MODULE-2
NOTES

Data warehouse implementation& Data mining: Efficient Data Cube computation: An overview, Indexing OLAP Data: Bitmap index and join index, Efficient processing of OLAP Queries, OLAP server Architecture ROLAP versus MOLAP Versus HOLAP. : Introduction: What is data mining, Challenges, Data Mining Tasks, Data: Types of Data, Data Quality, Data Preprocessing, Measures of Similarity and Dissimilarity

---

2. Data Warehouse Implementation:

- Data warehouses contain huge volumes of data.
- OLAP servers demand that decision support queries be answered in the order of seconds.
- Therefore, it is crucial for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques.

2.1 Efficient Data Cube Computation: An Overview

- At the core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions.
- In SQL terms, these aggregations are referred to as group-by's.
- Each group-by can be represented by a cuboid, where the set of group- by's forms a lattice of cuboids defining a data cube.
- In this section, we explore issues relating to the efficient computation of data cubes.

2.1.1The compute cube Operator and the Curse of Dimensionality

- One approach to cube computation extends SQL so as to include a compute cube operator.
- The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation.
- This can require excessive storage space, especially for large numbers of dimensions.

- **Example 2.1.** A data cube is a lattice of cuboids. Suppose thatyou would like to create a data cube for AllElectronics sales that contains the following: city, item, year, and sales in dollars. You would like to be able to analyze the data, with queries such as the following:

  "Compute the sum of sales, grouping by city and item."
  "Compute the sum of sales, grouping by city."
  "Compute the sum of sales, grouping by item."
  What is the total number of cuboids, or group-by's, that can be computed for this data cube?

- Taking the three attributes, city, item, and year, as the dimensions for the data cube, and sales in dollars as the measure, the total number of cuboids, or group-by's, that can be computed for this data cube is $2^3 = 8$.

- The possible group-by's are the following: {(city, item, year), (city, item),(city, year),(item, year),(city),(item),(year),()}, where() means that the group-by is empty(i.e., the dimensions are not grouped).
- These group-by's form a lattice of cuboids for the data cube, as shown in Figure 2.1. The base cuboid contains all three dimensions, city, item, and year. It can return the total sales for any combination of the three dimensions.

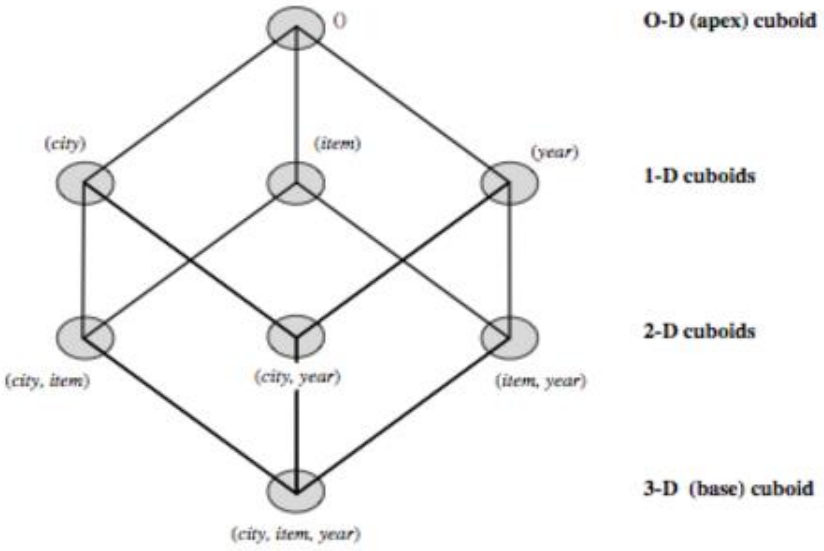


Figure 2.1: Lattice of cuboids, making up a 3-D data cube. Each cuboid represents a different group-by. The base cuboid contains the three dimensions city, item, and year.

- The apex cuboid, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales.
- The base cuboid is the least generalized (most specific)of the cuboids. The apex cuboid is the most generalized(least specific) of the cuboids, and is often denoted as all.
- If we start at the apex cuboid and explore downward in the lattice, this is equivalent to drilling down within the data cube. If we start at the base cuboid and explore upward, this is akin to rolling up.
- An SQL query containing no group-by, such as "compute the sum of total sales," is a zero-dimensional operation. An SQL query containing one group-by, such as "compute the sum of sales, group by city," is a one-dimensional operation.
- A cube operator on n dimensions is equivalent to a collection of group by statements, one for each subset of the n dimensions. Therefore, the cube operator is the n-dimensional generalization of the group by operator.
- Similar to the syntax of SQL, the data cube in Example 2.1 could be defined as define cube sales cube[city,item,year]: sum(salesin dollars) For a cube with n dimensions, there are a total of 2n cuboids, including the base cuboid.

- A statement such as compute cube sales cube would explicitly instruct the system to compute the sales aggregate cuboids for all of the eight subsets of the set {city, item, year}, including the empty subset.
- Online analytical processing may need to access different cuboids for different queries. Therefore, it may seem like a good idea to compute all or at least some of the cuboids in a data cube in advance.
- Precomputation leads to fast response time and avoids some redundant computation. Most, if not all, OLAP products resort to some degree of precomputation of multidimensional aggregates.
- A major challenge related to this precomputation, however, is that the required storage space may explode if all of the cuboids in a data cube are precomputed, especially when the cube has many dimensions.
- The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels. This problem is referred to as the curse of dimensionality. The extent of the curse of dimensionality is illustrated below.
- "How many cuboids are there in an n-dimensional data cube?" If there were no hierarchies associated with each dimension, then the total number of cuboids for an n-dimensional datacube, as we have seen above,is 2^n.
- However, in practice, many dimensions do have hierarchies. For example, the dimension time is usually not explored at only one conceptual level, such as year, but rather at multiple conceptual levels, such as in the hierarchy "day < month < quarter <year".
- For an n-dimensional data cube, the total number of cuboids that can be generated (including the cuboids generated by climbing up the hierarchies along each dimension) is

$$Total\ number\ of\ cuboids = \prod_{i=1}^{n}(L_i + 1),$$

where L_i is the number of levels associated with dimension i. One is added to Li in above Equation to include the virtual top level, all. (Note that generalizing to all is equivalent to the removal of the dimension.) This formula is based on the fact that, at most, one abstraction level in each dimension will appear in a cuboid. For example, the time dimension as specified above has 4 conceptual levels, or 5 if we include the virtual level all. If the cube has 10 dimensions and each dimension has 5 levels (including all), the total number of cuboids that canbegeneratedis$5^{10} \approx 9.8\times10^6$.

- The size of each cuboid also depends on the cardinality (i.e., number of distinct values)of each dimension. For example, if the All Electronics branch in each city sold every item, there would be |city|× |item| tuples in the city-item group-by alone. As the number of dimensions, number of conceptual hierarchies, or cardinality increases, the storage space required for many of the group-by's will grossly exceed the (fixed) size of the input relation.
- By now, you probably realize that it is unrealistic to precompute and materialize all of the cuboids that can possibly be generated for a data cube(i.e., from a base cuboid). If there are many cuboids, and these cuboids are large in size, a more reasonable option is partial materialization, that is, to materialize only some of the possible cuboids that can be generated.

- **Partial Materialization:** Selected Computation of Cuboids.
- **There are three choices f**or data cube materialization given a base cuboid:
- **No materialization:** Do not precompute any of the "nonbase" cuboids. This leads to computing expensive multidimensional aggregates on the fly, which can be extremely slow.
- **Full materialization:** Precompute all of the cuboids. The resulting lattice of computed cuboids is referred to as the full cube. This choice typically requires huge amounts of memory space in order to store all of the precomputed cuboids.
- **Partial materialization:** Selectively compute a proper subset of the whole set of possible cuboids. Alternatively, we may compute a subset of the cube, which contains only those cells that satisfy some user-specified criterion, such as where the tuple count of each cell is above some threshold. We will use the term subcube to refer to the latter case, where only some of the cells may be precomputed for various cuboids.Partial materialization represents an interesting trade-off between storage space and response time. **The partial materialization of cuboids or subcubes should consider three factors:**
  (1)identify the subset of cuboids or subcubes to materialize;
  (2) exploitthe materialized cuboids or subcubesduringqueryprocessing; and
  (3) efficiently update the materialized cuboids or subcubes during load and refresh.

2.1.2 Indexing OLAP Data: Bitmap Index and Join Index

To facilitate efficient data accessing, most data warehouse systems support index structures and materialized views (using cuboids). In this section, we examine how to index OLAP data by bitmap indexing and join indexing.

- The **bitmap indexing method** is popular in OLAP products because it allows quick searching in data cubes. The bitmap index is an alternative representation of the record ID(RID) list. In the bitmap index for a given attribute, there is a distinct bit vector, **Bv**, for each value v in the domain of the attribute.
- If the domain of a given attribute consists of **n** values, then **n** bits are needed for each entry in the bitmap index(i.e., there are n bit vectors). If the attribute has the value **v** for a given row in the data table, then the bit representing that value is set to 1 in the corresponding row of the bitmap index. All other bits for that row are set to 0.

Example 2.2 Bitmap indexing: In the All Electronics data warehouse, suppose the dimension item at the top level has four values(representing item types): "home entertainment," "computer," "phone," and "security." Each value(e.g., "computer")is represented by a bit vector in the bitmap index table for item. Suppose that the cube is stored as a relation table with 100,000 rows. Because the domain of item consists of four values, the bitmap index table requires four bit vectors (or lists), each with100,000bits. Figure 2.2 shows a base(data) table containing the dimensions item and city, and its mapping to bitmap index tables for each of the dimensions.

Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

Item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

City bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Note: H for "home entertainment, " C for "computer, " P for "phone, " S for "security, "V for "Vancouver, " T for "Toronto."

Figure 2.2: Indexing OLAP data using bitmap indices.

- Bitmap indexing is advantageous compared to hash and tree indices. It is especially useful for low-cardinality domains because comparison, join, and aggregation operations are then reduced to bit arithmetic, which substantially reduces the processing time. Bitmap indexing leads to significant reductions in space and I/O since a string of characters can be represented by a single bit. For higher-cardinality domains, the method can be adapted using compression techniques.
- The **join indexing** method gained popularity from its use in relational database query processing. Traditional indexing maps the value in a given column to a list of rows having that value.
- In contrast, join indexing registers the joinable rows of two relations from a relational database. For example, if two relations R(RID, A)and S(B, SID)join on the attributes A and B, then the join index record contains the pair(RID, SID), where RID and SID are record identifiers from the R and S relations, respectively. Hence, the join index records can identify joinable tuples without performing costly join operations. Join indexing is especially useful for maintaining the relationship between a foreign key2 and its matching primary keys, from the joinable relation.
- The star schema model of data warehouses makes join indexing attractive for cross-table search, because the linkage between a fact table and its corresponding dimension tables comprises the foreign key of the fact table and the primary key of the dimension table. Join indexing maintains relationships between attribute values of a dimension(e.g., within adimension table) and the corresponding rows in the fact table. Join indices may span multiple dimensions to form composite join indices. We can use join indices to identify subcubes that are of interest.

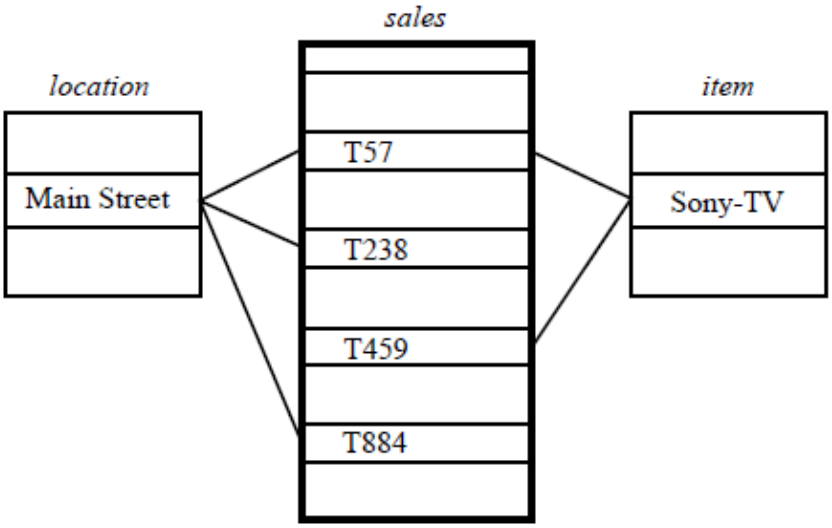


Figure 2.3: Linkages between a sales fact table and dimension tables for location and item.

Example 2.3 Join indexing. We have already seen a star schema for AllElectronics of the form "sales star [time, item, branch, location]: dollars sold = sum (sales in dollars)". An example of a join index relationship between the sales fact table and the dimension tables for location and item is shown in **Figure 2.3.**

For example, the "Main Street" value in the location dimension table joins with tuples T57, T238, and T884 of the sales fact table. Similarly, the "Sony-TV" value in the item dimensiontablejoins withtuplesT57 andT459 of the sales fact table. The corresponding join index tables are shown in Figure 2.4.

Join index table for *location/sales*

location	*sales_key*
...	...
Main Street	T57
Main Street	T238
Main Street	T884
...	...

Join index table for *item/sales*

item	*sales_key*
...	...
Sony-TV	T57
Sony-TV	T459
...	...

Join index table linking two dimensions *location/item/sales*

location	*item*	*sales_key*
...	...	...
Main Street	Sony-TV	T57
...	...	...

Join index tables based on the linkages between the *sales* fact table and dimension tables for *location* and *item* shown in Figure 3.16.

