Data Preprocessing

Content to be covered

- Data Quality,
- Major Tasks in Data Pre-processing,
- Data Reduction,
- Data Transformation and Data Discretization,
- Data Cleaning and Data Integration.

Data Quality

- Data have quality if they satisfy the requirements of the intended use.
- There are many factors comprising data quality, including accuracy, completeness, consistency, timeliness, believability, and interpretability.
- Database system may have reported errors, unusual values, and inconsistencies in the data recorded for some transactions.
- The data are
  - incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data);
  - inaccurate or noisy (containing errors, or values that deviate from the expected); and
  - inconsistent (e.g., containing discrepancies in the department codes used to categorize items).
- Inaccurate, incomplete, and inconsistent data are commonplace properties of large real-world databases and data warehouses.
- There are many possible reasons for inaccurate data.
  - The data collection instruments used may be faulty.
  - Human or computer errors occurring at data entry.
  - Users may purposely submit incorrect data values for mandatory fields. This is known as disguised missing data.
  - Errors in data transmission.
  - Technology limitations such as limited buffer size for coordinating synchronized data transfer and consumption.
  - Inconsistencies in naming conventions or data codes, or inconsistent formats for input fields (e.g., date).

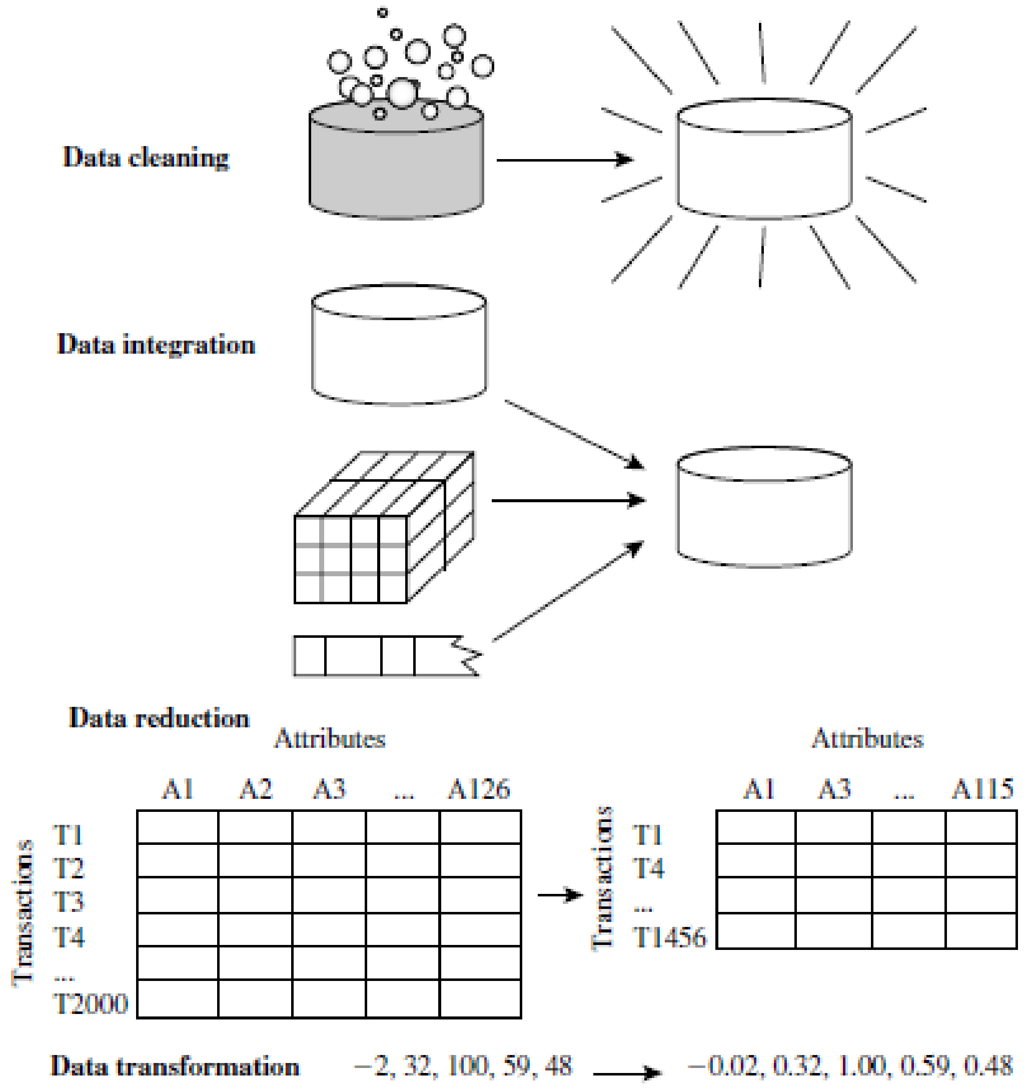
Data Quality

- Incomplete data can occur for a number of reasons.
    - Attributes of interest may not always be available, such as customer information for sales transaction data.
    - Other data may not be included simply because they were not considered important at the time of entry.
    - Relevant data may not be recorded due to a misunderstanding or because of equipment malfunctions.
    - Data that were inconsistent with other recorded data may have been deleted.
    - Furthermore, the recording of the data history or modifications may have been overlooked.
- Missing data, particularly for tuples with missing values for some attributes, may need to be inferred.
- Data quality depends on the intended use of the data. Two different users may have very different assessments of the quality of a given database.
    - For example, a marketing analyst may need to access the database mentioned before for a list of customer addresses. Some of the addresses are outdated or incorrect, yet overall, 80% of the addresses are accurate. The marketing analyst considers this to be a large customer database for target marketing purposes and is pleased with the with the database's accuracy, although, as sales manager, you found the data inaccurate.

Data Quality

- Timeliness also affects data quality.
  - For a period of time following each month, the data stored in the database are incomplete. However, once all of the data are received, it is correct.
  - The fact that the month-end data are not updated in a timely fashion has a negative impact on the data quality.
- Believability reflects how much the data are trusted by users, while interpretability reflects how easy the data are understood.
  - Suppose that a database, at one point, had several errors, all of which have since been corrected. The past errors, however, had caused many problems for sales department users, and so they no longer trust the data. The data also use many accounting codes, which the sales department does not know how to interpret. Even though the database is now accurate, complete, consistent, and timely, sales department users may regard it as of low quality due to poor believability and interpretability.

Major Tasks in Data Preprocessing



Major Tasks in Data Preprocessing

- **Data cleaning** routines work to “clean” the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- If users believe the data are dirty, they are unlikely to trust the results of any data mining that has been applied. Dirty data can cause confusion for the mining procedure, resulting in unreliable output.
- Suppose that we would like to include data from multiple sources. This would involve integrating multiple databases, data cubes, or files i.e., **Data integration**. Yet some attributes representing a given concept may have different names in different databases, causing inconsistencies and redundancies.
- Having a large amount of redundant data may slow down or confuse the knowledge discovery process.
- In addition to data cleaning, steps must be taken to help avoid redundancies during data integration.
- Typically, data cleaning and data integration are performed as a preprocessing step when preparing data for a data warehouse. Additional data cleaning can be performed to detect and remove redundancies that may have resulted from data integration.

