
Pre-processing: An Overview

- Real-world data tend to be incorrect (inaccurate or noisy or dirty), incomplete (missing), and inconsistent due to their typically huge size and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results.
- Process of transforming the raw data into appropriate, useful and efficient format for subsequent analysis is called data preprocessing.
- Data Preprocessing prepares raw data for further processing.
- Data processing techniques substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

Why Preprocess the Data?

(Need of Data Pre-processing)

- Real world data tend to be inaccurate (incorrect or noisy or dirty), incomplete and inconsistent.
 - *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
 - *inconsistent*: containing discrepancies in the department codes used to categorize items
- Data preprocessing techniques improve the quality, accuracy and efficiency of the subsequent mining.
- Data preprocessing is important step in the knowledge discovery process because quality decision must base on quality data.
- Detecting data anomalies, rectifying them early and reducing the data to be analyzed can lead to huge payoffs for decision making.

Major Tasks in Data Pre-processing

(Data Pre-processing Techniques (or) Forms of data Pre-processing)

- The major steps involved in data preprocessing are:
 - Data Cleaning
 - Data Integration
 - Data Reduction
 - Data Transformation.
 - *Data cleaning* can be applied to remove noise and correct inconsistencies in data. *Data integration* merges data from multiple sources into a coherent data store such as a data warehouse. *Data reduction* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. *Data transformations* (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0.
-

- **Data cleaning** routines work to “clean” the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- **Data integration** is 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.
- **Data reduction** obtains a reduced representation of the data set that is much smaller in volume, yet produces the same (or almost the same) analytical results.
- Data reduction strategies include *dimensionality reduction* and *numerosity reduction*.
 - In **dimensionality reduction**, data encoding schemes are applied so as to obtain a reduced or “compressed” representation of the original data. Examples include data compression techniques (e.g., *wavelet transforms* and *principal components analysis*), *attribute subset selection* (e.g., removing irrelevant attributes), and *attribute construction* (e.g., where a small set of more useful attributes is derived from the original set).
 - In **numerosity reduction**, the data are replaced by alternative, smaller representations using parametric models (e.g., *regression* or *log-linear models*) or nonparametric models (e.g., *histograms*, *clusters*, *sampling*, or *data aggregation*).
- **Data Transformation:** In data transformation, the data are transformed or consolidated into forms appropriate for mining. In this preprocessing step, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand.

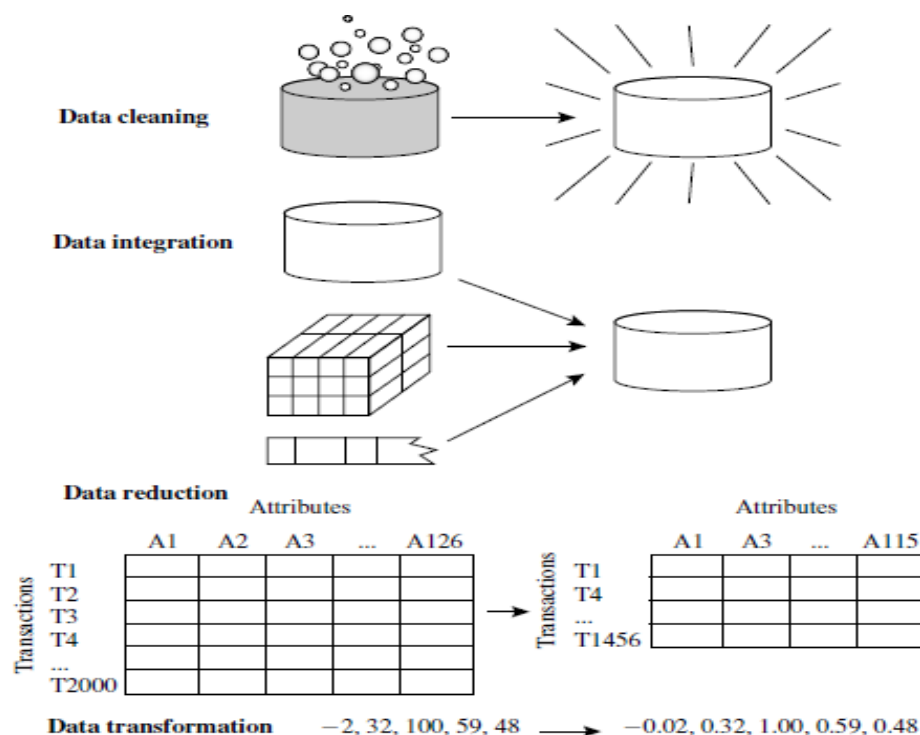


Figure 3.1 Forms of data preprocessing.