Figure 2.4: Join index tables based on the linkages between the sales fact table and dimension tables for location and item shown in Figure 2.3.

2.1.3 Efficient Processing of OLAP Queries

- The purpose of materializing cuboids and constructing OLAP index structures is to speed up query processing in data cubes. Given materialized views,query processing should proceed as follows:

1. **Determine which operations should be performed on the available cuboids:** This involves transforming any selection, projection, roll-up (group-by), and drill-down operations specified in the query into corresponding SQL and/or OLAP operations. For example, slicing and dicing adata cube may correspond to selection and/orprojection operations on a materialized cuboid.
2. **Determine to which materialized cuboid(s) the relevant operations should be applied:** This involves identifying all of the materialized cuboids that may potentially be used to answer the query, pruning the above set using knowledge of "dominance" relationships among the cuboids, estimating the costs of using the remaining materialized cuboids, and selectingthe cuboidwiththeleast cost.

- **Example 2.4 OLAP query processing.** Suppose that we define a data cube for AllElectronics of the form "sales cube [time, item, location]: sum(sales in dollars)".
  The dimension hierarchies used are "day < month < quarter < year" for time, "item name < brand < type" for item, and "street < city < province or state < country" for location.
  Suppose that the query to be processed is on {brand, province or state}, with the selection constant "year = 2010". Also, suppose that there are four materialized cuboids available, as follows:

  cuboid 1: {year, item name, city}
  cuboid 2: {year, brand, country}
  cuboid 3: {year, brand, province or state}
  cuboid 4: {item name, province or state} where year = 2010

- **"Which of the above four cuboids should be selected to process the query?"**
- Finer-granularity data cannot be generated from coarser-granularity data. Therefore, cuboid 2 cannot be used because country is a more general concept than province or state.Cuboids1,3,and 4 can be used to process the query because (1)they have the same set or a superset of the dimensions in the query,(2) the selection clause in the query can imply the selection in the cuboid, and(3) the abstractionlevelsfor the item and location dimensions in these cuboids are at a finer level than brand and province or state, respectively.

- **"How would the costs of each cuboid compare if used to process the query?"**
- It is likely that using cuboid 1 would cost the most because both item name and city are at a lower level than the brand and province or state concepts specified in the query. If there are not many year values associated with items in the cube, but there are several item names for each brand, then cuboid 3 will be smaller than cuboid 4, and thus cuboid 3 should be chosen to process the query. However, if efficient indices are available for cuboid 4, then cuboid 4 may

be a better choice. Therefore, some cost-based estimation is required in order to decide which set of cuboids should be selected for query processing.

2.1.4 OLAP Server Architectures: ROLAP vs. MOLAP vs. HOLAP

- Logically, OLAPservers present business users with multidimensional data from data warehouses or data marts, without concerns regarding how or where the data are stored. However, the physical architecture and implementation of OLAP servers must consider data storage issues. Implementations of a warehouse server for OLAP processing include the following:

- **Relational OLAP(ROLAP) servers:** These are the intermediate servers that stand in between a relational back-end server and client front-end tools. They use a relational or extended-relational DBMS to store and manage warehouse data, and OLAP middleware to support missing pieces. ROLAP servers include optimization for each DBMS back end, implementation of aggregation navigation logic, and additional tools and services. ROLAP technology tends to have greater capability than MOLAP technology. The DSS server of Micro strategy, for example, adopts the ROLAP approach.

- **Multidimensional OLAP(MOLAP)servers:** These servers support multi-dimensional views of data through array-based multidimensional storage engines. They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to pre computed summarized data. Notice that with multidimensional data stores, the storage utilization may be low if the data set is sparse. In such cases, sparse matrix compression techniques should be explored. Many MOLAP servers adopt a two-level storage representation to handle dense and sparse data sets: denser sub cubes are identified and stored as array structures, where as sparse sub cubes employ compression technology for efficient storage utilization.

- **Hybrid OLAP (HOLAP) servers:** The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP. For example, a HOLAP server may allow large volumes of detail data to be stored in a relational database, while aggregations are kept in a separate MOLAP store. The Microsoft SQL Server 2000 supports a hybrid OLAP server.

- **Specialized SQL servers:** To meet the growing demand of OLAP processing in relational databases, some database system vendors implement specialized SQL servers that provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

"How are data actually stored in ROLAP and MOLAP architectures?"

Let's first look at ROLAP name implies, ROLAP uses relational tables to store data for online analytical processing. Recall that the fact table associated with a base cuboid is referred to as a base

fact table. The base fact table stores data at the abstraction level indicated by the join keys in the schema for the given data cube. Aggregated data can also be stored in fact tables, referred to as summary fact tables. Some summary fact tables store both base fact table data and aggregated data, Alternatively, separate summary fact tables can be used for each level of abstraction, to store only aggregated data.

- **Example 2.5 A ROLAP data store.** Table 2.1 shows a summary fact table that contains both base fact data and aggregated data. The schema of the table is "record identifier(RID),item, ..., day, month, quarter, year, dollars_sold", where day, month, quarter, and year define the date of sales, and dollars sold is the sales amount. Consider the tuples with an RID of 1001 and 1002, respectively.
- The data of these tuples are at the base fact level, where the date of sales is October 15, 2010, and October 23, 2010, respectively. Consider the tuple with an RID of 5001. This tuple is at a more general level of abstraction than the tuples 1001 and 1002. The day value has been generalized to all, so that the corresponding time value is October 2010. That is, the dollars sold amount shownis an aggregation representing the entire month ofOctober2010,ratherthanjustOctober15 or23,2010. The special value all is used to represent subtotals in summarized data.MOLAP uses multidimensional array structures to store data for online analytical processing. Most data warehouse systems adopt a client-server architecture. A relational data store always resides at the data warehouse/data mart server site.A multidimensional data store can reside at either the database server site or the client site.

Table 4.4: Single table for base and summary facts.

RID	*item*	...	*day*	*month*	*quarter*	*year*	*dollars_sold*
1001	TV	...	15	10	Q4	2010	250.60
1002	TV	...	23	10	Q4	2010	175.00
...	...	...	...	...	...	...	...
5001	TV	...	all	10	Q4	2010	45,786.08
...	...	...	...	...	...	...	...

Table 2.1: Single table for base and summary facts.

2.2.1 What Is Data Mining?

- Data mining is the process of automatically discovering useful information in large data repositories.
- Data mining techniques are deployed to scour large databases in order to find novel and useful patterns that might otherwise remain unknown.
- They also provide capabilities to predict the outcome of a future observation, such as predicting whether a newly arrived. Not all information discovery tasks are considered to be data mining.

- For example, looking up individual records using a database management system or finding particular Web pages via a query to an Internet search engine are tasks related to the area of information retrieval.
- Although such tasks are important and may involve the use of the sophisticated algorithms and data structures, they rely on traditional computer science techniques and obvious features of the data to create index structures for efficiently organizing and retrieving information.

2.2.2Data Mining and Knowledge Discovery

- Data mining is an integral part of knowledge discovery in databases (KDD), which is the overall process of converting raw data into useful information as shown in Figure 2.2.2.
- This process consists of a series of transformation steps, from data preprocessing to post-processing of data mining results.
- The input data can be stored in a variety of formats (flat files, spreadsheets, or relational tables) and may reside in a centralized data repository or be distributed across multiple sites. The purpose of preprocessing is to transform the raw input data into an appropriate format for subsequent analysis.
- The steps involved in data preprocessing include fusing data from multiple sources, cleaning data to remove noise and duplicate observations, and selecting records and features that are relevant to the data mining task at hand. Because of the many ways data can be collected and stored, data preprocessing is perhaps the most laborious and time-consuming step in the overall knowledge discovery process.
- ”Closing the loop" is the phrase often used to refer to the process of integrating data mining results into decision support systems.
- For example, in business applications, the insights offered by data mining results can be integrated with campaign management tools so that effective marketing promotions can be conducted and tested. Such integration requires a post processing step that ensures that only valid and useful results are incorporated into the decision support system.
- An example of post processing is visualization which allows analysts to explore the data and the data mining results from a variety of viewpoints. Statistical measures or hypothesis testing methods can also be applied during post processing to eliminate spurious data mining results.

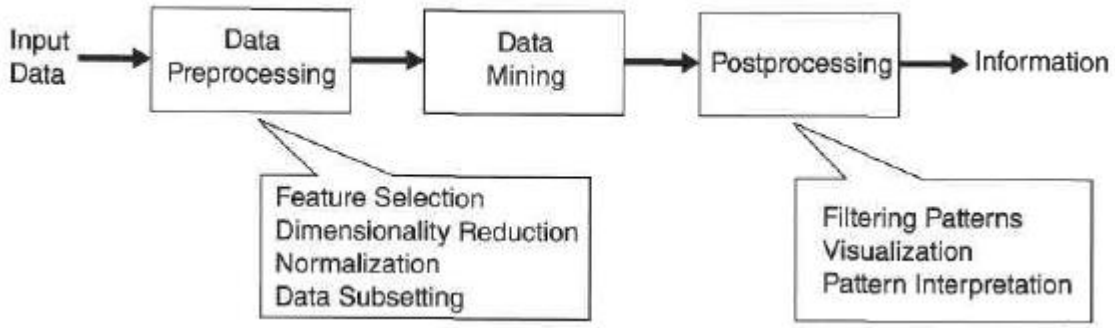


Figure2.2.2. The process of knowledge discovery database(KDD).

2.3 Motivating Challenges

- The following are some of the specific challenges that motivated the development of data mining.
- **Scalability:** Because of advances in data generation and collection, data sets with sizes of gigabytes, terabytes, or even petabytes are becoming common. If data mining algorithms are to handle these massive data sets, then they must be scalable. Many data mining algorithms employ special search strategies to handle exponential search problems. Scalability may also require the implementation of novel data structures to access individual records in an efficient manner. For instance, out-of-core algorithms may be necessary when processing data sets that cannot fit into main memory. Scalability can also be improved by using sampling or developing parallel and distributed algorithms.
- **High Dimensionality:** It is now common to encounter data sets with hundreds or thousands of attributes instead of the handful common a few decades ago. In bioinformatics, progress in microarray technology has produced gene expression data involving thousands of features. Data sets with temporal or spatial components also tend to have high dimensionality. For example, consider a data set that contains measurements of temperature at various locations. If the temperature measurements are taken repeatedly for an extended period, the number of dimensions (features) increases in proportion to the number of measurements taken. Traditional data analysis techniques that were developed for low-dimensional data often do not work well for such high dimensional data. Also, for some data analysis algorithms, the computational complexity increases rapidly as the dimensionality (the number of features) increases.
- **Heterogeneous and Complex Data:** Traditional data analysis methods often deal with data sets containing attributes of the same type, either continuous or categorical. As the role of data mining in business, science, medicine, and other fields has grown, so has the need for techniques that can handle heterogeneous attributes. Recent years have also seen the emergence of more complex data objects. Examples of such non-traditional types of data include collections of Web pages containing semi-structured text and hyperlinks; DNA data with sequential and three-dimensional structure; and climate data that consists of time series measurements (temperature, pressure, etc.) at various locations on the Earth's surface. Techniques developed for mining such complex objects should take into consideration relationships in the data, such as temporal and spatial autocorrelation, graph connectivity, and parent-child relationships between the elements in semi-structured text and XML documents. Data ownership and Distribution Sometimes, the data needed for an analysis is not stored in one location or owned by one organization. Instead, the data is geographically distributed among resources belonging to multiple entities. This requires the development of distributed data mining techniques. Among the key challenges faced by distributed data mining algorithms include (1) how to reduce the amount of communication needed to perform the distributed computatior, (2) how to effectively consolidate the data mining results obtained from multiple sources, and (3) how to address data security issues.
- **Non-traditional Analysis** The traditional statistical approach is based on a hypothesize-and-test paradigm. In other words, a hypothesis is proposed, an experiment is designed to gather

the data, and then the data is analyzed with respect to the hypothesis. Unfortunately, this process is extremely labor intensive. Current data analysis tasks often require the generation and evaluation of thousands of hypotheses, and consequently, the development of some data mining techniques has been motivated by the desire to automate the process of hypothesis generation and evaluation. Furthermore, the data sets analyzed in data mining are typically not the result of a carefully designed experiment and often represent opportunistic samples of the data, rather than random samples. Also, the data sets frequently involve non-traditional types of data and data distributions.

2.4 Data Mining Tasks

Data mining tasks are generally divided into two major categories:

- **Predictive tasks:** The objective of these tasks is to predict the value of a particular attribute based on the values of other attributes. The attribute to be predicted is commonly known as the target or dependent variable, while the attributes used for making the prediction are known as the explanatory or independent variables.
- **Descriptive tasks:** Here, the objective is to derive patterns (correlations, trends, clusters, trajectories, and anomalies) that summarize the underlying relationships in data. Descriptive data mining tasks are often exploratory in nature and frequently require postprocessing techniques to validate and explain the results.

- Figure 2.4.1 illustrates four of the core data mining tasks that are described in the remainder of this book.