Data Cleaning

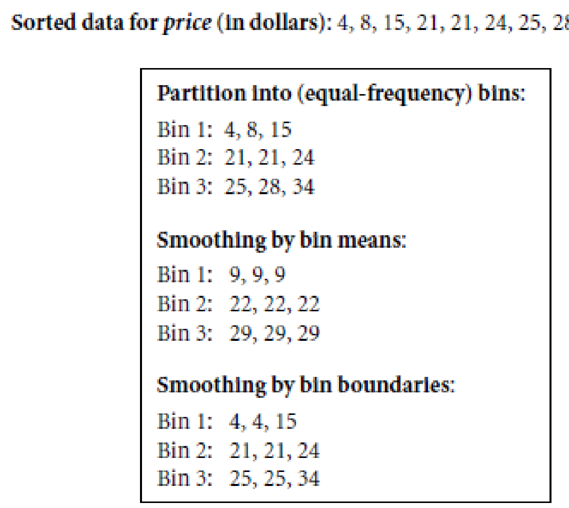
- Data cleaning (or data cleansing) routines attempt to
    - fill in missing values,
    - smooth out noise while identifying outliers, and
    - correct inconsistencies in the data.

Missing Values

- **Ignore the tuple:** This is usually done when the class label is missing. This method is not very effective, unless the tuple contains several attributes with missing values. It is especially poor when the percentage of missing values per attribute varies considerably. By ignoring the tuple, we do not make use of the remaining attributes' values in the tuple.
- **Fill in the missing value manually:** this approach is time consuming and may not be feasible given a large data set with many missing values.
- **Use a global constant to fill in the missing value:** Replace all missing attribute values by the same constant such as a label like "Unknown" or infinity. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of "Unknown."
- **Use a measure of central tendency for the attribute to fill in the missing value:** For normal (symmetric) data distributions, the mean can be used, while skewed data distribution should employ the median.
- **Use the attribute mean or median for all samples belonging to the same class as the given tuple:**
- **Use the most probable value to fill in the missing value:** This may be determined with regression, inference-based tools using a Bayesian formalism, or decision tree induction. In comparison to the other methods, it uses the most information from the present data to predict missing values.
- It is important to note that, in some cases, a missing value may not imply an error in the data. Ideally, each attribute should have one or more rules regarding the null condition. The rules may specify whether or not nulls are allowed and/or how such values should be handled or transformed. Fields may also be intentionally left blank if they are to be provided in a later step of the business process. Hence, although we can try our best to clean the data after it is seized, good database and data entry procedure design should help minimize the number of missing values or errors in the first place.

Noisy Data

- Noise is a random error or variance in a measured variable.
- Binning:
  - Binning methods smooth a sorted data value by consulting its “neighborhood,” that is, the values around it. The sorted values are distributed into a number of “buckets,” or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. Smoothing by bin means or medians can be employed, in which each bin value is replaced by the bin mean or median. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.



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

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

Smoothing by bin means:
Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

Smoothing by bin boundaries:
Bin 1: 4, 4, 15
Bin 2: 21, 21, 24
Bin 3: 25, 25, 34

Noisy Data

- Regression: Data smoothing can also be done by regression, a technique that conforms data values to a function. Linear regression involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.
- Outlier analysis: Outliers may be detected by clustering, for example, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers (Figure 3.3).
- Many data smoothing methods are also used for data discretization (a form of data transformation) and data reduction. For example, the binning techniques reduce the number of distinct values per attribute. This acts as a form of data reduction for logic-based data mining methods, such as decision tree induction, which repeatedly makes value comparisons on sorted data. Concept hierarchies can also be used for data smoothing. A concept hierarchy for price, for example, may map real price values into inexpensive, moderately priced, and expensive, thereby reducing the number of data values to be handled.

Data Cleaning as a Process

- The first step in data cleaning as a process is discrepancy detection. Discrepancies can be caused
  - poorly designed data entry forms that have many optional fields,
  - human error in data entry,
  - deliberate errors (e.g., respondents not wanting to divulge information about themselves),
  - data decay (e.g., outdated addresses).
  - inconsistent data representations and inconsistent use of codes.
  - in instrumentation devices that record data and system errors.
  - when the data are (inadequately) used for purposes other than originally intended.
  - data integration (e.g., where a given attribute can have different names in different databases).

Data Cleaning as a Process

- use any knowledge you may already have regarding properties of the data. Such knowledge or “data about data” is referred to as metadata.
    - For example, what are the data type and domain of each attribute? What are the acceptable values for each attribute?
    - Find the mean, median, and mode values. Are the data symmetric or skewed? What is the range of values? Do all values fall within the expected range? What is the standard deviation of each attribute? Values that are more than two standard deviations away from the mean for a given attribute may be flagged as potential outliers.
    - Are there any known dependencies between attributes?
    - From this, you may find noise, outliers, and unusual values that need investigation. As a data analyst, you should be on the lookout for the inconsistent use of codes and any inconsistent data representations (e.g., “2010/12/25” and “25/12/2010” for date).
- Field overloading is another error source that typically results when developers squeeze new attribute definitions into unused (bit) portions of already defined attributes (e.g., an unused bit of an attribute that has a value range that uses only, say, 31 out of 32 bits).

Data Cleaning as a Process

- The data should also be examined regarding unique rules, consecutive rules, and null rules.
  - A unique rule says that each value of the given attribute must be different from all other values for that attribute.
  - A consecutive rule says that there can be no missing values between the lowest and highest values for the attribute, and that all values must also be unique (e.g., as in check numbers).
  - A null rule specifies the use of blanks, question marks, special characters, or other strings that may indicate the null condition (e.g., where a value for a given attribute is not available), and how such values should be handled.
  - Reasons for missing values may include
    - the person originally asked to provide a value for the attribute refuses and/or finds that the information requested is not applicable (e.g., a license number attribute left blank by nondrivers);
    - the data entry person does not know the correct value; or
    - the value is to be provided by a later step of the process.
- There are a number of different commercial tools that can aid in the discrepancy detection step.
  - Data scrubbing tools use simple domain knowledge to detect errors and make corrections in the data.
  - Data auditing tools find discrepancies by analyzing the data to discover rules and relationships, and detecting data that violate such conditions.
- Some data inconsistencies may be corrected manually using external references. For example, errors made at data entry may be corrected by performing a paper trace. Most errors, however, will require data transformations. That is, once we find discrepancies, we typically need to define and apply (a series of) transformations to correct them.