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.
- Basic methods for data cleaning are.
 - ❖ Handling missing values
 - ❖ Data smoothing technique

Handling Missing Values

- No recorded value for attribute of tuple is called missing value. 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 non drivers)
 - the data entry person does not know the correct value
 - the value is to be provided by a later step of the process.
 - **Methods for filling in the missing values:**
 - **Ignore the tuple:** This is usually done when the class label is missing (assuming the mining task involves classification). 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.
 - **Fill in the missing value manually:** In general, this approach is time consuming and may not be feasible given a large data set with many missing values.
 - **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 If missing values are replaced by, say, “*Unknown*,” then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of “*Unknown*.” Hence, although this method is simple, it is not foolproof.
 - **Use a measure of central tendency for the attribute (e.g., the mean or median) 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:** For example, if classifying customers according to *credit risk*, we may replace the missing value with the mean *income* value for customers in the same credit risk category as that of the given tuple. If the data distribution for a given class is skewed, the median value is a better choice.
-

-
- **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.

Noisy Data

- **Noise** is a random error or variance in a measured variable.
- We can “smooth” out the data to remove the noise.
- Data smoothing techniques.
 - Binning
 - Regression
 - Outlier Analysis

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.
- In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin
- **Smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median.
- In **smoothing by bin boundaries**, the minimum and maximum values in a given bin are identified as the *bin boundaries*. Each bin value is then replaced by the closest boundary value.

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

Figure 3.2 Binning methods for data smoothing.

- Above figure illustrates some binning techniques. In this example, the data for *price* are first sorted and then partitioned into *equal-frequency* bins of size 3 (i.e., each bin contains three values). In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin. For example, the mean of the values 4, 8, and 15 in Bin 1 is 9. Therefore, each original value in this bin is replaced by the value 9.
-

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 regressions* 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.

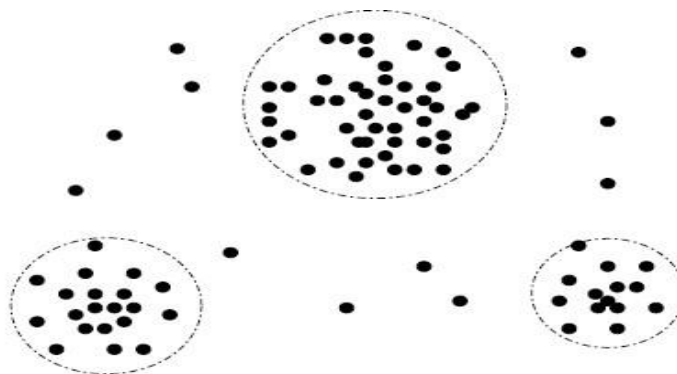


Figure 3.3 A 2-D customer data plot with respect to customer locations in a city, showing three data clusters. Outliers may be detected as values that fall outside of the cluster sets.

Data Integration

- **Data integration** combines data from multiple sources to form a coherent data store.
 - 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.
 - The resolution of semantic heterogeneity, metadata, correlation analysis, tuple duplication detection, and data conflict detection contribute to smooth data integration.
 - Issues that must be considered during such integration include:
 - **Schema integration:** The metadata from the different data sources must be integrated in order to match up equivalent real-world entities. This is referred to as the entity identification problem.
 - **Handling redundant data:** Derived attributes may be redundant, and inconsistent attribute naming may also lead to redundancies in the resulting data set. Some redundancies can be detected by **correlation analysis**
-

-
- **Tuple Duplication Detection:** Duplications at the tuple level may occur and thus need to be detected and resolved.
 - **Detection and resolution of data value conflicts:** Differences in representation, scaling, or encoding may cause the same real-world entity attribute values to differ in the data sources being integrated.