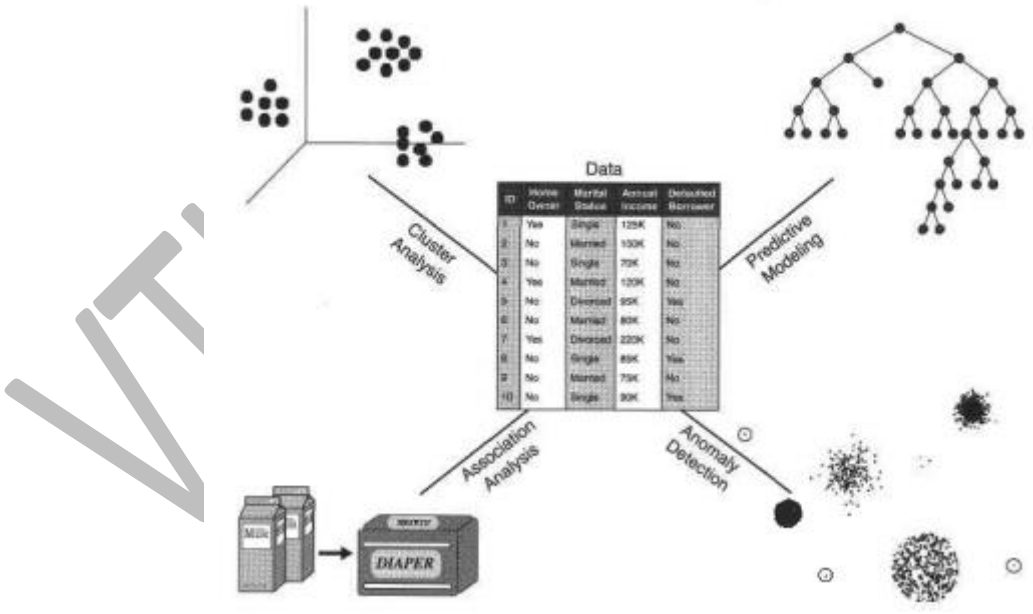


Figure 2.4.1: Four of the core data mining tasks.

- **Predictive modeling** refers to the task of building a model for the target variable as a function of the explanatory variables.
- There are two types of predictive modeling tasks: **classification,** which is used for discrete target variables, and regression, which is used for continuous target variables.

For example, predicting whether a Web user will make a purchase at an online bookstore is a classification task because the target variable is binary-valued. On the other hand, forecasting the future price of a stock is a regression task because price is a continuous-valued attribute. The goal of both tasks is to learn a model that minimizes the error between the predicted and true values of the target variable. **Predictive modeling** can be used to identify customers that will respond to a marketing campaign, predict disturbances in the Earth's ecosystem, or judge whether a patient has a particular disease based on the results of medical tests.

- **Example 2.4.1** (Predicting the Type of a Flower). Consider the task of predicting a species of flower based on the characteristics of the flower. In particular, consider classifying an Iris flower as to whether it belongs to one of the following three Iris species: Setosa, Versicolour, or Virginica. To perform this task, we need a data set containing the characteristics of various flowers of these three species. A data set with this type of information is the well-known Iris data set from the UCI Machine Learning Repository at http: /hrurw.ics.uci.edu/-mlearn. In addition to the species of a flower, this data set contains four other attributes: sepal width, sepal length, petal length, and petal width. (The Iris data set and its attributes are described further in Section 3.1.) Figure 2.4.2 shows a plot of petal width versus petal length for the 150 flowers in the Iris data set. Petal width is broken into the categories low, med'ium, and hi'gh, which correspond to the intervals [0' 0.75), [0.75, 1.75), [1.75, ∞), respectively. Also, petal length is broken into categories low, med,'ium, and hi,gh, which correspond to the intervals [0' 2.5), [2.5,5), [5,∞), respectively. Based on these categories of petal width and length, the following rules can be derived: Petal width low and petal length low implies Setosa. Petal width medium and petal length medium implies Versicolour. Petal width high and petal length high implies Virginica. While these rules do not classify all the flowers, they do a good (but not perfect) job of classifying most of the flowers. Note that flowers from the Setosa species are well separated from the Versicolour and Virginica species with respect to petal width and length, but the latter two species overlap somewhat with respect to these attributes.

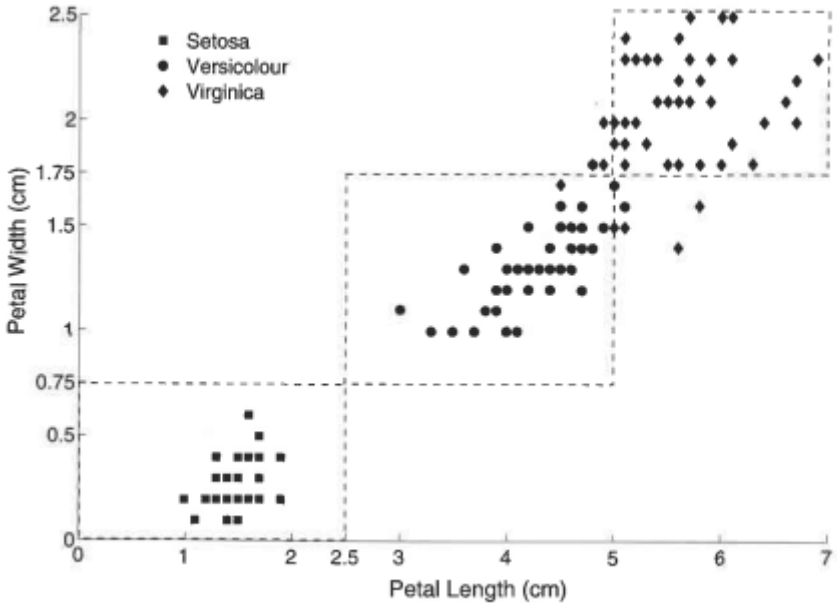


Figure 2.4.2: Petal width versus petal length for 150 Iris flowers

- **Association analysis:** is used to discover patterns that describe strongly associated features in the data. The discovered patterns are typically represented in the form of implication rules or feature subsets. Because of the exponential size of its search space, the goal of association analysis is to extract the most interesting patterns in an efficient manner. Useful applications of association analysis include finding groups of genes that have related functionality, identifying Web pages that are accessed together, or understanding the relationships between different elements of Earth's climate system.
- **Example 2.4.2** (Market Basket Analysis). The transactions shown in Table 2.4.1 illustrate point-of-sale data collected at the checkout counters of a grocery store. Association analysis can be applied to find items that are frequently bought together by customers. For example, we may discover the rule {Diapers} ----> {Milk}, which suggests that customers who buy diapers also tend to buy milk. This type of rule can be used to identify potential cross-selling opportunities among related items.

Table 2.4.1 Market Basket Data

Transaction ID	Items
1	{Bread, Butter, Diapers, Milk}
2	{Coffee, Sugar, Cookies, Salmon}
3	{Bread, Butter, Coffee, Diapers, Milk, Eggs}
4	{Bread, Butter, Salmon, Chicken}
5	{Eggs, Bread, Butter}
6	{Salmon, Diapers, Milk}
7	{Bread, Tea, Sugar, Eggs}
8	{Coffee, Sugar, Chicken, Eggs}
9	{Bread, Diapers, Milk, Salt}
10	{Tea, Eggs, Cookies, Diapers, Milk}

- **Cluster analysis** seeks to find groups of closely related observations so that observations that belong to the same cluster are more similar to each other than observations that belong to other clusters. Clustering has been used to group sets of related customers, find areas of the ocean that have a significant impact on the Earth's climate, and compress data.
- **Example 2.4.3 (Document Clustering).** The collection of news articles shown in Table 2.4.2 can be grouped based on their respective topics. Each article is represented as a set of word-frequency pairs (w, c), where **w** is a word and **c** is the number of times the word appears in the article. There are two natural clusters in the data set. The first cluster consists of the first four articles, which correspond to news about the economy, while the second cluster contains the last four articles, which correspond to news about health care. A good clustering algorithm should be able to identify these two clusters based on the similarity between words that appear in the articles.

Table 2.4.2 Collection of news articles

Transaction ID	Items
1	{Bread, Butter, Diapers, Milk}
2	{Coffee, Sugar, Cookies, Salmon}
3	{Bread, Butter, Coffee, Diapers, Milk, Eggs}
4	{Bread, Butter, Salmon, Chicken}
5	{Eggs, Bread, Butter}
6	{Salmon, Diapers, Milk}
7	{Bread, Tea, Sugar, Eggs}
8	{Coffee, Sugar, Chicken, Eggs}
9	{Bread, Diapers, Milk, Salt}
10	{Tea, Eggs, Cookies, Diapers, Milk}

- **Anomaly detection** is the task of identifying observations whose characteristics are significantly different from the rest of the data. Such observations are known as anomalies or outliers. The goal of an anomaly detection algorithm is to discover the real anomalies and avoid falsely labeling normal objects as anomalous. In other words, a good anomaly detector must have a high detection rate and a low false alarm rate. Applications of anomaly detection include the detection of fraud, network intrusions, unusual patterns of disease, and ecosystem disturbances.
- **Example 2.4.4 (Credit Card fraud Detection).** A credit card company records the transactions made by every credit card holder, along with personal information such as credit limit, age, annual income, and address. since the number of fraudulent cases is relatively small compared to the number of legitimate transactions, anomaly detection techniques can be applied to build a profile of legitimate transactions for the users. When a new transaction arrives, it is compared against the profile of the user. If the characteristics of the transaction are

very different from the previously created profile, then the transaction is flagged as potentially fraudulent.

2.5 Types of Data

- A data set can often be viewed as a collection of **data objects**. Other names for a data object are record, point, vector, pattern, event, case, sample, observation, or entity. In turn, data objects are described by a number of attributes that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred. Other names for an attribute are variable, characteristics, field, feature, dimensions.
- Example 2.5.1 (Student Information). Often, a data set is a file, in which the objects are records (or rows) in the file and each field (or column) corresponds to an attribute. For example, Table 2.5.1 shows a data set that consists of student information. Each row corresponds to a student and each column is an attribute that describes some aspect of a student, such as grade point average (GPA) or identification number (ID).

Table 2.5.1 A sample dataset containing student information

Student ID	Year	Grade Point Average (GPA)	.
	⋮		
1034262	Senior	3.24	.
1052663	Sophomore	3.51	.
1082246	Freshman	3.62	.
	⋮		

2.5.1 Attributes and Measurement

- In this section we address the issue of describing data by considering what types of attributes are used to describe data objects. We first define an attribute, then consider what we mean by the type of an attribute, and finally describe the types of attributes that are commonly encountered.
- An **attribute** is a property or characteristic of an object that may vary; either from one object to another or from one time to another. For example, eye color varies from person to person, while the temperature of an object varies over time. Note that eye color is a symbolic attribute with a small number of possible values {brown, black, blue, green, hazel, etc.}, while temperature is a numerical attribute with a potentially unlimited number of values.
- At the most basic level, attributes are not about numbers or symbols.
- **A measurement scale** is a rule (function) that associates a numerical or symbolic value with an attribute of an object.

- Formally, the process of measurement is the application of a measurement scale to associate a value with a particular attribute of a specific object.
- While this may seem a bit abstract, we engage in the process of measurement all the time.
- For instance, we step on a bathroom scale to determine our weight, we classify someone as male or female, or we count the number of chairs in a room to see if there will be enough to seat all the people coming to a meeting. In all these cases) the "physical value" of an attribute of an object is mapped to a numerical or symbolic value.

The Types of an attribute

- It should be apparent from the previous discussion that the properties of an attribute need not be the same as the properties of the values used to measure it. In other words, the values used to represent an attribute may have properties that are not properties of the attribute itself, and vice versa. This is illustrated with two examples.
- **Example 2.5.2 (Employee Age and ID Number).** Two attributes that might be associated with an employee are ID and age (in years). Both of these attributes can be represented as integers. However, while it is reasonable to talk about the average age of an employee, it makes no sense to talk about the average employee ID. Indeed, the only aspect of employees that we want to capture with the ID attribute is that they are distinct. Consequently, the only valid operation for employee IDs is to test whether they are equal. There is no hint of this limitation, however, when integers are used to represent the employee ID attribute. For the age attribute, the properties of the integers used to represent age are very much the properties of the attribute. Even so, the correspondence is not complete since, for example, ages have a maximum while integers do not.

- **Example 2.5.3(Length of Line Segments).** Consider Figure 2.5, which shows some objects-line segments and how the length attribute of these objects can be mapped to numbers in two different ways. Each successive line segment, going from the top to the bottom, is formed by appending the topmost line segment to itself. Thus, the second line segment from the top is formed by appending the topmost line segment to itself twice, the third line segment from the top is formed by appending the topmost line segment to itself three times, and so forth. In a very real (physical) sense, all the line segments are multiples of the first. This fact is captured by the measurements on the right-hand side of the figure, but not by those on the left hand-side. More specifically, the measurement scale on the left-hand side captures only the ordering of the length attribute, while the scale on the right-hand side captures both the ordering and additivity properties. Thus, an attribute can be measured in a way that does not capture all the properties of the attribute. The type of an attribute should tell us what properties of the attribute are reflected in the values used to measure it. Knowing the type of an attribute is important because it tells us which properties of the measured values are consistent with the underlying properties of the attribute, and therefore, it allows us to avoid foolish actions, such as computing the average employee ID. Note that it is common to refer to the type of an attribute as the type of a measurement scale.