Data Cleaning as a Process

- Commercial tools can assist in the data transformation step.
- Data migration tools allow simple transformations to be specified
- ETL (extraction/transformation/loading) tools allow users to specify transforms through a graphical user interface (GUI). These tools typically support only a restricted set of transforms
- The two-step process of discrepancy detection and data transformation (to correct discrepancies) iterates. This process, however, is error-prone and time consuming. Some transformations may introduce more discrepancies. Some nested discrepancies may only be detected after others have been fixed.
- Transformations are often done as a batch process while the user waits without feedback. Only after the transformation is complete can the user go back and check that no new anomalies have been mistakenly created. Typically, numerous iterations are required before the user is satisfied. Any tuples that cannot be automatically handled by a given transformation are typically written to a file without any explanation regarding the reasoning behind their failure. As a result, the entire data cleaning process also suffers from a lack of interactivity.
- New approaches to data cleaning emphasize increased interactivity.
- Another approach to increased interactivity in data cleaning is the development of declarative languages for the specification of data transformation operators. Such work focuses on defining powerful extensions to SQL and algorithms that enable users to express data cleaning specifications efficiently. As we discover more about the data, it is important to keep updating the metadata to reflect this knowledge. This will help speed up data cleaning on future versions of the same data store.

Data Cleaning as a Process

- Potter’s Wheel is a publicly available data cleaning tool that integrates discrepancy detection and transformation. Users gradually build a series of transformations by composing and debugging individual transformations, one step at a time, on a spreadsheet-like interface. The transformations can be specified graphically or by providing examples. Results are shown immediately on the records that are visible on the screen. The user can choose to undo the transformations, so that transformations that introduced additional errors can be “erased.” The tool automatically performs discrepancy checking in the background on the latest transformed view of the data. Users can gradually develop and refine transformations as discrepancies are found, leading to more effective and efficient data cleaning.

Data Integration

- The merging of data from multiple data stores. Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the subsequent data mining process.
- Methods of data integration:
  - Entity identification: How can we match schema and objects from different sources?
  - Correlation tests for numeric and nominal data.
  - Tuple duplication
  - Detection and resolution of data value conflicts.

Entity Identification Problem

- There are a number of issues to consider during data integration. Schema integration and object matching can be tricky. How can equivalent real-world entities from multiple data sources be matched up? This is referred to as the entity identification problem.
- For example, customer id in one database and cust number in another
- Examples of metadata for each attribute include the name, meaning, data type, and range of values permitted for the attribute, and null rules for handling blank, zero, or null values.
- Such metadata can be used to help avoid errors in schema integration. The metadata may also be used to help transform the data (e.g., where data codes for pay type in one database may be “H” and “S” but 1 and 2 in another). Hence, this step also relates to data cleaning, as described earlier.
- When matching attributes from one database to another during integration, special attention must be paid to the structure of the data. This is to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system. For example, in one system, a discount may be applied to the order, whereas in another system it is applied to each individual line item within the order.

Redundancy and Correlation Analysis

- Redundancy is another important issue in data integration. An attribute may be redundant if it can be “derived” from another attribute or set of attributes.
- Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set. Some redundancies can be detected by correlation analysis.
- Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data.
- For nominal data, we use the (chi-square) test. For numeric attributes, we can use the correlation coefficient and covariance, both of which access how one attribute’s values vary from those of another.

χ2 Correlation Test for Nominal Data

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r}\frac{(o_{ij}-e_{ij})^2}{e_{ij}},$$

$$e_{ij} = \frac{count(A=a_i)\times count(B=b_j)}{n},$$

	male	*female*	*Total*
fiction	250 (90)	200 (360)	450
non_fiction	50 (210)	1000 (840)	1050
Total	300	1200	1500

Note: Are *gender* and *preferred_reading* correlated?

Using Eq. (3.1) for χ^2 computation, we get

$$\chi^2 = \frac{(250-90)^2}{90}+\frac{(50-210)^2}{210}+\frac{(200-360)^2}{360}+\frac{(1000-840)^2}{840}$$
$$= 284.44+121.90+71.11+30.48 = 507.93.$$

Correlation Coefficient for Numeric Data

- If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other. Scatter plots can also be used to view correlations between attributes

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B},$$

Correlation Coefficient for Numeric Data

- Note that correlation does not imply causality. That is, if A and B are correlated, this does not necessarily imply that A causes B or that B causes A. For example, in analyzing a demographic database, we may find that attributes representing the number of hospitals and the number of car thefts in a region are correlated. This does not mean that one causes the other. Both are actually causally linked to a third attribute, namely, population.

Covariance for Numeric Data

- In probability theory and statistics, correlation and covariance are two similar measures for assessing how much two attributes change together.

$$Cov(A,B) = E((A-\bar{A})(B-\bar{B})) = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n}.$$

- If A and B are independent the covariance is 0.
- However, the converse is not true. Some pairs of random variables (attributes) may have a covariance of 0 but are not independent.

Tuple Duplication

- In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level (e.g., where there are two or more identical tuples for a given unique data entry case).
- The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy.
- Inconsistencies often arise between various duplicates, due to inaccurate data entry or updating some but not all data occurrences.
- For example, if a purchase order database contains attributes for the purchaser's name and address instead of a key to this information in a purchaser database, discrepancies can occur, such as the same purchaser's name appearing with different addresses within the purchase order database.

Data Value Conflict Detection and Resolution

- Data integration also involves the detection and resolution of data value conflicts. For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding. For instance, a weight attribute may be stored in metric units in one system and British imperial units in another.
- For a hotel chain, the price of rooms in different cities may involve not only different currencies but also different services (e.g., free breakfast) and taxes.
- When exchanging information between schools, for example, each school may have its own curriculum and grading scheme.
- Attributes may also differ on the abstraction level, where an attribute in one system is recorded at, say, a lower abstraction level than the “same” attribute in another. For example, the total sales in one database may refer to one branch of a company, while an attribute of the same name in another database may refer to the total sales for company stores in a given region.

Data Reduction

- Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data.
- That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results.
- Data reduction strategies include dimensionality reduction, numerosity reduction, and data compression.
- Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. Dimensionality reduction methods include wavelet transforms and principal components analysis, which transform or project the original data onto a smaller space. Attribute subset selection is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed.

Data Reduction

- Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or nonparametric.
- For parametric methods, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Regression and log-linear models are examples.
- Nonparametric methods for storing reduced representations of the data include histograms, clustering, sampling, and data cube aggregation.
- In data compression, transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be reconstructed from the compressed data without any information loss, the data reduction is called lossless. If, instead, we can reconstruct only an approximation of the original data, then the data reduction is called lossy. There are several lossless algorithms for string compression; however, they typically allow only limited data manipulation. Dimensionality reduction and numerosity reduction techniques can also be considered forms of data compression.

