessed data. These methods are typically recursive, where a large amount of time is spent on sorting the data at each step. Hence, the smaller the number of distinct values to sort, the faster these methods should be many discretization techniques can be applied recursively in order to provide a hierarchical or multiresolution partitioning of the attribute values, known as a concept hierarchy. Concept hierarchies are useful for mining at multiple levels of abstraction.

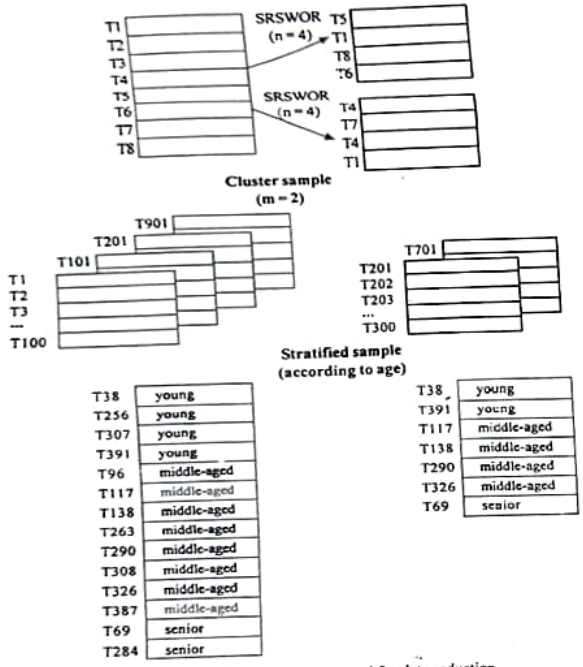


Figure 1.23 Sampling can be used for data reduction

A concept hierarchy for a given numeric attribute defines a discretization of the attribute. Concept hierarchies can be used to reduce the data by collecting and replacing low-level concepts (such as numeric values for the attribute *age)* by higher-level concepts (such as *young, middle-aged,* or *senior).* Although detail is lost by such data generalization, the generalized data may be more meaningful and easier to interpret, and will require less space than the original data. Mining on a reduced data set will require fewer input/output operations and be more efficient than mining on a larger, ungeneralized data set. An example of a concept hierarchy for the attribute *price* is given in Figure 1.24. More than one concept hierarchy can be defined for the same attribute in order to accommodate the needs of the various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for the user or domain expert. Fortunately, many hierarchies are implicit within the database schema and can be defined at the schema definition level. Concept hierarchies often can be automatically generated or dynamically refined based on statistical analysis of the data distribution. Let's look at the generation of concept hierarchies for numeric and categorical data.

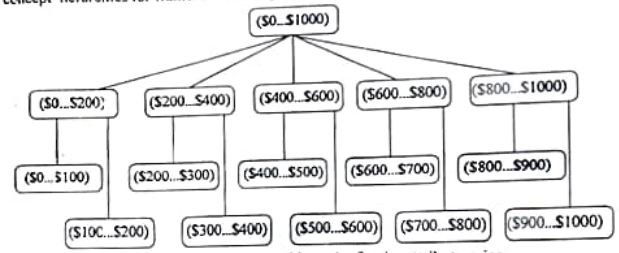


Figure 1.24 A Concept hierarchy for the attribute price

1.12.1 Discretization and Concept Hierarchy Generation for Numeric Data

It is difficult and laborious to specify concept hierarchies for numeric attributes due to the wide diversity of possible data ranges and the frequent updates of data values. Such manual specification can also be quite arbitrary.