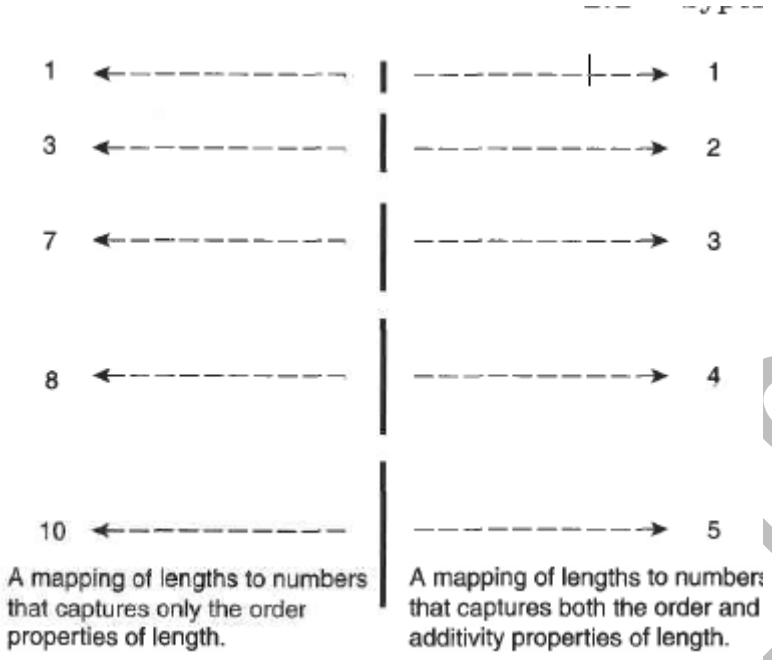


Figure2.5 The measurement of the length of line segments on two different scales of measurement

The Different Types of Attributes

- A useful way to specify the type of an attribute is to identify the properties of numbers that correspond to underlying properties of the attribute.
- The following properties (operations) of numbers are typically used to describe attributes.

1. Distinctness = and ≠
2. Order <, ≤, > and ≥
3. Addition + and -
4. Multiplication *and /

- Given these properties, we can define four types of attributes: **nominal, ordinal, interval, and ratio. Table 2.5.2** gives the definitions of these types, along with information about the statistical operations that are valid for each type.

Table 2.5.2 : Different attribute types

Attribute Type		Description	Examples	Operations
Categorical (Qualitative)	Nominal	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another. $(=, \neq)$	zip codes, employee ID numbers, eye color, gender	mode, entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. $(<, >)$	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. $(+, -)$	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
	Ratio	For ratio variables, both differences and ratios are meaningful. $(*, /)$	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

- Each attribute type possesses all of the properties and operations of the attribute types above it. Consequently, any property or operation that is valid for nominal, ordinal, and interval attributes is also valid for ratio attributes.
- Nominal and ordinal attributes are collectively referred to as **categorical** or **qualitative** attributes.
- As the name suggests, qualitative attributes, such as employee ID, lack most of the properties of numbers. Even if they are represented by numbers, i.e., integers, they should be treated more like symbols.
- The remaining two types of attributes, interval and ratio, are collectively referred to as quantitative or numeric attributes. Quantitative attributes are represented by numbers and have most of the properties of numbers.
- Note that quantitative attributes can be integer-valued or continuous. The types of attributes can also be described in terms of transformations that do not change the meaning of an attribute.
- Indeed the types of attributes shown in **Table 5.2.3**, defined them in terms of these permissible transformations. For example

Table 5.2.3 : Transformations that define attribute levels

Attribute Type		Description	Examples	Operations
Categorical (Qualitative)	Nominal	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another. $(=, \neq)$	zip codes, employee ID numbers, eye color, gender	mode, entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. $(<, >)$	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. $(+, -)$	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
	Ratio	For ratio variables, both differences and ratios are meaningful. $(*, /)$	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Example 2.5.4 (Temperature Scales): Temperature provides a good illustration of some of the concepts that have been described. First, temperature can be either an interval or a ratio attribute, depending on its measurement scale. When measured on the Kelvin scale, a temperature of 2o is, in a physically meaningful way, twice that of a temperature of 1o. This is not true when temperature is measured on either the Celsius or Fahrenheit scales, because, physically, a temperature of 1o Fahrenheit (Celsius) is not much different than a temperature of 2" Fahrenheit (Celsius). The problem is that the zero points of the Fahrenheit and Celsius scales are, in a physical sense, arbitrary, and therefore, the ratio of two Celsius or Fahrenheit temperatures is not physically meaningful.

Describing Attributes by the number of values

- An independent way of distinguishing between attributes is by the number of values they can take.
- **Discrete**: A discrete attribute has a finite or countably infinite set of values. Such attributes can be categorical, such as zip codes or ID numbers, or numeric, such as counts. Discrete attributes are often represented using integer variables. **Binary attributes** are a special case of discrete attributes and assume only two values, e.g., true/false, yes/no, male/female, or 0f 1. Binary attributes are often represented as Boolean variables, or as integer variables that only take the values 0 or 1.
- **Continuous:** A continuous attribute is one whose values are real numbers. Examples include attributes such as temperature, height, or weight. Continuous attributes are typically represented as floating-point variables. Practically, real values can only be measured and represented with limited precision.

- **Asymmetric Attributes:** For asymmetric attributes, only presence a non-zero attribute value-is regarded as important. Consider a data set where each object is a student and each attribute records whether or not a student took a particular course at a university. For a specific student, an attribute has a value of 1 if the student took the course associated with that attribute and a value of 0 otherwise. Because students take only a small fraction of all available courses, most of the values in such a data set would be 0.
- Therefore, it is more meaningful and more efficient to focus on the non-zero values. To illustrate, if students are compared on the basis of the courses they don't take, then most students would seem very similar, at least if the number of courses is large. Binary attributes where only non-zero values are important are called asymmetric binary attributes. This type of attribute is particularly important for association analysis, which is discussed in Chapter 6. It is also possible to have discrete or continuous asymmetric features. For instance, if the number of credits associated with each course is recorded, then the resulting data set will consist of asymmetric discrete or continuous attributes.

2.5.2 Types of Data Sets

1. There are many types of data sets, and as the field of data mining develops and matures, a greater variety of data sets become available for analysis. In this section, we describe some of the most common types. For convenience, we have grouped the types of data sets into three groups: record data, graph based data, and ordered data.

General Characteristics of Data Sets

2. Before providing details of specific kinds of data sets, we discuss three characteristics that apply to many data sets and have a significant impact on the data mining techniques that are used: dimensionality, sparsity, and resolution.
3. **Dimensionality:** The dimensionality of a data set is the number of attributes that the objects in the data set possess. Data with a small number of dimensions tends to be qualitatively different than moderate or high-dimensional data. Indeed, the difficulties associated with analyzing high-dimensional data are sometimes referred to as the curse of dimensionality. Because of this, an important motivation in preprocessing the data is dimensionality reduction
4. **Sparsity:** For some data sets, such as those with asymmetric features, most attributes of an object have values of 0; in many cases, fewer than 1% of the entries are non-zero. In practical terms, sparsity is an advantage because usually only the non-zero values need to be stored and manipulated. This results in significant savings with respect to computation time and storage.
5. **Resolution**: It is frequently possible to obtain data at different levels of resolution, and often the properties of the data are different at different resolutions. For instance, the surface of the Earth seems very uneven at a resolution of a few meters, but is relatively smooth at a resolution of tens of kilometers. The patterns in the data also depend on the level of resolution. If the resolution is too fine, a pattern may not be visible or may be buried in noise; if the resolution is too coarse, the pattern may disappear.

Types of data sets

- **Record Data**:

  Much data mining work assumes that the data set is a collection of records (data objects), each of which consists of a fixed set of data fields (attributes). See Figure 2.6(a). For the most basic form of record data, there is no explicit relationship among records or data fields, and every record (object) has the same set of attributes. Record data is usually stored either in flat files or in relational databases. Relational databases are certainly more than a collection of records, but data mining often does not use any of the additional information available in a relational database. Rather, the database serves as a convenient place to find records.

Different types of record data are described below and are illustrated in Figure 2.6.

1. **Transaction or Market Basket Data**: Transaction data is a special type of record data, where each record (transaction) involves a set of items. Consider a grocery store. The set of products purchased by a customer during one shopping trip constitutes a transaction, while the individual products that were purchased are the items. This type of data is called market basket data because the items in each record are the products in a person's "market basket." Transaction data is a collection of sets of items, but it can be viewed as a set of records whose fields are asymmetric attributes. Most often, the attributes are binary, indicating whether or not an item was purchased, but more generally, the attributes can be discrete or continuous, such as the number of items purchased or the amount spent on those items. Figure 2.6(b) shows a sample transaction data set. Each row represents the purchases of a particular customer at a particular time.
2. **Data Matrix**: If the data objects in a collection of data all have the same fixed set of numeric attributes, then the data objects can be thought of as points (vectors) in a multidimensional space, where each dimension represents a distinct attribute describing the object. A set of such data objects can be interpreted as an **m** by **n** matrix, where there are **m** rows, one for each object, and **n** columns, one for each attribute. This matrix is called a **data matrix or a pattern matrix**. A data matrix is a variation of record data, but because it consists of numeric attributes, standard matrix operation can be applied to transform and manipulate the data. Therefore, the data matrix is the standard data format for most statistical data. Figure 2.6(c) shows a sample data matrix.
3. **The Sparse Data Matrix**: A sparse data matrix is a special case of a data matrix in which the attributes are of the same type and are asymmetric; i.e., only non-zero values are important. Transaction data is an example of a sparse data matrix that has only 0-1 entries. Another common example is document data. In particular, if the order of the terms (words) in a document is ignored, then a document can be represented as a term vector, where each term is a component (attribute) of the vector and the value of each component is the number of times the corresponding term occurs in the document. This representation of a collection of documents is often called a document-term matrix. Figure 2.6(d) shows a sample document-term matrix. The documents are the rows of

this matrix, while the terms are the columns. In practice, only the non-zero entries of sparse data matrices are stored.

Tid	Refund	Marital Status	Taxable Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(a) Record data.

TID	ITEMS
1	Bread, Soda, Milk
2	Beer, Bread
3	Beer, Soda, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Soda, Diaper, Milk

(b) Transaction data.

Projection of x Load	Projection of y Load	Distance	Load	Thickness
10.23	5.27	15.22	27	1.2
12.65	6.25	16.22	22	1.1
13.54	7.23	17.34	23	1.2
14.27	8.43	18.45	25	0.9

(c) Data matrix.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

(d) Document-term matrix.

Figure 2.6 Different variations of record data

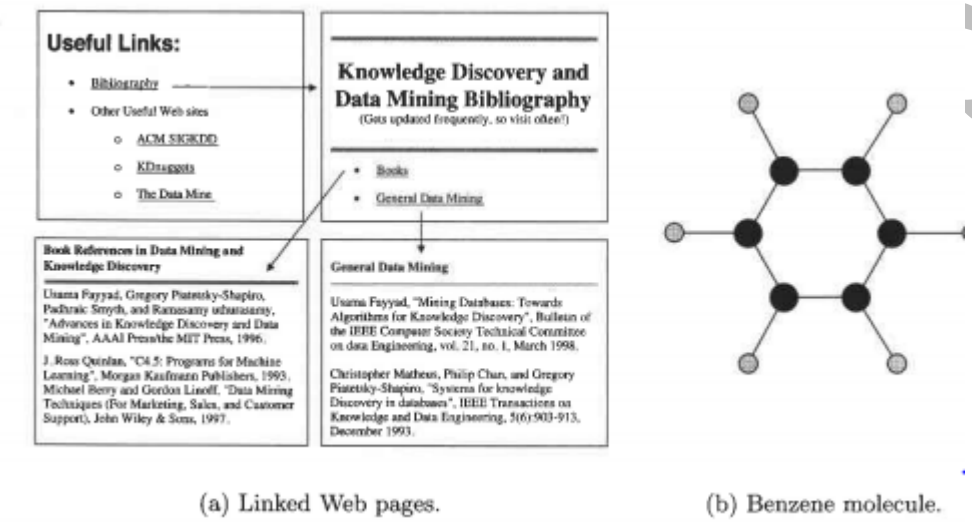
- **Graph-Based Data**

A graph can sometimes be a convenient and powerful representation for data. We consider two specific cases: (1) the graph captures relationships among data objects and (2) the data objects themselves are represented as graphs.

1. **Data with Relationships among Objects:**

The relationships among objects frequently convey important information. In such cases, the data is often represented as a graph. In particular, the data objects are mapped to nodes of the graph, while the relationships among objects are captured by the links between objects and link properties, such as direction and weight. Consider Web pages on the World Wide Web, which contain both text and links to other pages. In order to process search queries, Web search engines collect and process Web pages to extract their contents. It is well known, however, that the links to and from each page provide a great deal of information about the relevance of a Web page to a query, and thus, must also be taken into consideration. Figure 2.7(a) shows a set of linked Web pages.

2. **Data with Objects That Are Graphs** If objects have structure, that is, the objects contain sub objects that have relationships, then such objects are frequently represented as graphs. For example, the structure of chemical compounds can be represented by a graph, where the nodes are atoms and the links between nodes are chemical bonds. Figure 2.7(b) shows a ball-and-stick diagram of the chemical compound benzene, which contains atoms of carbon (black) and hydrogen (gray). A graph representation makes it possible to determine which substructures occur frequently in a set of compounds and to ascertain whether the presence of any of these substructures is associated with the presence or absence of certain chemical properties, such as melting point or heat of formation.



(a) Linked Web pages. (b) Benzene molecule.

Figure 2.7 Different variations of graph data

3. Ordered Data

For some types of data, the attributes have relationships that involve order in time or space. Different types of ordered data are described next and are shown in Figure 2.8

- **Sequential Data** : Sequential data, also referred to as temporal data, can be thought of as an extension of record data, where each record has a time associated with it. Consider a retail transaction data set that also stores the time at which the transaction took place. This time information makes it possible to find patterns such as "candy sales peak before Halloween." A time can also be associated with each attribute. For example, each record could be the purchase history of a customer, with a listing of items purchased at different times. Using this information, it is possible to find patterns such as "people who buy DVD players tend to buy DVDs in the period immediately following the purchase." Figure 2.8(a) shows an example of sequential transaction data. There are five different times-t1, t2, t3, t4, and t5; three different customers-Cl, C2, and C3; and five different items A, B, C, D, and E. In the top table, each row corresponds to the items purchased at a particular time by each customer. For instance, at time f3, customer C2 purchased items A and D. In the bottom table, the same information is displayed, but each row corresponds to a particular customer. Each row contains information on each transaction involving the customer, where a transaction is considered to be a set of

items and the time at which those items were purchased. For example, customer C3 bought items A and C at time t2.