Data Transformation

→ In data transformation, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand. i.e.; in *data transformation*, the data are transformed or consolidated into forms appropriate.

Data Transformation Strategies:

Strategies for data transformation include the following:

1. **Smoothing**, which works to remove noise from the data. Techniques include binning, regression, and clustering.
2. **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.
3. **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.
4. **Normalization**, where the attribute data are scaled so as to fall within a smaller range, such as -1.0 to 1.0, or 0.0 to 1.0.
5. **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*).
6. **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 by Normalization

- Normalization 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. It is also useful when given no prior knowledge of the data.
-

Various methods for data normalization are:

- ❖ Min-max normalization
- ❖ z-score normalization using the standard deviation
- ❖ Normalization by decimal scaling.

Let A be a numeric attribute with n observed values, v_1, v_2, \dots, v_n .

- **Min-max normalization** performs a linear transformation on the original data. Suppose that $\min A$ and $\max A$ are the minimum and maximum values of an attribute, A . Min-max normalization maps a value, v_i , of A to v_i' in the range $[\text{new_min}_A, \text{new_max}_A]$ by computing

$$v_i' = \frac{v_i - \min_A}{\max_A - \min_A} (\text{new_max}_A - \text{new_min}_A) + \text{new_min}_A.$$

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 .

Example 3.4 Min-max normalization. Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range $[0.0, 1.0]$. By min-max normalization, a value of \$73,600 for *income* is transformed to $\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$. ■

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

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

Where σ_A and \bar{A} are the mean and standard deviation, respectively, of attribute A .

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

Example 3.5 z-score normalization. Suppose that the mean and standard deviation of the values for the attribute *income* are \$54,000 and \$16,000, respectively. With z-score normalization, a value of \$73,600 for *income* is transformed to $\frac{73,600 - 54,000}{16,000} = 1.225$. ■

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

$$v'_i = \frac{v_i}{10^j},$$

where j is the smallest integer such that $\max(|v'_i|) < 1$. i.e., number of digits in maximum absolute value of A.

Example: Suppose that the recorded values of A range from -986 to 917. The Maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1,000 (i.e., $j = 3$) so that -986 normalizes to -0.986 and 917 normalizes to 0.917.

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* can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering.

Data Reduction Strategies

- **Dimensionality reduction** reduces the number of random variables or attributes under consideration. Methods include *wavelet transforms*, *principal components analysis*, *attribute subset selection*, and *attribute creation*.
 - **Numerosity reduction** methods use parametric or non parametric models to obtain smaller representations of the original data. Parametric models store only the model parameters instead of the actual data. Examples include regression and log-linear models. Non parametric methods include histograms, clustering, sampling, and data cube aggregation.
 - **Data compression** methods apply transformations to obtain a reduced or “compressed” representation of the original data. The data reduction is lossless if the original data can be reconstructed from the compressed data without any loss of information; otherwise, it is lossy.
-

Data Reduction Strategies:

- **Dimensionality Reduction**

- Wavelet transforms
- Principal components analysis
- Attribute subset selection

- **Numerosity Reduction**

- Parametric methods
 - Regression
 - Log-linear models.
- Non parametric methods
 - Histograms
 - Clustering
 - Sampling
 - Data cube aggregation.

- **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, X' , of **wavelet coefficients**. The two vectors are of the same length.
- The usefulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients.
- The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines
- 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.
- 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 (PCA)

- 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 $k \leq n$. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction.

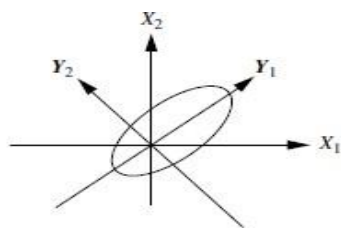


Figure 3.5 Principal components analysis. Y_1 and Y_2 are the first two principal components for the given data.

The basic procedure is as follows:

1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.
 2. PCA computes k orthogonal vectors that provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the *principal components*. The input data are a linear combination of the principal components.
 3. The principal components are sorted in order of decreasing “significance” or strength. The principal components essentially serve as a new set of axes for
-

the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, below Figure shows the first two principal components, Y_1 and Y_2 , for the given set of data originally mapped to the axes X_1 and X_2 . This information helps identify groups or patterns within the data.