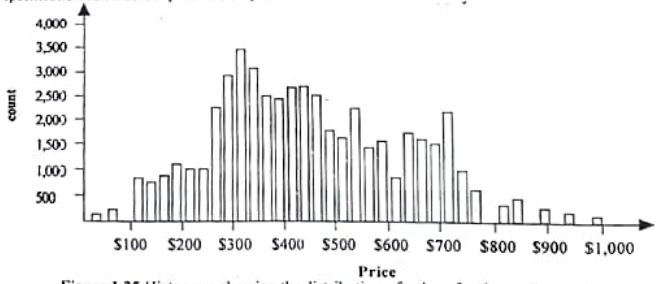


Figure 1.25 Histogram showing the distribution of values for the attribute price

Concept hierarchies for numeric attributes can be constructed automatically based on data distribution analysis. We examine five methods for numeric concept hierarchy generation: *binning, histogram analysis, cluster analysis, entropy-based discretization,* and *data segmentation by "natural partitioning."*

Binning

Section 1.9.2 discussed binning methods for data smoothing. These methods are also forms of discretization. For example, attribute values can be discretized by distributing the values into bins, and replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions in order to generate concept hierarchies.

Histogram Analysis

Histograms, as discussed in Section 1.11.4, can also be used for discretization. Figure 1.25 presents a histogram showing the data distribution of the attribute *price* for a given data set. For example, the most frequent price range is roughly $300-$325. Partitioning rules can be used to define the ranges of values. For instance, in an *equiwidth* histogram, the values are partitioned into equal-sized partions or ranges (e.g., ($0 . . . $100], ($100 . . . $200],....., ($900 . . . $1000]). With an *equidepth* histogram, the values are partitioned so that, ideally, each partition contains the same number of data samples. The histogram analysis algorithm can be applied recursively to each partition in order to automatically generate a multilevel concept hierarchy, with the procedure terminating once a prespecified number of concept levels has been reached. A *minimum interval size* can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level.

Cluster Analysis

A clustering algorithm can be applied to partition data into clusters or groups. Each cluster forms a node of a concept hierarchy, where all nodes are at the same conceptual level. Each cluster may be further decomposed into several subclusters, forming a lower level of the hierarchy. Clusters may also be grouped together in order to form a higher conceptual level of the hierarchy.

Entropy-Based Discretization

An information-based measure called *entropy* can be used to recursively partition the values of a numeric attribute A, resulting in a hierarchical discretization. Such a discretization forms a numerical concept hierarchy for the attribute. Given a set of data tuples S, the basic method for entropy-based discretization of A is as follows:

1. Each value of A can be considered a potential interval boundary or threshold T. For example, a value v of A can partition the samples in S into two subsets satisfying the conditions $A < v$ and $A \geq V$, respectively, thereby creating a binary discretization.

2. Given S, the threshold value selected is the one that maximizes the information gain resulting from the subsequent partitioning. The information gain is

$$I(S,T) = |S_1|/|S| \, Ent(S_1) + |S_2|/|S| \, Ent(S_2). \qquad (1.6)$$

where S_1 and S_2 correspond to the samples in S satisfying the conditions $A < T$ and $A \geq T$, respectively. The entropy function *Ent* for a given set is calculated based on the class distribution of the samples in the set.

For example, given m classes, the entropy of S_1 is

$$Ent(S_1) = -\sum_{i=1}^{m} Pi \log_2(Pi), \qquad (1.7)$$

Where Pi is the probability of class i in S1 determined by dividing the number of samples of class i in S_1 by the total number of samples in S1. The value of $Ent(S_2)$ can be computed similarly.

3. The process of determining a threshold value is recursively applied to each Partition obtained, until some stopping criterion is met, such as

$$Ent(S) - I(S,T) > \delta. \qquad (1.8)$$

Entropy-based discretization can reduce data size. Unlike the other methods mentioned here so far, entropy-based discretization uses class information. This makes it more likely that the interval boundaries are defined to occur in places that may help improve classification accuracy. The information gain and entropy measures described here are also used for decision tree induction.

Segmentation by Natural Partitioning

Although binning, histogram analysis, clustering, and entropy-based discretization are useful in the generation of numerical hierarchies, many users would like to see numerical ranges partitioned into relatively uniform, easy-to-read intervals that appear intuitive or "natural." For example, annual salaries broken into ranges like ($50,000, $60,000) are often more desirable than ranges like [$51,263.98, $60,872.34], obtained by some sophisticated clustering analysis.