- **Sequence Data**: Sequence data consists of a data set that is a sequence of individual entities, such as a sequence of words or letters. It is quite similar to sequential data, except that there are no time stamps; instead, there are positions in an ordered sequence. For example, the genetic information of plants and animals can be represented in the form of sequences of nucleotides that are known as genes. Many of the problems associated with genetic sequence data involve predicting similarities in the structure and function of genes from similarities in nucleotide sequences. Figure 2.8(b) shows a section of the human genetic code expressed using the four nucleotides from which all DNA is constructed: A, T, G, and C.
- **Time Series Data:** Time series data is a special type of sequential data in which each record is a time series, i.e., a series of measurements taken over time. For example, a financial data set might contain objects that are time series of the daily prices of various stocks. As another example, consider Figure 2.8(c), which shows a time series of the average monthly temperature for Minneapolis during the years 1982 to 1994. When working with temporal data, it is important to consider temporal autocorrelation; i.e., if two measurements are close in time, then the values of those measurements are often very similar.
- **Spatial Data**: Some objects have spatial attributes, such as positions or areas, as well as other types of attributes. An example of spatial data is weather data (precipitation, temperature, pressure) that is collected for a variety of geographical locations. An important aspect of spatial data is spatial autocorrelation; i.e., objects that are physically close tend to be similar in other ways as well. Thus, two points on the Earth that are close to each other usually have similar values for temperature and rainfall. Important examples of spatial data are the science and engineering data sets that are the result of measurements or model output taken at regularly or irregularly distributed points on a two- or three-dimensional grid or mesh. For instance, Earth science data sets record the temperature or pressure measured at points (grid cells) on latitude-longitude spherical grids of various resolutions, ex., 1^{o} by 1_{o}. (See Figure 2.8(d).) As another example, in the simulation of the flow of a gas, the speed and direction of flow can be recorded for each grid point in the simulation.

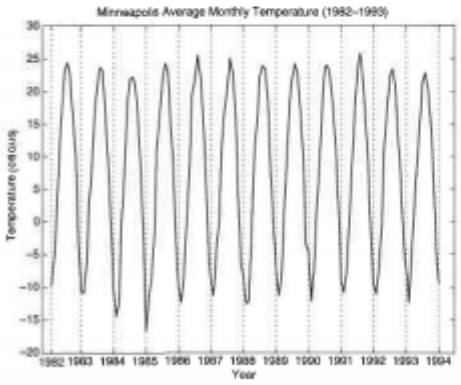
Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A,B) (t2:C,D) (t5:A,E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)

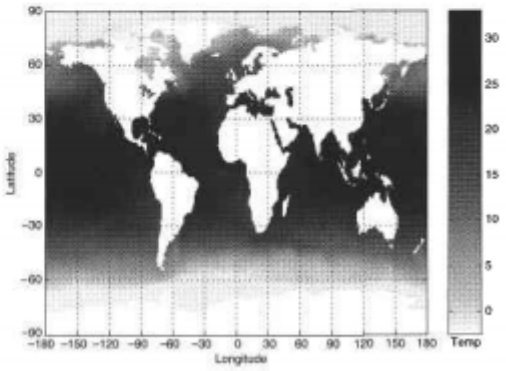
(a) Sequential transaction data.

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

(b) Genomic sequence data.



(c) Temperature time series.



(d) Spatial temperature data.

Figure 2.8 Different variations of graph data

- **Handling Non-Record Data**: Most data mining algorithms are designed for record data or its variations, such as transaction data and data matrices. Record-oriented techniques can be applied to non-record data by extracting features from data objects and using these features to create a record corresponding to each object. Consider the chemical structure data that was described earlier. Given a set of common substructures, each compound can be represented as a record with binary attributes that indicate whether a compound contains a specific substructure. Such a representation is actually a transaction data set, where the transactions are the compounds and the items are the substructures.
    1. In some cases, it is easy to represent the data in a record format, but this type of representation does not capture all the information in the data. Consider spatio-temporal data consisting of a time series from each point on a spatial grid. This data is often stored in a data matrix, where each row represents a location and each column represents a particular point in time. However, such a representation does not explicitly capture the time relationships that are present among attributes and the spatial relationships that exist among objects. This does not mean that such a representation is inappropriate, but rather that these relationships must be taken into consideration during

the analysis. For example, it would not be a good idea to use a data mining technique that assumes the attributes are statistically independent of one another.

2.6 Data Quality

Data mining applications are often applied to data that was collected for another purpose, or for future, but unspecified applications. For that reason, data mining cannot usually take advantage of the significant benefits of "addressing quality issues at the source." In contrast, much of statistics deals with the design of experiments or surveys that achieve a prespecified level of data quality. Because preventing data quality problems is typically not an option, data mining focuses on (1) the detection and correction of data quality problems and (2) the use of algorithms that can tolerate poor data quality. The first step, detection and correction, is often called data cleaning.

The following sections discuss specific aspects of data quality. The focus is on measurement and data collection issues, although some application-related issues are also discussed.

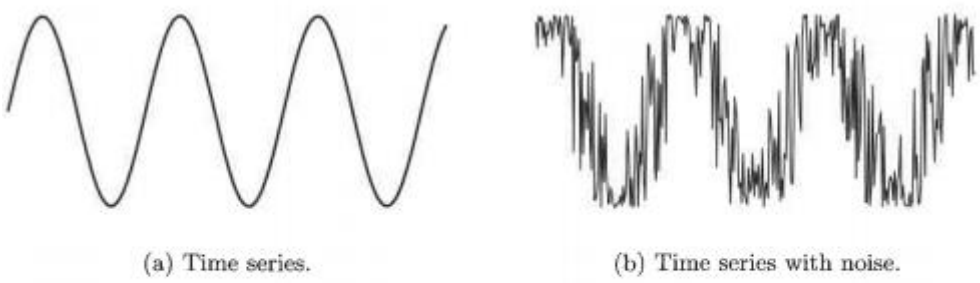
2.6.1 Measurement and Data Collection Issues

- It is unrealistic to expect that data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process.
- Values or even entire data objects may be missing. In other cases, there may be spurious or duplicate objects; i.e., multiple data objects that all correspond to a single "real" object. For example, there might be two different records for a person who has recently lived at two different addresses. Even if all the data is present and "looks fine," there may be inconsistencies-a person has a height of 2 meters, but weighs only 2 kilograms.
- In the next few sections, we focus on aspects of data quality that are related to data measurement and collection. We begin with a definition of measurement and data collection errors and then consider a variety of problems that involve measurement error: noise, artifacts, bias, precision, and accuracy.
- **Measurement and Data Collection Errors** The term measurement error refers to any problem resulting from the measurement process. A common problem is that the value recorded differs from the true value to some extent. For continuous attributes, the numerical difference of the measured and true value is called the error.
- The term data collection error refers to errors such as omitting data objects or attribute values, or inappropriately including a data object. For example, a study of animals of a certain species might include animals of a related species that are similar in appearance to the species of interest.
- Both measurement errors and data collection errors can be either systematic or random. We will only consider general types of errors.

Now will discuss the variety of problems that involve measurement error

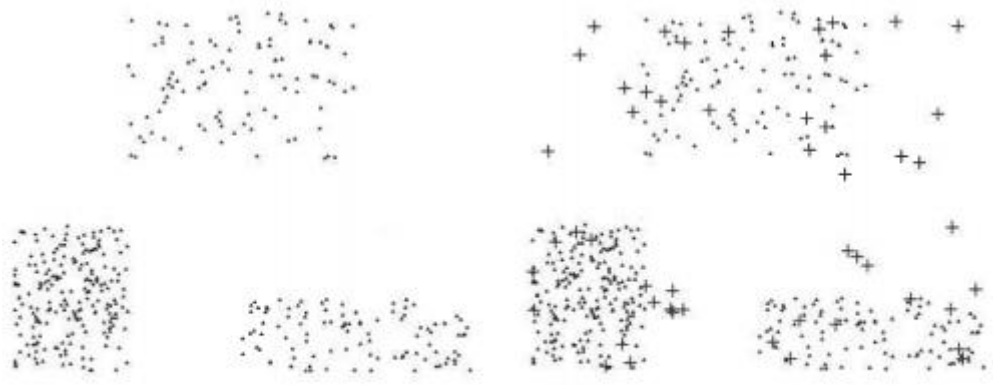
- **Noise and Artifacts:** Noise is the random component of a measurement error. It may involve the distortion of a value or the addition of spurious objects. Figure 2.9 shows a time series

before and after it has been disrupted by random noise. If a bit more noise were added to the time series, its shape would be lost. Figure 2.9 shows a set of data points before and after some noise points (indicated by '+'s) have been added. Notice that some of the noise points are intermixed with the non-noise points.



(a) Time series. (b) Time series with noise.

Figure 2.5. Noise in a time series context.



(a) Three groups of points. (b) With noise points (+) added.

Figure 2.9: Noise in a special context

- **Precision:**The closeness of repeated measurements (of the same quantity) to one another.
- **Bias:** A systematic quantity being measured. Precision is often measured by the standard deviation of a set of values, while bias is measured by taking the difference between the mean of the set of values and the known value of the quantity being measured. Bias can only be determined for objects whose measured quantity is known by means external to the current situation. Suppose that we have a standard laboratory weight with a mass of 1g and want to assess the precision and bias of our new laboratory scale. We weigh the mass five times, and obtain the following five values: {1.015, 0.990, 1.013, 1.001,0.986}. The mean of these values is 1.001, and hence, the bias is 0.001. The precision, as measured by the standard deviation, is 0.013.

- **Accuracy:** The closeness of measurements to the true value of the quantity being measured. Accuracy depends on precision and bias, but since it is a general concept, there is no specific formula for accuracy in terms of these two quantities. One important aspect of accuracy is the

use of significant digits. The goal is to use only as many digits to represent the result of a measurement or calculation as are justified by the precision of the data. For example, if the length of an object is measured with a meter stick whose smallest markings are millimeters, then we should only record the length of data to the nearest millimeter. The precision of such a measurement would be * 0.5mm.

- **Outliers:** Outliers are either (1) data objects that, in some sense, have characteristics that are different from most of the other data objects in the data set, or (2) values of an attribute that are unusual with respect to the typical values for that attribute. Alternatively, we can speak of anomalous objects or values. Furthermore, it is important to distinguish between the notions of noise and outliers. Outliers can be legitimate data objects or values. Thus, unlike noise, outliers may sometimes be of interest. In fraud and network intrusion detection, for example, the goal is to find unusual objects or events from among a large number of normal ones.
- **Missing Values**: It is not unusual for an object to be missing one or more attribute values. In some cases, the information was not collected; e.g., some people decline to give their age or weight. In other cases, some attributes are not applicable to all objects; e.g., often, forms have conditional parts that are filled out only when a person answers a previous question in a certain way, but for simplicity, all fields are stored. Regardless, missing values should be taken into account during the data analysis.
  - There are several strategies (and variations on these strategies) for dealing with missing data, each of which may be appropriate in certain circumstances. These strategies are listed next, along with an indication of their advantages and disadvantages.
  - **Eliminate Data Objects or Attributes:** A simple and effective strategy is to eliminate objects with missing values. However, even a partially specified data object contains some information, and if many objects have missing values, then a reliable analysis can be difficult or impossible. Nonetheless, if a data set has only a few objects that have missing values, then it may be expedient to omit them.
  - **Estimate Missing Values**: Sometimes missing data can be reliably estimated. For example, consider a time series that changes in a reasonably smooth fashion, but has a few, widely scattered missing values. In such cases, the missing values can be estimated (interpolated) by using the remaining values. As another example, consider a data set that has many similar data points. In this situation, the attribute values of the points closest to the point with the missing value are often used to estimate the missing value.
  - **Ignore the Missing Value during Analysis** Many data mining approaches can be modified to ignore missing values. For example, suppose that objects are being clustered and the similarity between pairs of data objects needs to be calculated. If one or both objects of a pair have missing values for some attributes, then the similarity can be calculated by using only the attributes that do not have missing values. It is true that the similarity will only be approximate, but unless the total number of attributes is small or the number of missing values is high, this degree of inaccuracy may not matter much.
- **Inconsistent Values:** Data can contain inconsistent values. Consider an address field, where both a zip code and city are listed, but the specified zip code area is not contained in that city. It may be that the individual entering this information transposed two digits, or perhaps a digit was misread when the information was scanned from a handwritten form. Regardless of the

cause of the inconsistent values, it is important to detect and, if possible, correct such problems. Some types of inconsistences are easy to detect. For instance, a person's height should not be negative. In other cases, it can be necessary to consult an external source of information. For example, when an insurance company processes claims for reimbursement, it checks the names and addresses on the reimbursement forms against a database of its customers.

- **Duplicate Data**

  A data set may include data objects that are duplicates, or almost duplicates, of one another. Many people receive duplicate mailings because they appear in a database multiple times under slightly different names. To detect and eliminate such duplicates, two main issues must be addressed. First, if there are two objects that actually represent a single object, then the values of corresponding attributes may differ, and these inconsistent values must be resolved. Second, care needs to be taken to avoid accidentally combining data objects that are similar, but not duplicates, such as two distinct people with identical names. The term deduplication is often used to refer to the process of dealing with these issues.