Wavelet Transforms

- The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector X, transforms it to a numerically different vector, X0, of wavelet coefficients. The two vectors are of the same length. When applying this technique to data reduction, we consider each tuple as an n-dimensional data vector, depicting n measurements made on the tuple from n database attributes.
- Wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients. For example, all wavelet coefficients larger than some user-specified threshold can be retained. All other coefficients are set to 0. The resulting data representation is therefore very sparse, so that operations that can take advantage of data sparsity are computationally very fast if performed in wavelet space.
- The technique also works to remove noise without smoothing out the main features of the data, making it effective for data cleaning as well. Given a set of coefficients, an approximation of the original data can be constructed by applying the inverse of the DWT used.

Wavelet Transforms

- The DWT is closely related to the discrete Fourier transform (DFT), a signal processing technique involving sines and cosines.
- DWT achieves better lossy compression. Hence, for an equivalent approximation, the DWT requires less space than the DFT. Unlike the DFT, wavelets are quite localized in space, contributing to the conservation of local detail.
- Popular wavelet transforms include the Haar-2, Daubechies-4, and Daubechies-6. The general procedure for applying a discrete wavelet transform uses a hierarchical pyramid algorithm that halves the data at each iteration, resulting in fast computational speed. The method is as follows:

Wavelet Transforms

- The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros as necessary (L >= n).
- Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.
- The two functions are applied to pairs of data points in X, that is, to all pairs of measurements (x2i ,x2i+1). This results in two data sets of length L=2. In general, these represent a smoothed or low-frequency version of the input data and the high frequency content of it, respectively.
- The two functions are recursively applied to the data sets obtained in the previous loop, until the resulting data sets obtained are of length 2.
- Selected values from the data sets obtained in the previous iterations are designated the wavelet coefficients of the transformed data.

Wavelet Transforms

- Equivalently, a matrix multiplication can be applied to the input data in order to obtain the wavelet coefficients, where the matrix used depends on the given DWT. The matrix must be orthonormal, meaning that the columns are unit vectors and are mutually orthogonal, so that the matrix inverse is just its transpose. This property allows the reconstruction of the data from the smooth and smooth-difference data sets. By factoring the matrix used into a product of a few sparse matrices, the resulting "fast DWT" algorithm has a complexity of O(n) for an input vector of length n.
- Wavelet transforms can be applied to multidimensional data such as a data cube. This is done by first applying the transform to the first dimension, then to the second, and so on. The computational complexity involved is linear with respect to the number of cells in the cube. Wavelet transforms give good results on sparse or skewed data and on data with ordered attributes. Lossy compression by wavelets is reportedly better than JPEG compression, the current commercial standard. Wavelet transforms have many real world applications, including the compression of fingerprint images, computer vision, analysis of time-series data, and data cleaning.

Principal Components Analysis

- Suppose that the data to be reduced consist of tuples or data vectors described by n attributes or dimensions. Principal components analysis (PCA; also called the Karhunen-Loeve, or K-L, method) searches for k n-dimensional orthogonal vectors that can best be used to represent the data, where k <= n. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction. Unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA “combines” the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set. PCA often reveals relationships that were not previously suspected and thereby allows interpretations that would not ordinarily result.

Principal Components Analysis

- The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.
- PCA computes k orthonormal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the principal components. The input data are a linear combination of the principal components.
- The principal components are sorted in order of decreasing “significance” or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. This information helps identify groups or patterns within the data.
- Because the components are sorted in decreasing order of “significance,” the data size can be reduced by eliminating the weaker components, that is, those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

Principal Components Analysis

- PCA can be applied to ordered and unordered attributes, and can handle sparse data and skewed data.
- Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions.
- Principal components may be used as inputs to multiple regression and cluster analysis.
- In comparison with wavelet transforms, PCA tends to be better at handling sparse data, whereas wavelet transforms are more suitable for data of high dimensionality.

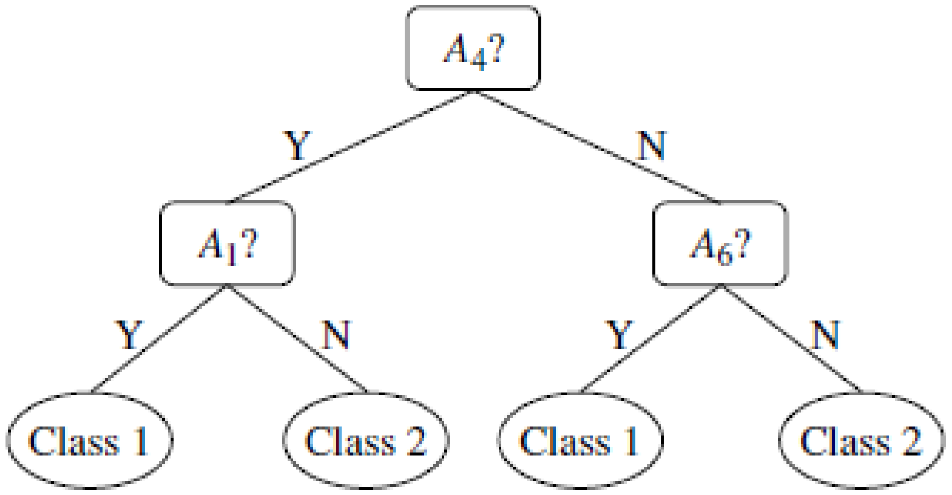
Attribute Subset Selection

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

Attribute Subset Selection

- Attribute subset selection reduces the data set size by removing irrelevant or redundant attributes (or dimensions). The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Mining on a reduced set of attributes has an additional benefit: It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.
- For n attributes, there are power(2,n) possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as n and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.
- The “best” (and “worst”) attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used such as the information gain measure used in building decision trees for classification.

Attribute Subset Selection

Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  A_4? Y N A_1? A_6? Y N Y N Class 1 Class 2 Class 1 Class 2 => Reduced attribute set: $\{A_1, A_4, A_6\}$

Attribute Subset Selection

- Stepwise forward selection: The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.
- Stepwise backward elimination: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
- Combination of forward selection and backward elimination: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
- Decision tree induction: Decision tree algorithms (e.g., ID3, C4.5, and CART) were originally intended for classification. Decision tree induction constructs a flowchart like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes. When decision tree induction is used for attribute subset selection, a tree is constructed From the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes.
- The stopping criteria for the methods may vary. The procedure may employ a threshold on the measure used to determine when to stop the attribute selection process.
- In some cases, we may want to create new attributes based on others. Such attribute construction6 can help improve accuracy and understanding of structure in high dimensional data. For example, we may wish to add the attribute area based on the attributes height and width. By combining attributes, attribute construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

Regression and Log-Linear Models: Parametric Data Reduction

- Regression and log-linear models can be used to approximate the given data. In (simple) linear regression, the data are modeled to fit a straight line. For example, a random variable, y (called a response variable), can be modeled as a linear function of another random variable, x (called a predictor variable), with the equation

$$y = wx + b$$

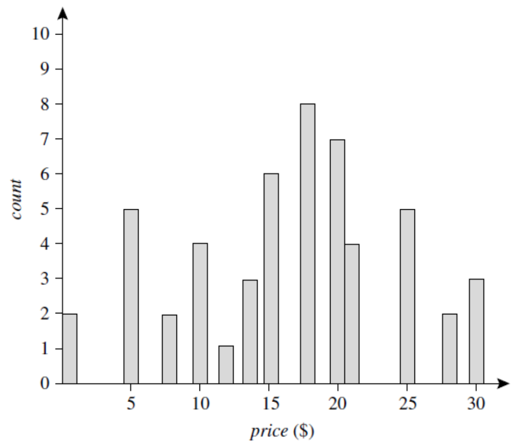
- where the variance of y is assumed to be constant. In the context of data mining, x and y are numeric database attributes. The coefficients, w and b (called regression coefficients), specify the slope of the line and the y-intercept, respectively. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line. Multiple linear regression is an extension of (simple) linear regression, which allows a response variable, y, to be modeled as a linear function of two or more predictor variables.