The **3-4-5 rule** can be used to segment numeric data into relatively uniform, "natural" intervals. In general, the rule partitions a given range of data into 3, 4, or 5 relatively equiwidth intervals, recursively and level by level, based on the value range at the most significant digit. The rule is as follows:

- If an interval covers 3, 6, 7, or 9 distinct values at the most significant digit, then partition the range into 3 intervals (3 equiwidth intervals for 3; 6, 9, and 3 intervals in the grouping of 2-3-2 for 7).
- If it covers 2, 4, or 8 distinct values at the most significant digit, then partition the range into 4 equiwidth intervals.
- If it covers 1, 5, or 10 distinct values at the most significant digit, then partition the range into 5 equiwidth intervals.

Malaviya College of E...

The rule can be recursively applied to each interval, creating a concept hierarchy for the given numeric attribute. Since there could be some dramatically large positive or negative values in a data set, the top-level segmentation, based merely on the minimum and maximum values, may derive distorted results. For example, the assets of a few people could be several orders of magnitude higher than those of others in a data set. Segmentation based on the maximal asset values may lead to a highly biased hierarchy. Thus the top-level segmentation can be performed based on the range of data values representing the majority (e.g., 5th percentile to 95th Percentile) of the given data. The extremely high or low values beyond the top level segmentation will form distinct, interval(s) that can be handled separately, but in a similar manner.

The following example illustrates the use of the 3-4-5 rule for the automatic construction of a numeric hierarchy.

Example 1.8

Suppose that profits at different branches of *AllElectronics* for the year 1999 cover a wide range, from -$351,976.00 to $4,700,896.50. A user wishes to have a concept hierarchy for *profit* automatically generated. For improved readability, we use the notation *(i...r]* to represent the interval *(i, r]*. For example, (-$1,000,000...$0] denotes the range from -$1,000,000 (exclusive) to $0 (inclusive).

Suppose that the data within the 5th percentile and 95th percentile are between -$159,876 and $1,838,761. The results of applying the 3-4-5 rule are shown in Figure 1.26.

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

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

3. Since this interval ranges over three distinct values at the most significant digit, that is, (2,000,000 - (-1,000,000))/1,000,000 = 3, the segment is partitioned into three equiwidth subsegments according to the 3-4-5 rule: (-$1,000,000 ... $0], ($0 ... $1,000,000], and ($1,000,000 ... $2,000,000]. This represents the top tier of the hierarchy.

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

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

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

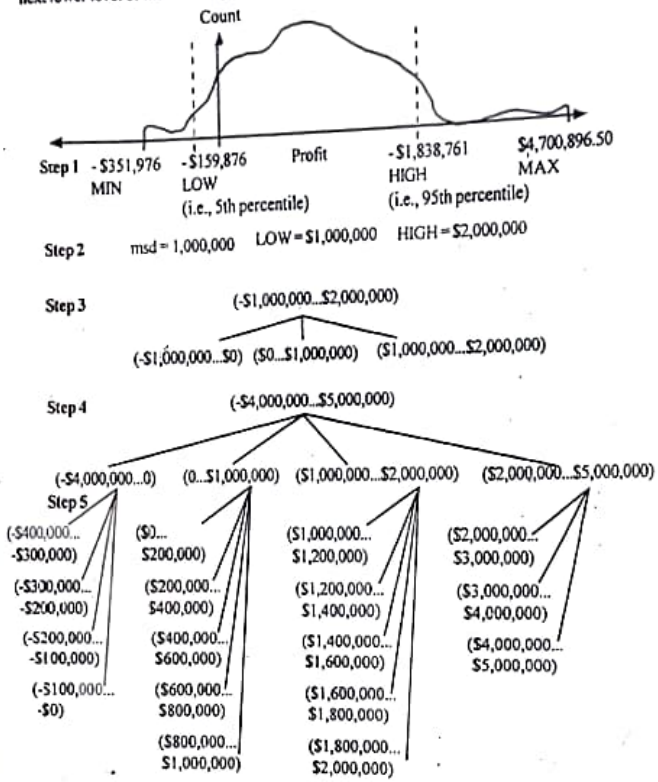


Figure 1.26 Automatic generation of a concept hierarchy for a profit based on the 3-4-5 rule

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

Note that this heuristic rule cannot be pushed to the extreme since there are obvious cases that do not follow such a heuristic. For example, a time dimension in a database may contain 20 distinct years, 12 distinct months, and 7 distinct days of the week. However, this does not suggest that the time hierarchy should be *"year < month < days_of_the_week"*, *with days_of_the_week* at the top of the hierarchy.