2.6.2 Issues Related to Applications

- Data quality issues can also be considered from an application viewpoint as expressed by the statement "data is of high quality if it is suitable for its intended use." This approach to data quality has proven quite useful, particularly in business and industry. As with quality issues at the measurement and data collection level, there are many issues that are specific to particular applications and fields. Again, we consider only a few of the general issues.
- **Timeliness:** Some data starts to age as soon as it has been collected. In particular, if the data provides a snapshot of some ongoing phenomenon or process, such as the purchasing behavior of customers or Web browsing patterns, then this snapshot represents reality for only a limited time. If the data is out of date, then so are the models and patterns that are based on it.
- **Relevance:** The available data must contain the information necessary for the application. Consider the task of building a model that predicts the accident rate for drivers. If information about the age and gender of the driver is omitted, then it is likely that the model will have limited accuracy unless this information is indirectly available through other attributes. Making sure that the objects in a data set are relevant is also challenging. A common problem is sampling bias, which occurs when a sample does not contain different types of objects in proportion to their actual occurrence in the population.
- **Knowledge about the Data:** Ideally, data sets are accompanied by documentation that describes different aspects of the data; the quality of this documentation can either aid or hinder the subsequent analysis. For example, if the documentation identifies several attributes as being strongly related, these attributes are likely to provide highly redundant information, and we may decide to keep just one. (Consider sales tax and purchase price.) If the documentation is poor, however, and fails to tell us, for example, that the missing values for a particular field are indicated with a -9999, then our analysis of the data may be faulty.

2.7 Data Preprocessing

- In this section, we address the issue of which preprocessing steps should be applied to make the data more suitable for data mining.
- Data preprocessing that are interrelated in complex ways. We will present some of the most important ideas and approaches, and try to point out the interrelationships among them.
- We will discuss the following topics:
- Aggregation
- Sampling
- Dimensionality reduction
- Feature subset selection
- Feature creation
- Discretization and binarization
- Variable transformation

2.7.1 Aggregation

- Sometimes "less is more" and this is the case with **aggregation**, the combining of two or more objects into a single object.
- Consider a data set consisting of transactions (data objects) recording the daily sales of products in various store locations (Minneapolis, Chicago, Paris, ...) for different days over the course of a year. See Table 2.7.1. One way to aggregate transactions for this data set is to replace all the transactions of a single store with a single storewide transaction.

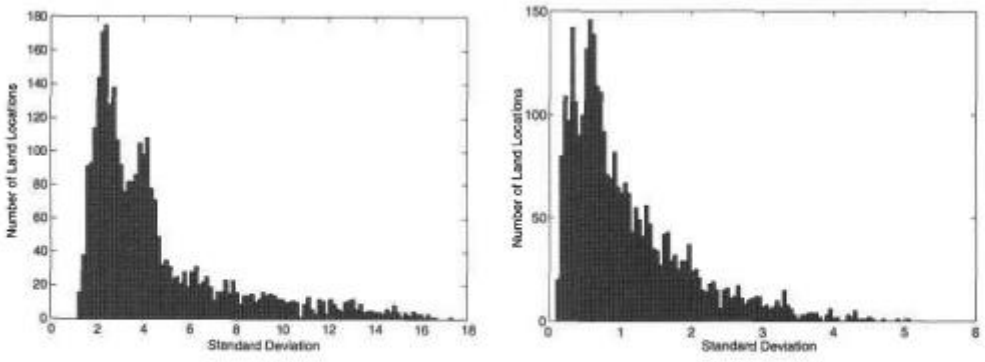
Table 2.7.1: Data set containing information about customer purchases.

Transaction ID	Item	Store Location	Date	Price	...
⋮	⋮	⋮	⋮	⋮	
101123	Watch	Chicago	09/06/04	$25.99	...
101123	Battery	Chicago	09/06/04	$5.99	...
101124	Shoes	Minneapolis	09/06/04	$75.00	...
⋮	⋮	⋮	⋮	⋮	

- This reduces the hundreds or thousands of transactions that occur daily at a specific store to a single daily transaction, and the number of data objects is reduced to the number of stores. An obvious issue is how an aggregate transaction is created; i.e., how the values of each attribute are combined across all the records corresponding to a particular location to create the aggregate transaction that represents the sales of a single store or date.
- Quantitative attributes, such as price, are typically aggregated by taking a sum or an average. A qualitative attribute, such as item, can either be omitted or summarized as the set of all the items that were sold at that location.
- The data in Table 2.7.1 can also be viewed as a multidimensional array, where each attribute is a dimension. From this viewpoint, aggregation is the process of eliminating attributes, such as the type of item, or reducing the number of values for a particular

attribute; e.g., reducing the possible values for date from 365 days to 12 months. This type of aggregation is commonly used in Online Analytical Processing (OLAP),

- There are several motivations for aggregation. First, the smaller data sets resulting from data reduction require less memory and processing time, and hence, aggregation may permit the use of more expensive data mining algorithms. Second, aggregation can act as a change ofscope or scale by providing a high-level view of the data instead of a low-level view
- A disadvantage of aggregation is the potential loss of interesting details. In the store example aggregating over months loses information about which day of the week has the highest sales.
- Example 2.7 (Australian Precipitation). This example is based on precipitation in Australia from the period 1982 to 1993. Figure 2.8(a) shows a histogram for the standard deviation of average monthly precipitation for 3,030 0.5 degree by 0.5degree grid cells in Australia, while Figure 2.8(b) shows a histogram for the standard deviation of the average yearly precipitation for the same locations. The average yearly precipitation has less variability than the average monthly precipitation. All precipitation measurements (and their standard deviations) are in centimeters.



(a) Histogram of standard deviation of average monthly precipitation

(b) Histogram of standard deviation of average yearly precipitation

Figure 2.10: Histograms of standard deviation for monthly and yearly precipitation in Australia for the period 1982 to 1993

2.7.2 Sampling

- Sampling is a commonly used approach for selecting a subset of the data objects to be analyzed. In statistics, it has long been used for both the preliminary investigation of the data and the final data analysis.
- In some cases, using a sampling algorithm can reduce the data size to the point where a better, but more expensive algorithm can be used.
- The key principle for effective sampling is the following: Using a sample will work almost as well as using the entire data set if the sample is representative. In turn, **a sample is representative** if it has approximately the same property (of interest) as the

original set of data. If the mean (average) of the data objects is the property of interest, then a sample is representative if it has a mean that is close to that of the original data.

Sampling Approaches

- There are many sampling techniques, but only a few of the most basic ones and their variations will be covered here. The simplest type of sampling is **simple random sampling**. For this type of sampling, there is an equal probability of selecting any particular item. There are two variations on random sampling (and other sampling techniques as well): (**1) sampling without replacement**- as each item is selected, it is removed from the set of all objects that together constitute the population, and **(2) sampling with replacement-objects** are not removed from the population as they are selected for the sample. In sampling with replacement, the same object can be picked more than once. The samples produced by the two methods are not much different when samples are relatively small compared to the data set size, but sampling with replacement is simpler to analyze since the probability of selecting any object remains constant during the sampling process.
- When the population consists of different types of objects, with widely different numbers of objects, simple random sampling can fail to adequately represent those types of objects that are less frequent. This can cause problems when the analysis requires proper representation of all object types. For example, when building classification models for rare classes, it is critical that the rare classes be adequately represented in the sample. Hence, a sampling scheme that can accommodate differing frequencies for the items of interest is needed. **Stratified sampling,** which starts with prespecified groups of objects, is such an approach. In the simplest version, equal numbers of objects are drawn from each group even though the groups are of different sizes. In another variation, the number of objects drawn from each group is proportional to the size of that group.
- **Example 2.8 (Sampling and Loss of Information).** Once a sampling technique has been selected, it is still necessary to choose the sample size. Larger sample sizes increase the probability that a sample will be representative, but they also eliminate much of the advantage of sampling. Conversely, with smaller sample sizes, patterns may be missed or erroneous patterns can be detected. Figure 2.11(a) shows a data set that contains 8000 two-dimensional points, while Figures 2.9(11) and 2.9(11) show samples from this data set of size 2000 and 500, respectively. Although most of the structure of this data set is present in the sample of 2000 points, much of the structure is missing in the sample of 500 points.

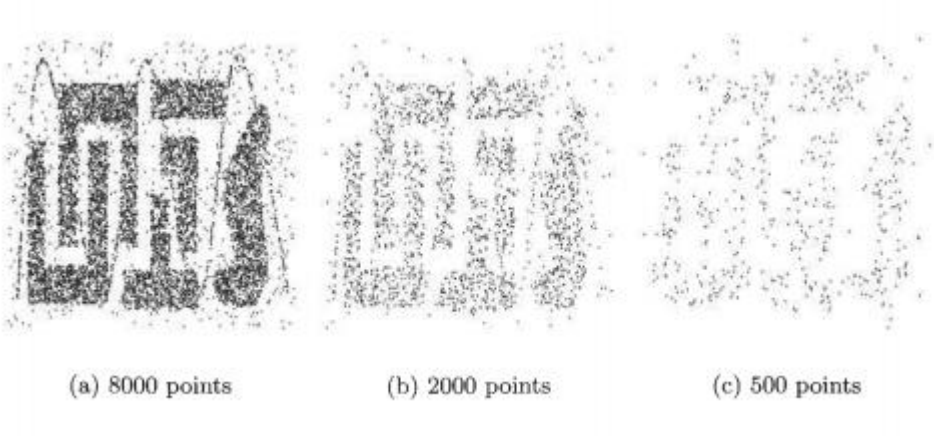
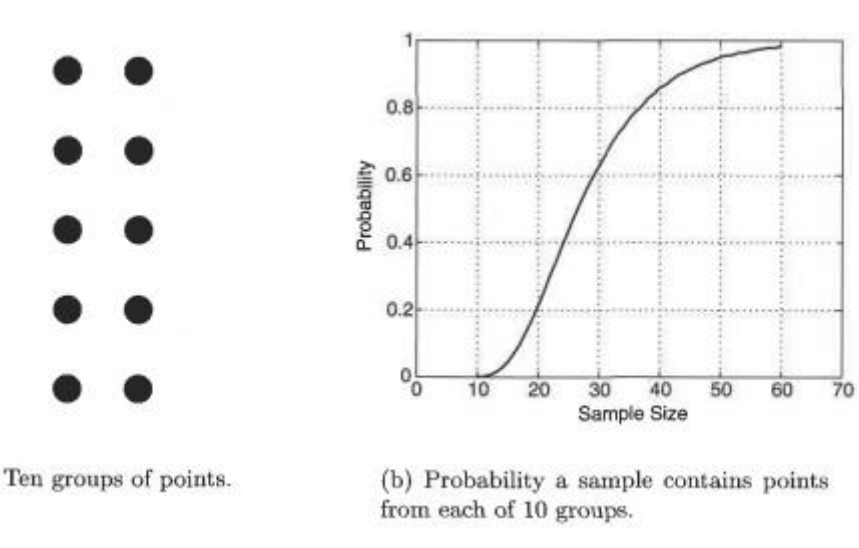


Figure 2.11 Example of the loss of structure with sampling

Example 2.9 (Determining the Proper Sample Size). To illustrate that determining the proper sample size requires a methodical approach, consider the following task. Given a set of data that consists of a small number of almost equal sized groups, find at least one representative point for each of the groups. Assume that the objects in each group are highly similar to each other, but not very similar to objects in different groups. Also assume that there are a relatively small number of groups, e.g., 10. Figure 2.12(a) shows an idealized set of clusters (groups) from which these points might be drawn. This problem can be efficiently solved using sampling. One approach is to take a small sample of data points, compute the pairwise similarities between points, and then form groups of points that are highly similar. The desired set of representative points is then obtained by taking one point from each of these groups. To follow this approach, however, we need to determine a sample size that would guarantee, with a high probability, the desired outcome; that is, that at least one point will be obtained from each cluster. Figure 2.12(b) shows the probability of getting one object from each of the 10 groups as the sample size runs from 10 to 60. Interestingly, with a sample size of 20, there is little chance (20%) of getting a sample that includes all 10 clusters. Even with a sample size of 30, there is still a moderate chance (almost 40%) of getting a sample that doesn't contain objects from all 10 clusters.



(a) Ten groups of points.

(b) Probability a sample contains points from each of 10 groups.

Figure 2.12. Finding representative points from 10 groups

Progressive Sampling

- The proper sample size can be difficult to determine, so adaptive or progressive sampling schemes are sometimes used. These approaches start with a small sample, and then increase the sample size until a sample of sufficient size has been obtained. While this technique eliminates the need to determine the correct sample size initially, it requires that there be a way to evaluate the sample to judge if it is large enough. Suppose, for instance, that progressive sampling is used to learn a predictive model. Although the accuracy of predictive models increases as the sample size increases, at some point the increase in accuracy levels off. We want to stop increasing the sample size at this leveling-off point. By keeping track of the change in accuracy of the model as we take progressively larger samples, and by taking other samples close to the size of the current one, we can get an estimate as to how close we are to this leveling-off point, and thus, stop sampling.

2.7.3 Dimensionality Reduction

- Data sets can have a large number of features. Consider a set of documents, where each document is represented by a vector whose components are the frequencies with which each word occurs in the document.
- In such cases, there are typically thousands or tens of thousands of attributes (components), one for each word in the vocabulary.
- As another example, consider a set of time series consisting of the daily closing price of various stocks over a period of 30 years. In this case, the attributes, which are the prices on specific days, again number in the thousands. There are a variety of benefits to dimensionality reduction.
- A key benefit is that many data mining algorithms work better if the dimensionality the number of attributes in the data-is lower. This is partly because dimensionality

reduction can eliminate irrelevant features and reduce noise and partly because of the curse of dimensionality, which is explained below.