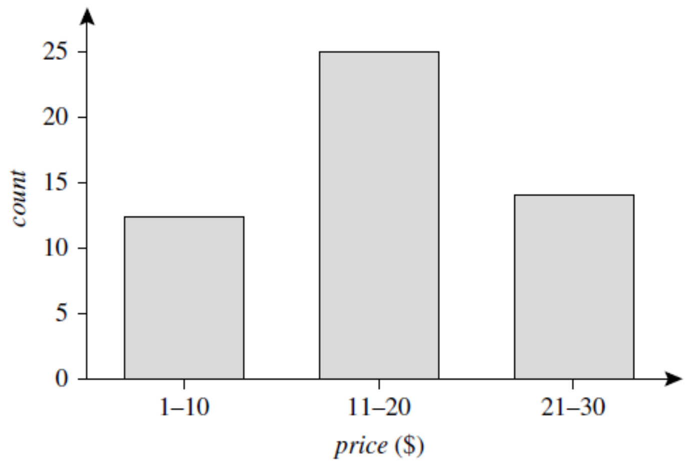
Regression and Log-Linear Models: Parametric Data Reduction

- Log-linear models approximate discrete multidimensional probability distributions. Given a set of tuples in n dimensions (e.g., described by n attributes), we can consider each tuple as a point in an n-dimensional space.
- Log-linear models can be used to estimate the probability of each point in a multidimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations. This allows a higher-dimensional data space to be constructed from lower-dimensional spaces.
- Log-linear models are therefore also useful for dimensionality reduction and data smoothing.
- Regression and log-linear models can both be used on sparse data, although their application may be limited. While both methods can handle skewed data, regression does exceptionally well. Regression can be computationally intensive when applied to high-dimensional data, whereas log-linear models show good scalability for up to 10 or so dimensions.

Histograms

- Histograms use binning to approximate data distributions and are a popular form of data reduction.
- A histogram for an attribute, A, partitions the data distribution of A into disjoint subsets, referred to as buckets or bins. If each bucket represents only a single attribute–value/frequency pair, the buckets are called singleton buckets.
- Often, buckets instead represent continuous ranges for the given attribute.





Histograms

- “How are the buckets determined and the attribute values partitioned?”
  - Equal-width: In an equal-width histogram, the width of each bucket range is uniform.
  - Equal-frequency (or equal-depth): In an equal-frequency histogram, the buckets are created so that, roughly, the frequency of each bucket is constant.
- Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data.
- Multidimensional histograms can capture dependencies between attributes. These histograms have been found effective in approximating data with up to five attributes.
- Singleton buckets are useful for storing high-frequency outliers.

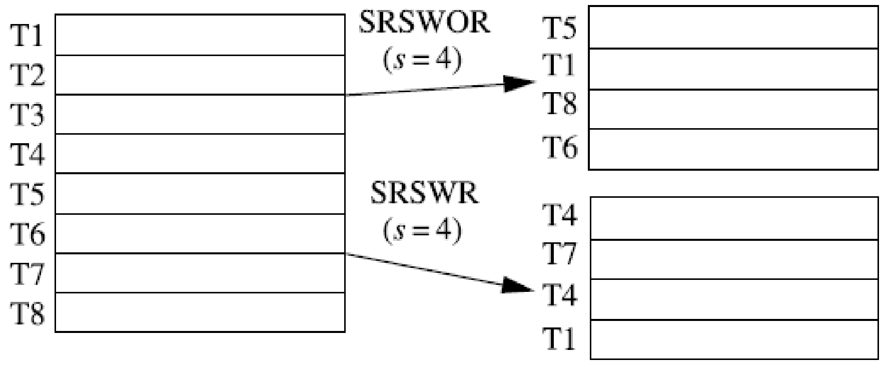
Clustering

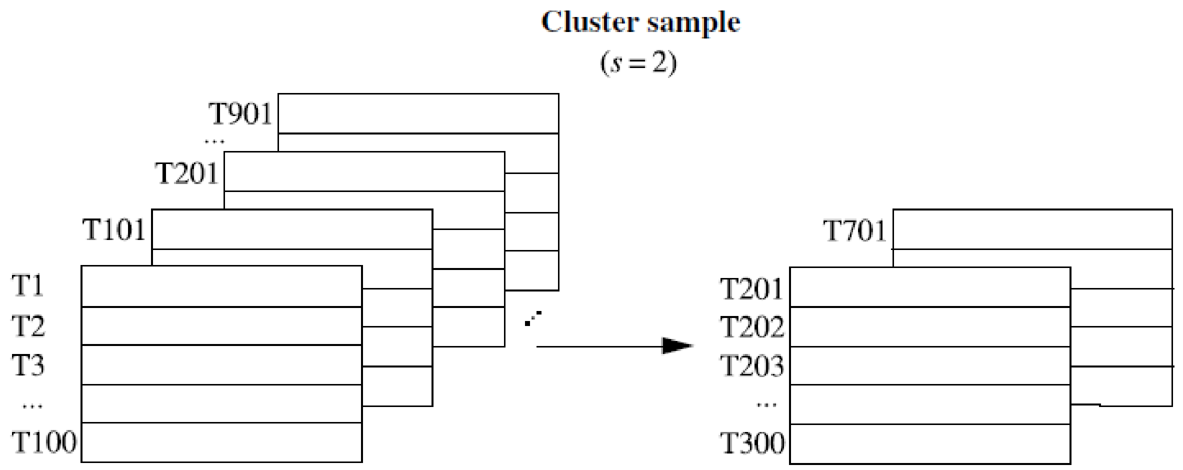
- Clustering techniques consider data tuples as objects. They partition the objects into groups, or clusters, so that objects within a cluster are “similar” to one another and “dissimilar” to objects in other clusters.
- Similarity is commonly defined in terms of how “close” the objects are in space, based on a distance function. The “quality” of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster.
- Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid (denoting the “average object,” or average point in space for the cluster).
- In data reduction, the cluster representations of the data are used to replace the actual data. The effectiveness of this technique depends on the data’s nature. It is much more effective for data that can be organized into distinct clusters than for smeared data. There are many measures for defining clusters and cluster quality.