Specification of only a partial set of attributes: Sometimes a user can be sloppy when defining a hierarchy, or may have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in a hierarchy specification. For example,

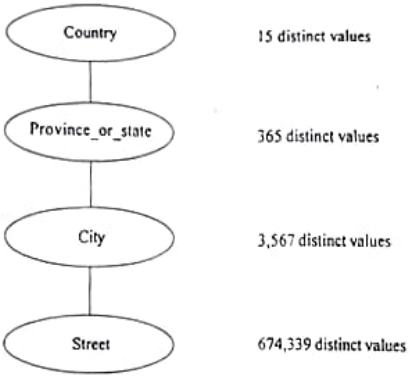


Figure 1.27 Automatic generation of a schema concept hierarchy based on the number of distinct attribute values.

instead of including all the hierarchically relevant attributes for *location*, the user may have specified only *street* and *city*. To handle such partially specified hierarchies, it is important to embed data semantics in the database schema so that attributes with tight semantic connections can be pinned together. In this way, the specification of one attribute may trigger a whole group of semantically tightly linked attributes to be "dragged in" to form a complete hierarchy. Users, however, should have the option to override this feature, as necessary.

Example 1.10

Suppose that a database system has pinned together the five attributes *number*, *street*, *city*, *province_or _state*, and *country* because they are closely linked semantically, regarding the notion of *location*. If a user were to specify only the attribute *city* for a hierarchy defining *location*, the system may automatically drag in all of the above five semantically related attributes to form a hierarchy. The user may choose to drop any of these attributes, such as *number* and *street*, from the hierarchy, keeping *city* as the lowest conceptual level in the hierarchy.

Summary

Data preprocessing is an important issue for both data warehousing and data mining, as real-world data tend to be incomplete, noisy, and inconsistent. Data preprocessing includes data cleaning, data integration, data transformation, and data reduction.

Data cleaning routines can be used to fill in missing values, smooth noisy data, identify outliers, and correct data inconsistencies.

Data integration combines data from multiple sources to form a coherent data store. Metadata, correlation analysis, data conflict detection, and the resolution of semantic heterogeneity contribute towards smooth data integration.

Data transformation routines convert the data into appropriate forms for mining. For example, attribute data may be normalized so as to fall between a small range, such as 0.0 to 1.0.

Data reduction techniques such as data cube aggregation, dimension reduc-tion, data compression, numerosity reduction, and discretization can be used to obtain a reduced representation of the data, while minimizing the loss of information content.

Automatic generation of concept hierarchies for numeric data can involve techniques such as binning, histogram analysis, cluster analysis, entropy-based discretization, and segmentation by natural partitioning. For categoric data, concept hierarchies may be generated based on the number of distinct values of the attributes defining the hierarchy.

Although several methods of data preprocessing have been developed, data preparation remains an active area of research.

Exercises

1. Data quality can be assessed in terms of accuracy, completeness, and consistency. Propose two other dimensions of data quality.
2. In real-world data, tuples with missing values for some attributes are a common occurrence. Describe various methods for handling this problem.
3. Suppose that the data for analysis include the attribute *age*. The *age* values for the data tuples are (in increasing order): 13, IS, 16, 16, 19, 20,20,21,22,22,25,25, 25,25,30,33,33,35,35,35,35,36,40,45,46,52,70.
   (a) Use smoothing by bin means to smooth the above data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.
   (b) How might you determine outliers in the data?
   (c) What other methods are there for data smoothing?
4. Discuss issues to consider during data integration.