- Another benefit is that a reduction of dimensionality can lead to a more understandable model because the model may involve fewer attributes.
- Also, dimensionality reduction may allow the data to be more easily visualized. Even if dimensionality reduction doesn't reduce the data to two or three dimensions, data is often visualized by looking at pairs or triplets of attributes, and the number of such combinations is greatly reduced.
- Finally, the amount of time and memory required by the data mining algorithm is reduced with a reduction in dimensionality.
- The term dimensionality reduction is often reserved for those techniques that reduce the dimensionality of a data set by creating new attributes that are a combination of the old attributes. The reduction of dimensionality by selecting new attributes that are a subset of the old is known as feature subset selection or feature selection.

The curse of dimensionality

- refers to the phenomenon that many types of data analysis become significantly harder as the dimensionality of the data increases.
- Specifically, as dimensionality increases, the data becomes increasingly sparse in the space that it occupies.
- For classification, this can mean that there are not enough data objects to allow the creation of a model that reliably assigns a class to all possible objects.
- For clustering, the definitions of density and the distance between points, which are critical for clustering, become less meaningful. As a result many clustering and classification algorithms have trouble with high-dimensional data-reduced classification accuracy and poor quality clusters.
- **Linear Algebra Techniques for Dimensionality Reduction**: Some of the most common approaches for dimensionality reduction, particularly for continuous data, use techniques from linear algebra to project the data from a high-dimensional space into a lower-dimensional space. Principal Components Analysis (PCA) is a linear algebra technique for continuous attributes that finds new attributes (principal components) that (1) are linear combinations of the original attributes, (2) are **orthogonal** (perpendicular) to each other, and (3) capture the maximum amount of variation in the data. For example, the first two principal components capture as much of the variation in the data as is possible with two orthogonal attributes that are linear combinations of the original attributes. **Singular Value Decomposition (SVD)** is a linear algebra technique that is related to PCA and is also commonly used for dimensionality reduction.

2.7.4 Feature Subset Selection

- Another way to reduce the dimensionality is to use only a subset of the features. While it might seem that such an approach would lose information, this is not the case if redundant and irrelevant features are present.

- **Redundant features** duplicate much or all of the information contained in one or more other attributes. For example, the purchase price of a product and the amount of sales tax paid contain much of the same information.
- **Irrelevant features** contain almost no useful information for the data mining task at hand. For instance, students' ID numbers are irrelevant to the task of predicting students' grade point averages.
- Redundant and irrelevant features can reduce classification accuracy and the quality of the clusters that are found. While some irrelevant and redundant attributes can be eliminated immediately by using common sense or domain knowledge, selecting the best subset of features frequently requires a systematic approach.
- The ideal approach to feature selection is to try all possible subsets of features as input to the data mining algorithm of interest, and then take the subset that produces the best results. This method has the advantage of reflecting the objective and bias of the data mining algorithm that will eventually be used.
- There are three standard approaches to feature selection: **embedded, filter, and wrapper**.
- **Embedded approaches:** Feature selection occurs naturally as part of the data mining algorithm. Specifically, during the operation of the data mining algorithm, the algorithm itself decides which attributes to use and which to ignore.
- **Filter approaches:** Features are selected before the data mining algorithm is run, using some approach that is independent of the data mining task. For example, we might select sets of attributes whose pair wise correlation is as low as possible.
- **Wrapper approaches:** These methods use the target data mining algorithm as a black box to find the best subset of attributes, in a way similar to that of the ideal algorithm described above, but typically without enumerating all possible subsets.

An Architecture for Feature Subset Selection

- It is possible to encompass both the filter and wrapper approaches within a common architecture.
- The feature selection process is viewed as consisting of four parts: a measure for evaluating a subset, a search strategy that controls the generation of a new subset of features, a stopping criterion, and a validation procedure.
- Filter methods and wrapper methods differ only in the way in which they evaluate a subset of features. For a wrapper method, subset evaluation uses the target data mining algorithm, while for a filter approach, the evaluation technique is distinct from the target data mining algorithm.
- The following discussion provides some details of this approach, which is summarized in Figure 2.13. Conceptually, feature subset selection is a search over all possible subsets of features. Many different types of search strategies can be used, but the search strategy should be computationally inexpensive and should find optimal or near optimal sets of features.
- It is usually not possible to satisfy both requirements, and thus, tradeoffs are necessary. An integral part of the search is an evaluation step to judge how the

current subset of features compares to others that have been considered. This requires an evaluation measure that attempts to determine the goodness of a subset of attributes with respect to a particular data mining task, such as classification or clustering.

- For the filter approach, such measures attempt to predict how well the actual data mining algorithm will perform on a given set of attributes. For the wrapper approach, where evaluation consists of actually running the target data mining application, the subset evaluation function is simply the criterion normally used to measure the result of the data mining. Because the number of subsets can be enormous and it is impractical to examine them all, some sort of stopping criterion is necessary.
- This strategy is usually based on one or more conditions involving the following: the number of iterations, whether the value of the subset evaluation measure is optimal or exceeds a certain threshold, whether a subset of a certain size has been obtained, whether simultaneous size and evaluation criteria have been achieved, and whether any improvement can be achieved by the options available to the search strategy.
- Finally, once a subset of features has been selected, the results of the target data mining algorithm on the selected subset should be validated. A straightforward evaluation approach is to run the algorithm with the full set of features and compare the full results to results obtained using the subset of features. Hopefully, the subset of features will produce results that are better than or almost as good as those produced when using all features. Another validation approach is to use a number of different feature selection algorithms to obtain subsets of features and then compare the results of running the data mining algorithm on each subset.

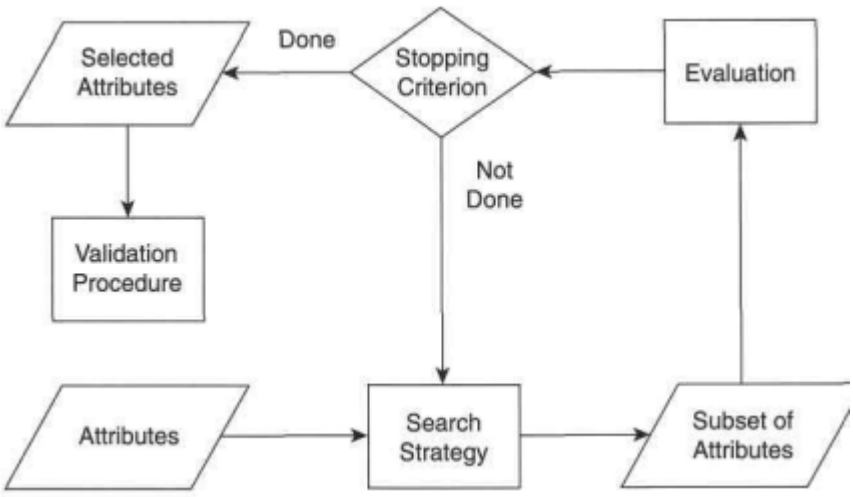


Figure 2.13: Flowchart of a feature subset selection process

Feature Weighting:

Feature weighting is an alternative to keeping or eliminating features. More important features are assigned a higher weight, while less important features are given a lower weight. These weights are sometimes assigned based on domain knowledge about the relative importance of features. Alternatively, they may be determined automatically. For example, some classification schemes, such as support vector machines, produce classification models in which each feature is given a weight. Features with larger weights play a more important role in the model. The normalization of objects that takes place when computing the cosine similarity can also be regarded as a type of feature weighting

2.7.5 Feature Creation:

- It is frequently possible to create, from the original attributes, a new set of attributes that captures the important information in a data set much more effectively.
- Furthermore, the number of new attributes can be smaller than the number of original attributes, allowing us to reap all the previously described benefits of dimensionality reduction.
- Three related methodologies for creating new attributes are described next: **feature extraction, mapping the data to a new space, and feature construction.**

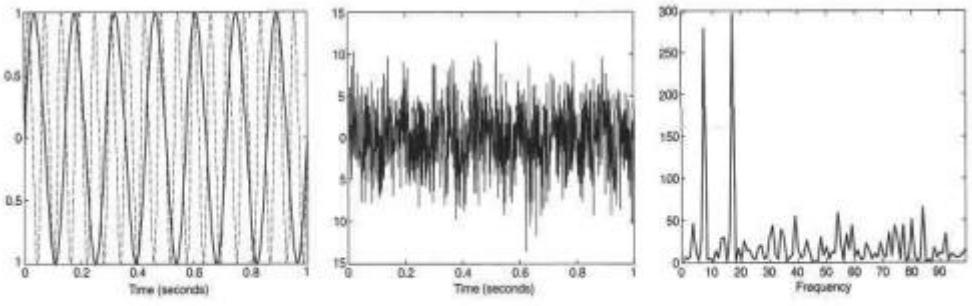
Feature Extraction

- The creation of a new set of features from the original raw data is known as feature extraction.
- Consider a set of photographs, where each photograph is to be classified according to whether or not it contains a human face. The raw data is a set of pixels, and as such, is not suitable for many types of classification algorithms. However, if the data is processed to provide higher level features, such as the presence or absence of certain types of edges and areas that are highly correlated with the presence of human faces, then a much broader set of classification techniques can be applied to this problem.
- Unfortunately, in the sense in which it is most commonly used, feature extraction is highly domain-specific. For a particular field, such as image processing, various features and the techniques to extract them have been developed over a period of time, and often these techniques have limited applicability to other fields. Consequently, whenever data mining is applied to a relatively new area, a key task is the development of new features and feature extraction methods.

Mapping the Data to a New Space

- A totally different view of the data can reveal important and interesting features. Consider, for example, time series data, which often contains periodic patterns. If there is only a single periodic pattern and not much noise' then the pattern is easily detected.
- If, on the other hand, there are a number of periodic patterns and a significant amount of noise is present, then these patterns are hard to detect. Such patterns can, nonetheless, often be detected by applying a Fourier transform to the time series in order to change to a representation in which frequency information is explicit.

- In the example that follows, it will not be necessary to know the details of the Fourier transform. It is enough to know that, for each time series, the Fourier transform produces a new data object whose attributes are related to frequencies.
- **Example 2.10 (Fourier Analysis).** The time series presented in Figure 2.14(b) is the sum of three other time series, two of which are shown in Figure 2.14(a) and have frequencies of 7 and 17 cycles per second, respectively. The third time series is random noise. Figure 2.14(c) shows the power spectrum that can be computed after applying a Fourier transform to the original time series. (Informally, the power spectrum is proportional to the square of each frequency attribute.) In spite of the noise, there are two peaks that correspond to the periods of the two original, non-noisy time series. Again, the main point is that better features can reveal important aspects of the data.



(a) Two time series. (b) Noisy time series. (c) Power spectrum

Figure **2.14: Application of the Fourier tansform to identify the underlying frequencies in time series data**

Feature Construction

- Sometimes the features in the original data sets have the necessary information, but it is not in a form suitable for the data mining algorithm. In this situation, one or more new features constructed out of the original features can be more useful than the original features.
- **Example 2.11- (Density)**. To illustrate this, consider a data set consisting of information about historical artifacts, which, along with other information, contains the volume and mass of each artifact. For simplicity, assume that these artifacts are made of a small number of materials (wood, clay, bronze, gold) and that we want to classify the artifacts with respect to the material of which they are made. In this case, a density feature constructed from the mass and volume features, i.e., density : mass/volume, would most directly yield an accurate classification. Although there have been some attempts to automatically perform feature construction by exploring simple mathematical combinations of existing attributes, the most common approach is to construct features using domain expertise.

2.7.6 Discretization and Binarization

- Algorithms that find association patterns require that the data be in the form of binary attributes. Thus, it is often necessary to transform a continuous attribute into a categorical

attribute **(discretization),** and both continuous and discrete attributes may need to be transformed into one or more binary attributes **(binarization).**

- Additionally, if a categorical attribute has a large number of values (categories), or some values occur infrequently, then it may be beneficial for certain data mining tasks to reduce the number of categories by combining some of the values.
- As with feature selection, the best discretization and binarization approach is the one that "produces the best result for the data mining algorithm that will be used to analyze the data." It is typically not practical to apply such a criterion directly. Consequently, discretization or binarization is performed in a way that satisfies a criterion that is thought to have a relationship to good performance for the data mining task being considered.

Binarization

- A simple technique to binarize a categorical attribute is the following: If there are m categorical values, then uniquely assign each original value to an integer in the interval [0,m-1]. If the attribute is ordinal, then order must be maintained by the assignment. (Note that even if the attribute is originally represented using integers, this process is necessary if the integers are not in the interval 10, m- 1] .) Next, convert each of these m integers to a binary number. Since $n = \lceil \log_2(m) \rceil$ binary digits are required to represent these integers, represent these binary numbers using n binary attributes. To illustrate, a categorical variable with 5 values {awful, poor, OK, good, great} would require three binary variables x_1,x_2 and x_3. The conversion is shown in Table 2.5.
- Such a transformation can cause complications, such as creating unintended relationships among the transformed attributes. For example, in Table 2.5, attributes x_2 and x_3 are correlated because information about the good value is encoded using both attributes.
- Furthermore, association analysis requires asymmetric binary attributes, where only the presence of the attribute (value=1) is important. For association problems, it is therefore necessary to introduce one binary attribute for each categorical value) as in Table 2.6.
- If the number of resulting attributes is too large, then the techniques described below can be used to reduce the number of categorical values before binarization.
- Likewise, for association problems, it may be necessary to replace a single binary attribute with two asymmetric binary attributes. Consider a binary attribute that records a person's gender, male or female. For traditional association rule algorithms, this information needs to be transformed into two asymmetric binary attributes, one that is a 1 only when the person is male and one that is a 1 only when the person is female.