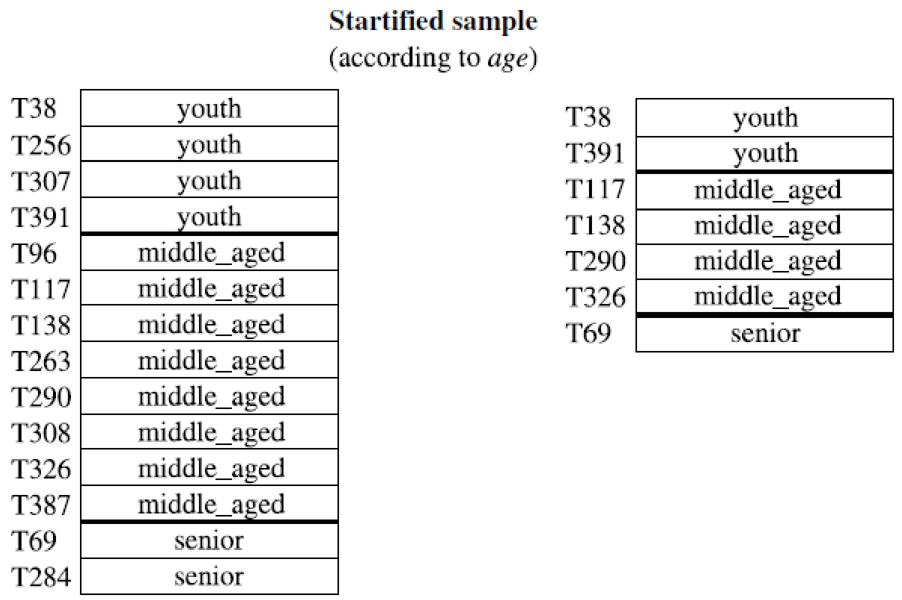
Sampling

- Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random data sample (or subset). Suppose that a large data set, D, contains N tuples.
  - Simple random sample without replacement (SRSWOR) of size s: This is created by drawing s of the N tuples from D (s < N), where the probability of drawing any tuple in D is 1/N.
  - Simple random sample with replacement (SRSWR) of size s: This is similar to SRSWOR, except that each time a tuple is drawn from D, it is recorded and then replaced. That is, after a tuple is drawn, it is placed back in D so that it may be drawn again.
  - Cluster sample: If the tuples in D are grouped into M mutually disjoint "clusters," then an SRS of s clusters can be obtained, where s < M. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples. Other clustering criteria conveying rich semantics can also be explored. For example, in a spatial database, we may choose to define clusters geographically based on how closely different areas are located.
  - Stratified sample: If D is divided into mutually disjoint parts called strata, a stratified sample of D is generated by obtaining an SRS at each stratum. This helps ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer age group. In this way, the age group having the smallest number of customers will be sure to be represented.

Sampling






Startified sample
(according to *age*)

T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

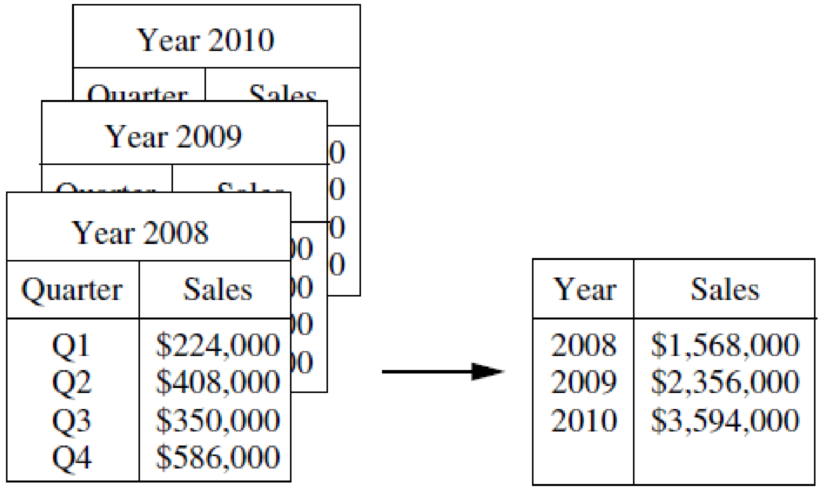
T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior

Sampling

- An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, s, as opposed to N, the data set size.
- Other data reduction techniques can require at least one complete pass through D. For a fixed sample size, sampling complexity increases only linearly as the number of data dimensions, n, increases.
- When applied to data reduction, sampling is most commonly used to estimate the answer to an aggregate query. It is possible to determine a sufficient sample size for estimating a given function within a specified degree of error. This sample size, s, may be extremely small in comparison to N.
- Sampling is a natural choice for the progressive refinement of a reduced data set. Such a set can be further refined by simply increasing the sample size.

Data Cube Aggregation

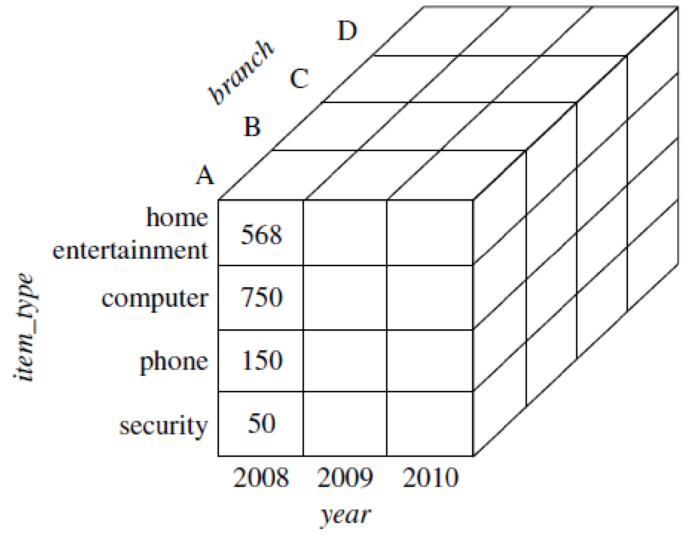
- Imagine that you have collected the data for your analysis. These data consist of the sales per quarter, for the years 2008 to 2010. You are, however, interested in the annual sales (total per year), rather than the total per quarter. Thus, the data can be aggregated so that the resulting data summarize the total sales per year instead of per quarter.


Year 2010
Year 2009

Year 2008

Quarter	Sales
Q1	$224,000
Q2	$408,000
Q3	$350,000
Q4	$586,000

Year	Sales
2008	$1,568,000
2009	$2,356,000
2010	$3,594,000



- The resulting data set is smaller in volume, without loss of information necessary for the analysis task.