5. Using the data for *age* given in Exercise 3, answer the following:
   (a) Use min-max normalization to transform the value 35 for *age* onto the range [0.0,1.0].
   (b) Use z-score normalization to transform the value 35 for *age*, where the standard deviation of *age* is 12.94 years.
   (c) Use normalization by decimal scaling to transform the value 35 for *age*.
   (d) Comment on which method you would prefer to use for the given data, giving reasons as to why.

6. Use a flow chart to summarize the following procedures for attribute subset selection:
   (a) Stepwise forward selection
   (b) Stepwise backward elimination
   (c) A combination of forward selection and backward elimination

7. Using the data for *age* given in Exercise 3,
   (a) Plot an equiwidth histogram of width 10.
   (b) Sketch examples of each of the following sampling techniques: SRSWOR, SRSWR, cluster sampling, stratified sampling. Use samples of size 5 and the strata "young", "middle-aged", and "senior".

8. Propose an algorithm, in pseudo code or in your favorite programming language, for the following:
   (a) The automatic generation of a concept hierarchy for categorical data based on the number of distinct values of attributes in the given schema
   (b) The automatic generation of a concept hierarchy for numeric data based on the *equiwidth* partitioning rule
   (c) The automatic generation of a concept hierarchy for numeric data based on the *equidepth* partitioning rule

Objective

1. **Which of the following performs a linear transformation on the original data?**
   a. Z-score normalization
   b. Normalization with decimal scaling
   c. Zero-standard deviation
   d. **Min-max normalization**

2. **Which of the following is the best method for missing values in data cleaning?**
   a. Fill in the missing value manually
   b. **Use the most probable value to fill in the missing value**
   c. Use the attribute mean to fill the missing value
   d. Use a global constant to fill in the missing value

3. **The minimum and maximum values in a given bin are identified as the**
   a. Bin means
   b. Bin average
   c. Bin medians
   d. **Bin boundaries**

4. **Which of the following is data transformation operation?**
   a. **Normalization**
   b. Regression
   c. Clustering
   d. Binning

5. **_____methods smooth a sorted data value by consulting in neighborhood i.e. the values around it.**
   a. Clustering
   b. **Binning**
   c. Regression
   d. Data reduction

6. **Z-score normalization is also called as**
   a. Min-max normalization
   b. Zero-standard deviation normalization
   c. **Zero-mean normalization**
   d. Normalization by decimal scaling

7. **_____is a random error or variance in a measured variable.**
   a. Bin
   b. Cluster
   c. Noise
   d. Regression

8. **The data are consolidated into forms appropriate for mining is called as**
   a. Data reduction
   b. Data Redundancy
   c. Data clean
   d. Data transformation

9. **Which of the following is a decision tree algorithm?**
   a. C32
   b. ID3
   c. PP2
   d. DIM

10. If the tuples in D are grouped into M mutually disjoint Clustering, then an simple random sample of m clusters can be obtained, where m M which of the following suits the above sentence?
    a. Stratified sample
    b. SRS without replacement
    c. **Cluster sample**
    d. SRS with replacement

11. Multidimensional index trees include
    a. A- trees
    b. T-trees
    c. P-trees
    d. **R-trees**

12. Which of the following strategy for data reduction is irrelevant, weakly relevant, or redundant attributes may be detected and removed?
    a. Data cube aggregation
    b. **Dimension reduction**
    c. Data compression
    d. Numerosity reduction

13. In database systems, _____ are primarily used for providing fast data access.
    a. Red-black trees
    b. Game trees
    c. **Multidimensional index trees**
    d. splay trees

14. If the mining task is classification, and the mining algorithm itself is used to determine the attribute subset, then this is called a _____ approach.
    a. Filter
    b. Reduction
    c. Smoothing
    d. **Wrapper**

15. The discrete wavelet transformation is closely related to the _____ transform.
    a. **Discrete fourier**
    b. Fourier
    c. Laplace
    d. wavelet