Table 2.5. Conversion of a categorical attribute to three binary attributes.

Categorical Value	Integer Value	x_1	x_2	x_3
awful	0	0	0	0
poor	1	0	0	1
OK	2	0	1	0
good	3	0	1	1
great	4	1	0	0

Table 2.6. Conversion of a categorical attribute to five asymmetric binary attributes.

Categorical Value	Integer Value	x_1	x_2	x_3	x_4	x_5
awful	0	1	0	0	0	0
poor	1	0	1	0	0	0
OK	2	0	0	1	0	0
good	3	0	0	0	1	0
great	4	0	0	0	0	1

Discretization of Continuous Attributes

- Discretization is typically applied to attributes that are used in classification or association analysis.
- In general, the best discretization depends on the algorithm being used, as well as the other attributes being considered.
- Typically, however, the discretization of an attribute is considered in isolation. Transformation of a continuous attribute to a categorical attribute involves two subtasks: deciding how many categories to have and determining how to map the values of the continuous attribute to these categories.
- In the first step, after the values of the continuous attribute are sorted, they are then divided into n intervals by specifying n - 1 split points.
- In the second, rather trivial step, all the values in one interval are mapped to the same categorical value.
- Therefore, the problem of discretization is one of deciding how many split points to choose and where to place them
- The result can be represented either as a set of intervals $\{(x_0, x_1], (x_1, x_2], \ldots, (x_{n-1}, x_n)\}$, where x_0 and x_n may be $+\infty$ or $-\infty$ respectively, or equivalently, as a series of inequalities $x_0 < x \leq x_1, \ldots, x_{n-1} < x < x_n$.

Unsupervised Discretization

- A basic distinction between discretization methods for classification is whether class information is used (supervised) or not (unsupervised).
- If class information is not used, then relatively simple approaches are common. For instance, the equal width approach divides the range of the attribute into a user-specified number of intervals each having the same width.

- Such an approach can be badly affected by outliers, and for that reason, an equal frequency (equal depth) approach, which tries to put the same number of objects into each interval, is often preferred.
- **Example 2.12(Discretization Techniques) :** This example demonstrates how these approaches work on an actual data set. Figure 2.13(a) shows data points belonging to four different groups, along with two outliers-the large dots on either end. The techniques of the previous paragraph were applied to discretize the r values of these data points into four categorical values. (Points in the data set have a random g component to make it easy to see how many points are in each group.) Visually inspecting the data works quite well, but is not automatic, and thus, we focus on the other three approaches. The split points produced by the techniques equal width, equal frequency, and K-means are shown in Figures 2.13(b), 2.13(c), and 2.13(d), respectively. The split points are represented as dashed lines. If we measure the performance of a discretization technique by the extent to which different objects in different groups are assigned the same categorical value, then K-means performs best, followed by equal frequency, and finally, equal width.

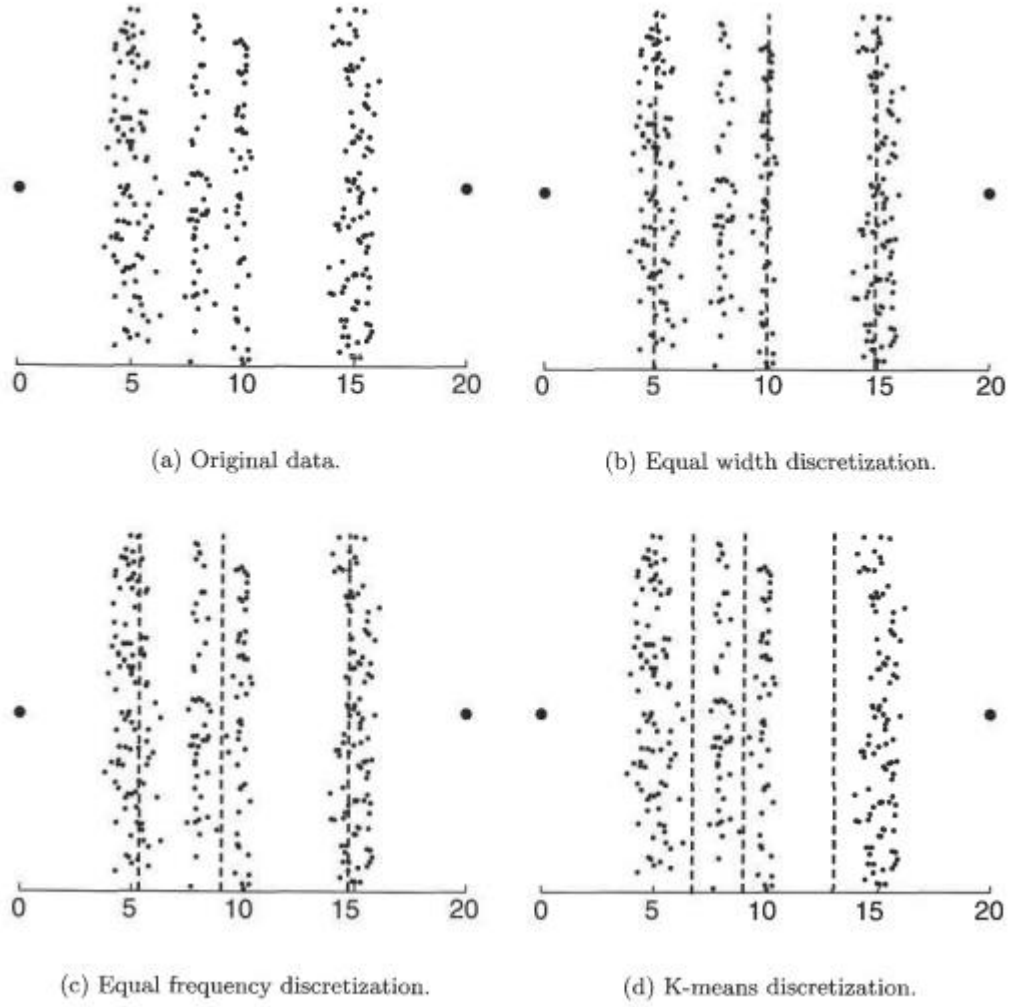


Figure 2.13. Different discretization techniques.

Supervised Discretization

- The discretization methods described above are usually better than no discretization, but keeping the end purpose in mind and using additional information (class labels) often produces better results. This should not be surprising, since an interval constructed with no knowledge of class labels often contains a mixture of class labels.
- A conceptually simple approach is to place the splits in a way that maximizes the purity of the intervals. In practice, however, such an approach requires potentially arbitrary decisions about the purity of an interval and the minimum size of an interval. To overcome such concerns, some statistically based approaches start with each attribute value as a separate interval and create larger intervals by merging adjacent intervals that are similar according to a statistical test.
- Entropy based approaches are one of the most promising approaches to discretization, and a simple approach based on entropy will be presented.

- First, it is necessary to define entropy. Let **k** be the number of different class labels, mibe the number of values in the **i^{th}** interval of a partition, and **m_{ij}** be the number of values of class j in interval i. Then the entropy ei of the 'ith interval is given by the equation

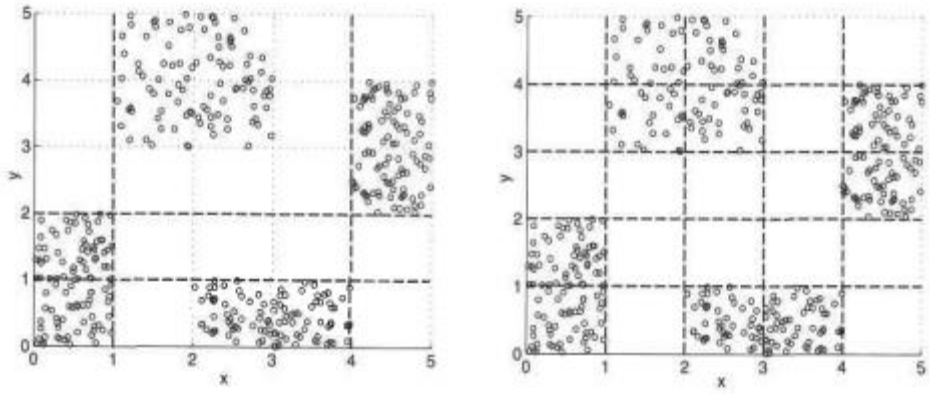
$$e_i = \sum_{i=1}^{k} p_{ij} \log_2 p_{ij},$$

where $p_{ij} = m_{ij}/m_i$ is the probability (fraction of values) of class j in the i^{th} interval. The total entropy, e, of the partition is the weighted average of the individual interval entropies, i.e.,

$$e = \sum_{i=1}^{n} w_i e_i,$$

where m is the number of values, $w_i = m_i/m$ is the fraction of values in the i^{th} interval, and n is the number of intervals. Intuitively, the entropy of an interval is a measure of the purity of an interval. If an interval contains only values of one class (is perfectly pure), then the entropy is 0 and it contributes nothing to the overall entropy.

- A simple approach for partitioning a continuous attribute starts by bisecting the initial values so that the resulting two intervals give minimum entropy. This technique only needs to consider each value as a possible split point, because it is assumed that intervals contain ordered sets of values. The splitting process is then repeated with another interval, typically choosing the interval with the worst (highest) entropy, until a user-specified number of intervals is reached, or a stopping criterion is satisfied.
- **Example 2.13 (Discretization of Two Attributes).** This method was used to independently discretize both the z and y attributes of the twodimensional data shown in Figure 2.I4. In the first discretization, shown in Figure 2.14(a), the x and y attributes were both split into three intervals. (The dashed lines indicate the split points.) In the second discretization, shown in Figure 2.14(b), the x and y attributes were both split into five intervals.
- This simple example illustrates two aspects of discretization. First, in two dimensions, the classes of points are well separated, but in one dimension, this is not so. In general, discretizing each attribute separately often guarantees suboptimal results. Second, five intervals work better than three, but six intervals do not improve the discretization much, at least in terms of entropy. (Entropy values and results for six intervals are not shown.) Consequently, it is desirable to have a stopping criterion that automatically finds the right number of partitions.



(a) Three intervals (b) Five intervals

Figure 2.14. Discretizing x and y attributes for four groups (classes) of points.

Categorical Attributes with Too Many Values

- Categorical attributes can sometimes have too many values. If the categorical attribute is an ordinal attribute, then techniques similar to those for continuous attributes can be used to reduce the number of categories. If the categorical attribute is nominal, however, then other approaches are needed. Consider a university that has a large number of departments. Consequently, a department name attribute might have dozens of different values. In this situation, we could use our knowledge of the relationships among different departments to combine departments into larger groups, such as eng'ineering, soc'ial sciences, or biological sciences. If domain knowledge does not serve as a useful guide or such an approach results in poor classification performance, then it is necessary to use a more empirical approach, such as grouping values together only if such a grouping results in improved classification accuracy or achieves some other data mining objective.

2.7.7Variable Transformation

- A variable transformation refers to a transformation that is applied to all the values of a variable. In other words, for each object, the transformation is applied to the value of the variable for that object. For example, if only the magnitude of a variable is important, then the values of the variable can be transformed by taking the absolute value. In the following section, we discuss two important types of variable transformations: simple functional transformations and normalization.

Simple Functions

- For this type of variable transformation, a simple mathematical function is applied to each value individually. If x is a variable, then examples of such transformations include x^k , log x, e^x, 1/x, sinx or mod x. In statistics, variable transformations, especially sqrt, log , and

1/x , are often used to transform data that does not have a Gaussian (normal) distribution into data that does. While this can be important, other reasons often take precedence in data mining.

- Suppose the variable of interest is the number of data bytes in a session) and the number of bytes ranges from 1 to 1 billion.
- This is a huge range, and it may be advantageous to compress it by using a $\log_{10}$ transformation.
- In this case, sessions that transferred $10^8$and 10^9 bytes would be more similar to each other than sessions that transferred 10 and 1000 bytes (9 – 8= 1 versus 3 – 1=2). For some applications, such as network intrusion detection, this may be what is desired, since the first two sessions most likely represent transfers of large files, while the latter two sessions could be two quite distinct types of sessions.

Normalization or Standardization

- Another common type of variable transformation is **the standardization or normalization** of a variable.
- The goal of standardization or normalization is to make an entire set of values have a particular property.
- To illustrate, consider comparing people based on two variables: age and income. For any two people, the difference in income will likely be much higher in absolute terms (hundreds or thousands of dollars) than the difference in age (less than 150). If the differences in the range ofvalues ofage and income are not taken into account, then the comparison between people will be dominated by differences in income. In particular, if the similarity or dissimilarity of two people is calculated using the similarity or dissimilarity measures such as that of Euclidean distance, the income values will dominate the calculation. The mean and standard deviation are strongly affected by outliers, so the above transformation is often modified.
- First, the mean is replaced by the median, i.e., the middle value. Second, the standard deviation is replaced by the absolute standard deviation.

2.8 Measures of similarity and dissimilarity (hard copy of this content will be given)

Question Bank

1. **Explain the concept of cube operator and curse of dimensionality**
2. **Explain with example bitmap and join indexing**
3. **Explain the efficient processing of OLAP Query**
4. **Explain the different OLAP servers**
5. **Define data mining and the process of KDD**
6. **Explain the different data mining tasks.**
7. **Explain the characteristics of data sets and different types of datasets**

8. **Explain the different measurement and data collection errors.**
9. **Explain the issues related to data quality**
10. **Write a short notes on data i)aggregation ii) Sampling**
11. **Explain the concept of feature subset selection briefly**
12. **Explain Discretization and binarization**
13. **Explain the three methods for feature creation**