4. 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.
 - 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.

Attribute Subset Selection

- Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant.
- **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.
- Basic heuristic methods of attribute subset selection include the following:

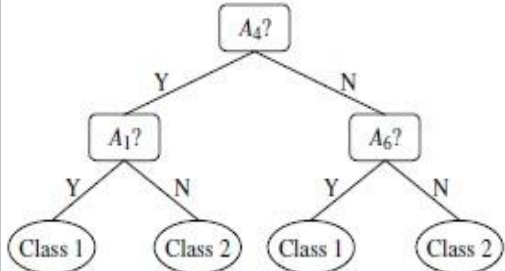
Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Initial reduced set: $\{\}$ $\Rightarrow \{A_1\}$ $\Rightarrow \{A_1, A_4\}$ \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_4, A_5, A_6\}$ \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  \Rightarrow Reduced attribute set: $\{A_1, A_4, A_6\}$

Figure 3.6 Greedy (heuristic) methods for attribute subset selection.

1. **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.

-
2. **Stepwise backward elimination:** The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
 3. **Combination of forward selection and backward elimination:** The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.
 4. **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 (non leaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the “best” attribute to partition the data into individual classes. 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.

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.
- **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

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.

- **Example:** The following data are a list of prices for commonly sold items (rounded to the nearest dollar). The numbers have been sorted: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 28, 28, 30, 30, 30.
- Figure 3.7 shows a histogram for the data using singleton buckets. To further reduce the data, it is common to have each bucket denote a continuous value range for the given attribute. In Figure 3.8, each bucket represents a different \$10 range for *price*.

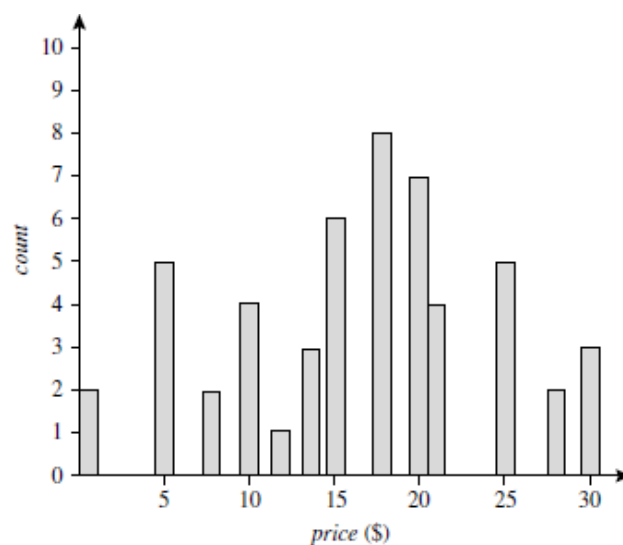


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

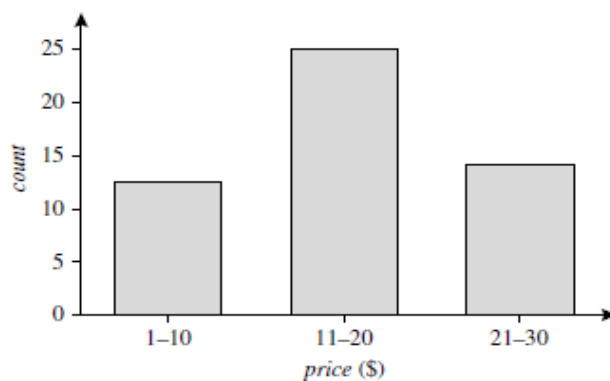


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of \$10.

There are several partitioning rules, including the following:

- ❖ **Equal-width:** In an equal-width histogram, the width of each bucket range is uniform (e.g., the width of \$10 for the buckets in Figure 3.8).
 - ❖ **Equal-frequency**(or equal-depth): In an equal-frequency histogram, the buckets are created so that, roughly, the frequency of each bucket is constant (i.e., each bucket contains roughly the same number of contiguous data samples).
- Histograms are highly effective at approximating both sparse and dense data, as well as highly skewed and uniform data.
 - The histograms described before for single attributes can be extended for multiple attributes.
 - *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.