Data Cube Aggregation

- Data cubes store multidimensional aggregated information. Each cell holds an aggregate data value, corresponding to the data point in multidimensional space. Concept hierarchies may exist for each attribute, allowing the analysis of data at multiple abstraction levels. For example, a hierarchy for branch could allow branches to be grouped into regions, based on their address.
- Data cubes provide fast access to precomputed, summarized data, thereby benefiting online analytical processing as well as data mining.
- The cube created at the lowest abstraction level is referred to as the base cuboid. The base cuboid should correspond to an individual entity of interest such as sales or customer. In other words, the lowest level should be usable, or useful for the analysis. A cube at the highest level of abstraction is the apex cuboid. For the sales data, the apex cuboid would give one total—the total sales for all three years, for all item types, and for all branches.
- Data cubes created for varying levels of abstraction are often referred to as cuboids, so that a data cube may instead refer to a lattice of cuboids. Each higher abstraction level further reduces the resulting data size. When replying to data mining requests, the smallest available cuboid relevant to the given task should be used.

Data Transformation and Data Discretization

- Data Transformation Strategies Overview
  - Smoothing, which works to remove noise from the data. Techniques include binning, regression, and clustering.
  - Attribute construction (or feature construction), where new attributes are constructed and added from the given set of attributes to help the mining process.
  - Aggregation, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.
  - Normalization, where the attribute data are scaled so as to fall within a smaller range.
  - Discretization, where the raw values of a numeric attribute (e.g., age) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., youth, adult, senior). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a concept hierarchy for the numeric attribute. More than one concept hierarchy can be defined for the same attribute to accommodate the needs of various users.
  - Concept hierarchy generation for nominal data, where attributes such as street can be generalized to higher-level concepts, like city or country. Many hierarchies for nominal attributes are implicit within the database schema and can be automatically defined at the schema definition level.

Data Transformation and Data Discretization

- Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up).
- If the discretization process uses class information, then we say it is supervised discretization. Otherwise, it is unsupervised.
- If the process starts by first finding one or a few points (called split points or cut points) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called top-down discretization or splitting. This contrasts with bottom-up discretization or merging, which starts by considering all of the continuous values as potential split-points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals.
- Data discretization and concept hierarchy generation are also forms of data reduction. The raw data are replaced by a smaller number of interval or concept labels. This simplifies the original data and makes the mining more efficient. The resulting patterns mined are typically easier to understand. Concept hierarchies are also useful for mining at multiple abstraction levels.

Data Transformation by Normalization

- The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to very different results.
- In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or “weight.”
- To help avoid dependence on the choice of measurement units, the data should be normalized or standardized. This involves transforming the data to fall within a smaller or common range such as [-1, 1] or [0.0, 1.0].
- Normalizing the data attempts to give all attributes an equal weight. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering.
- If using the neural network backpropagation algorithm for classification mining, normalizing the input values for each attribute measured in the training tuples will help speed up the learning phase.
- For distance-based methods, normalization helps prevent attributes with initially large ranges (e.g., income) from outweighing attributes with initially smaller ranges (e.g., binary attributes). It is also useful when given no prior knowledge of the data.

Data Transformation by Normalization

- Min-max normalization performs a linear transformation on the original data. Suppose that minA and maxA are the minimum and maximum values of an attribute, A Min-max normalization maps a value, vi of A to Vi in the range [new minA, new maxA] by computing

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

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

Data Transformation by Normalization

- In z-score normalization (or zero-mean normalization), the values for an attribute, A, are normalized based on the mean (i.e., average) and standard deviation of A. A value, vi , of A is normalized to Vi

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A},$$

A

- This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.

Data Transformation by Normalization

- In z-score normalization (or zero-mean normalization), the values for an attribute, A, are normalized based on the mean (i.e., average) and standard deviation of A. A value, vi , of A is normalized to Vi

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A},$$

- This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization.
- A variation of this z-score normalization replaces the standard deviation by the mean absolute deviation of A. The mean absolute deviation of A, denoted sA, is

$$s_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + \cdots + |v_n - \bar{A}|).$$

- The mean absolute deviation is more robust to outliers than the standard deviation. When computing the mean absolute deviation, the deviations from the mean are not squared; hence, the effect of outliers is somewhat reduced.

Data Transformation by Normalization

- Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A. A value, vi , of A is normalized to Vi by computing $$v'_i = \frac{v_i}{10^j},$$

- Note that normalization can change the original data quite a bit, especially when using z-score normalization or decimal scaling. It is also necessary to save the normalization parameters (e.g., the mean and standard deviation if using z-score normalization) so that future data can be normalized in a uniform manner.

Discretization by Binning

- Binning is a top-down splitting technique based on a specified number of bins.
- These methods are used as discretization methods for data reduction and concept hierarchy generation. For example, attribute values can be discretized by applying equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in smoothing by bin means or smoothing by bin medians, respectively.
- These techniques can be applied recursively to the resulting partitions to generate concept hierarchies.
- Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers.

Discretization by Histogram Analysis

- Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information.
- A histogram partitions the values of an attribute, A, into disjoint ranges called buckets or bins.
- In an equal-width histogram, the values are partitioned into equal-size partitions
- With an equal-frequency histogram, the values are partitioned so that, ideally, each partition contains the same number of data tuples.
- The histogram analysis algorithm can be applied recursively to each partition in order to automatically generate a multilevel concept hierarchy, with the procedure terminating once a prespecified number of concept levels has been reached. A minimum interval size can also be used per level to control the recursive procedure. This specifies the minimum width of a partition, or the minimum number of values for each partition at each level.
- Histograms can also be partitioned based on cluster analysis of the data distribution.

Discretization by Cluster, Decision Tree, and Correlation Analyses

- Cluster analysis is a popular data discretization method.
- A clustering algorithm can be applied to discretize a numeric attribute, A, by partitioning the values of A into clusters or groups.
- Clustering takes the distribution of A into consideration, as well as the closeness of data points, and therefore is able to produce high-quality discretization results.
- Clustering can be used to generate a concept hierarchy for A by following either a top-down splitting strategy or a bottom-up merging strategy, where each cluster forms a node of the concept hierarchy. In the former, each initial cluster or partition may be further decomposed into several subclusters, forming a lower level of the hierarchy. In the latter, clusters are formed by repeatedly grouping neighboring clusters in order to form higher-level concepts.

Discretization by Cluster, Decision Tree, and Correlation Analyses

- Techniques to generate decision trees can be applied to discretization. Such techniques employ a top-down splitting approach.
- Decision tree approaches to discretization are supervised, that is, they make use of class label information. For example, we may have a data set of patient symptoms (the attributes) where each patient has an associated diagnosis class label. Class distribution information is used in the calculation and determination of split-points (data values for partitioning an attribute range). Intuitively, the main idea is to select split-points so that a given resulting partition contains as many tuples of the same class as possible. Entropy is the most commonly used measure for this purpose. To discretize a numeric attribute, A, the method selects the value of A that has the minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Such discretization forms a concept hierarchy for A.
- Because decision tree–based discretization uses class information, it is more likely that the interval boundaries (split-points) are defined to occur in places that may help improve classification accuracy.