16. Principal components analysis is also called as
    a. **Karhunen-loeve method**
    b. Kinen-liva method
    c. Kruskal-leam method
    d. Kutni-lafa method

17. _____ can be used as a data reduction technique since it allows a large data set to be represented by a much smaller random subset of the data.
    a. Clustering
    b. Regression
    c. Histograms
    d. **Sampling**

18. Loy-linear models are
    a. **Parametric methods**
    b. Discrete methods
    c. Non-parametric methods
    d. Non- discrete methods

19. Which of the following method is the generation of concept of hierarchies for categorical data?
    a. Specification of a portion of a hierarchy by implicit data grouping
    b. Specification of their partial ordering, but not of a set of attributes
    c. **Specification of a set of attributes, but not of their partial order**
    d. Specification of only a partial set of entities

20. Which of the following method uses class information?
    a. Histogram analysis
    b. Binning
    c. Cluster analysis
    d. **Entropy-based Discretization**

21. _____ hierarchies for categorical attributes or dimensions typically involve a group of attributes
    a. Discretization
    b. Semantic
    c. Index
    d. **Concept**

22. Which of the following is based on the maximal asset values, which may lead to a highly biased hierarchy?
    a. Cluster analysis
    b. **Segmentation**
    c. Binning
    d. Histogram analysis

23. The _____ can be used to segment numeric data into relatively uniform, "natural" intervals.
    a. 1-2-3 rule
    b. 2-3-4 rule
    c. **3-4-5 rule**
    d. 4-5-6rule

Jharkhand College of

24. **Which of the following support the bitmap indices**
    a. **Sybase IQ**
    b. Oracle 7
    c. COBOL
    d. SQL

25. **_____ are created for the data names and definitions of the given warehouse**
    a. Data cube
    b. Summarized data
    c. **Meta data**
    d. Detailed Information

26. **Chunking technique involves "overlapping" some of the aggregation computations, it is referred to as _____ aggregation in data cube computation**
    a. Two way array
    b. Three way array
    c. **Multi way array**
    d. Sparse array

27. **The _____ operator computes aggregates over all subsets of the dimensions specified in the operation.**
    a. Data base
    b. **Computer cube**
    c. Define cube
    d. Group by

28. **Which of the following is a subcube that is small enough to fit into the memory available for cube computation?**
    a. Bulk
    b. Array
    c. Structure
    d. **Chunk**

29. **The bit mapped join indices method is an integrated form of**
    a. Composite join indexing and bitmap indexing
    b. Join indexing and composite join indexing
    c. **Join indexing and bitmap indexing**
    d. Bitmap indexing and outer join indexing

30. **Which of the following sets a good example for interactive data analysis and provides the necessary preparations for exploratory data mining?**
    a. OLP
    b. **OLAP**
    c. OLTP
    d. OLDP

31. **A _____ algorithm can be applied to partition data into groups**
    a. Binning
    b. Histogram
    c. **Clustering**
    d. Entropy-based

References

D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Communications of ACM, 42:73-78, 1999.

Jagadish et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), December 1997.

D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999.

T. Redman. Data Quality: Management and Technology. Bantam Books, New York, 1992.

Y. Wand and R. Wang. Anchoring data quality dimensions ontological foundations. Communications of ACM, 39:86-95, 1996.

R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995.

Han and Fu, Concept hierarchies and their automatic generation from categorical data [HF94].

Jam8S, Dob90, JW92, Dev9S, NKNW96,An introduction to regression and log-linear models

Pearl [Pea88], For log-linear mod-els (known as *multiplicative models* in the computer science literature).

ChiMerge by Kerber [Kern] and Chi2 by Liu and Setiono [L59S] are meth-ods for the automatic discret zation of numeric attributes that both employ the X 2 statistic.

Muralikrishna and DeWitt [MD88] and Poosala and Ioannidis [PI97], For extensions of single attribute histograms to multi-ple attributes.

Siedlecki and Sklansky, A combination forward selection and backward elimination method [5588].

Fayyad and Irani [FI93] apply the minimum description length prin-ciple to determine the number of intervals for numeric discretization.