Discretization by Cluster, Decision Tree, and Correlation Analyses

- Measures of correlation can be used for discretization. ChiMerge is a ChiSquare-based discretization method.
- ChiMerge, employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively.
- ChiMerge is supervised in that it uses class information. The basic notion is that for accurate discretization, the relative class frequencies should be fairly consistent within an interval. Therefore, if two adjacent intervals have a very similar distribution of classes, then the intervals can be merged. Otherwise, they should remain separate.
- ChiMerge proceeds as follows.
  - Each distinct value of a numeric attribute A is considered to be one interval.
  - ChiSquare tests are performed for every pair of adjacent intervals.
  - Adjacent intervals with the least ChiSquare values are merged together, because low ChiSquare values for a pair indicate similar class distributions.
  - This merging process proceeds recursively until a predefined stopping criterion is met.

Concept Hierarchy Generation for Nominal Data

- Nominal attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include geographic location, job category, and item type.
- Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert.
- Fortunately, many hierarchies are implicit within the database schema and can be automatically defined at the schema definition level. The concept hierarchies can be used to transform the data into multiple levels of granularity.
- For example, data mining patterns regarding sales may be found relating to specific regions or countries, in addition to individual branch locations.

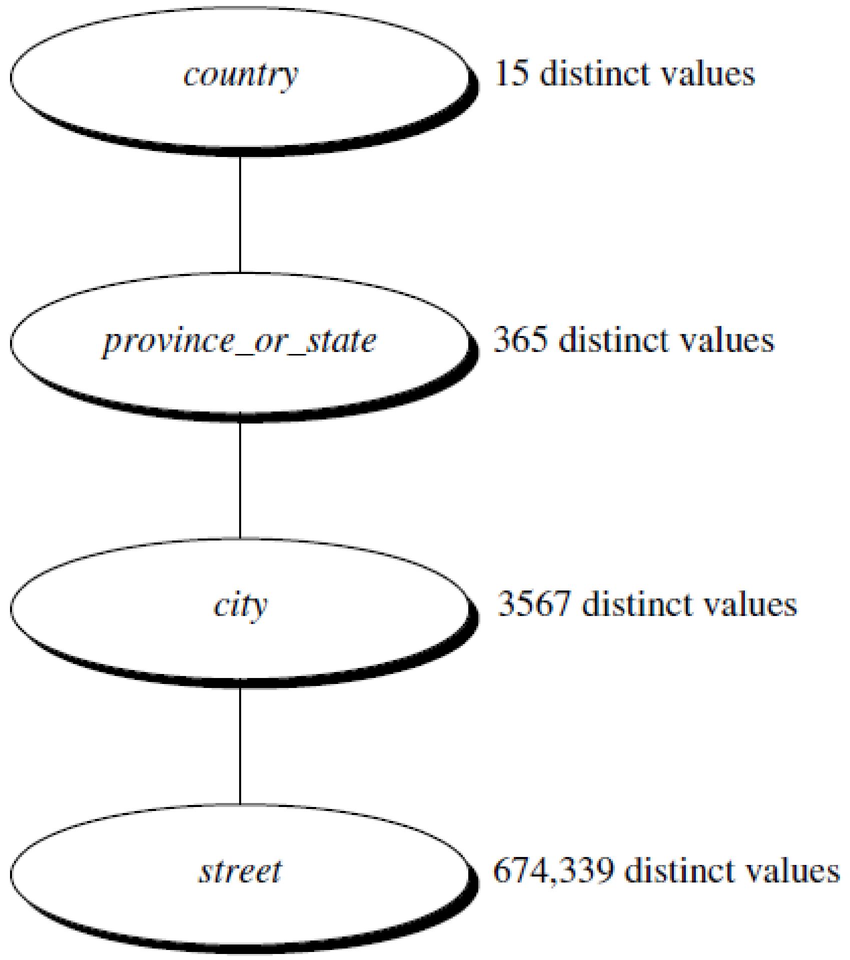
Concept Hierarchy Generation for Nominal Data

- Specification of a partial ordering of attributes explicitly at the schema level by users or experts: Concept hierarchies for nominal attributes or dimensions typically involve a group of attributes. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level.
    - For example, suppose that a relational database contains the following group of attributes: street, city, province or state, and country. Similarly, a data warehouse location dimension may contain the same attributes. A hierarchy can be defined by specifying the total ordering among these attributes at the schema level such as street < city <province or state < country.
- Specification of a portion of a hierarchy by explicit data grouping: This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it is unrealistic to define an entire concept hierarchy by explicit value enumeration. On the contrary, we can easily specify explicit groupings for a small portion of intermediate-level data. For example, after specifying that province and country form a hierarchy at the schema level, a user could define some intermediate levels manually.

Concept Hierarchy Generation for Nominal Data

- Specification of a set of attributes, but not of their partial ordering:
  - A user may specify a set of attributes forming a concept hierarchy, but omit to explicitly state their partial ordering. The system can then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy.
  - Consider the observation that since higher-level concepts generally cover several subordinate lower-level concepts, an attribute defining a high concept level (e.g., country) will usually contain a smaller number of distinct values than an attribute defining a lower concept level (e.g., street). Based on this observation, a concept hierarchy can be automatically generated based on the number of distinct values per attribute in the given attribute set. The attribute with the most distinct values is placed at the lowest hierarchy level. The lower the number of distinct values an attribute has, the higher it is in the generated concept hierarchy. This heuristic rule works well in many cases. Some local-level swapping or adjustments may be applied by users or experts, when necessary, after examination of the generated hierarchy.
  - Note that this heuristic rule is not foolproof. For example, a time dimension in a database may contain 20 distinct years, 12 distinct months, and 7 distinct days of the week. However, this does not suggest that the time hierarchy should be “year <month < days of the week,” with days of the week at the top of the hierarchy.

Concept Hierarchy Generation for Nominal Data



Concept Hierarchy Generation for Nominal Data

- Specification of only a partial set of attributes: Sometimes a user can be careless when defining a hierarchy, or have only a vague idea about what should be included in a hierarchy. Consequently, the user may have included only a small subset of the relevant attributes in the hierarchy specification.
- For example, instead of including all of the hierarchically relevant attributes for location, the user may have specified only street and city. To handle such partially specified hierarchies, it is important to embed data semantics in the database schema so that attributes with tight semantic connections can be pinned together. In this way, the specification of one attribute may trigger a whole group of semantically tightly linked attributes to be “dragged in” to forma complete hierarchy. Users, however, should have the option to override this feature, as necessary.
- In summary, information at the schema level and on attribute–value counts can be used to generate concept hierarchies for nominal data. Transforming nominal data with the use of concept hierarchies allows higher-level knowledge patterns to be found. It allows mining at multiple levels of abstraction, which is a common requirement for data mining applications.