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. The most common ways that we could sample D for data reduction, as illustrated in following figure.
-

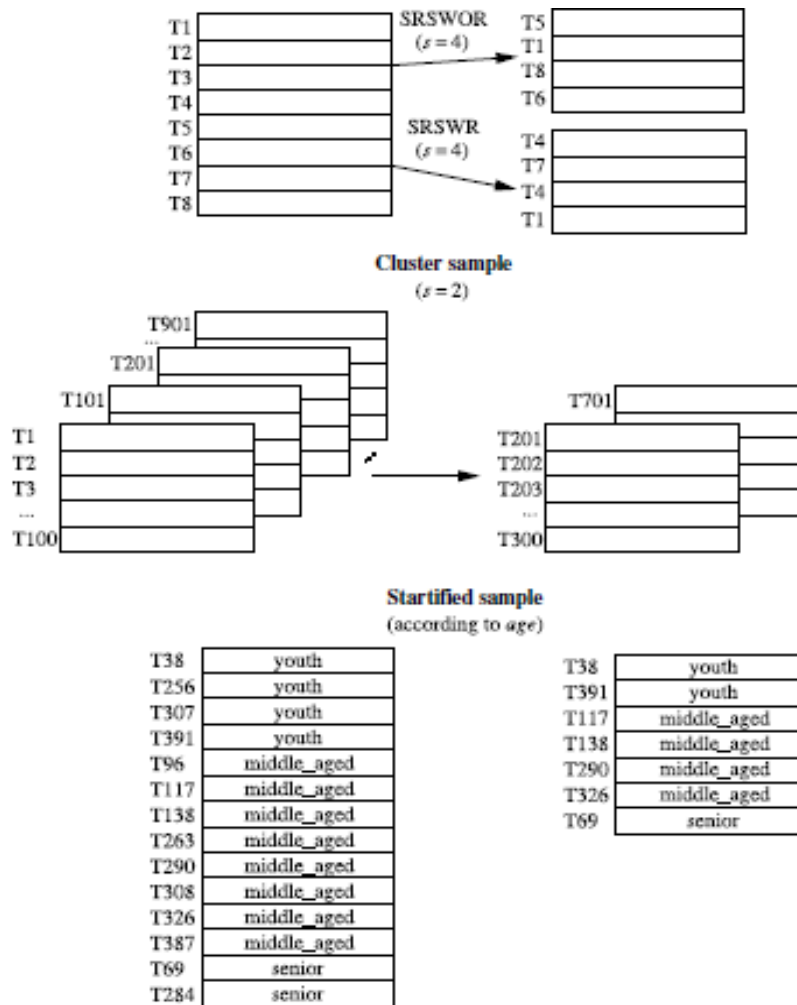


Figure 3.9 Sampling can be used for data reduction.

Simple random sample without replacement (SRSWOR):

→ This is created by drawing s of the N tuples from D ($s < N$), where the probability of drawing any tuple in D is $1/N$, that is, all tuples are equally likely to be sampled.

Simple random sample with replacement(SRSWR) :

→ 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 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.

Data Cube Aggregation

- Data cubes store multidimensional aggregated information. Each cell holds an aggregate data value, corresponding to the data point multidimensional space.

The diagram illustrates the aggregation of quarterly sales data. On the left, three stacked tables represent quarterly data for the years 2008, 2009, and 2010. Each table has columns for 'Quarter' and 'Sales'. The 2008 table shows specific sales values for each quarter, while the 2009 and 2010 tables show placeholder values (000000). An arrow points from these quarterly tables to a table on the right, which shows the aggregated annual sales for each year.

Year 2010	
Quarter	Sales
Q1	000000
Q2	000000
Q3	000000
Q4	000000

Year 2009	
Quarter	Sales
Q1	000000
Q2	000000
Q3	000000
Q4	000000

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

Figure 3.10 Sales data for a given branch of *AllElectronics* for the years 2008 through 2010. On the left, the sales are shown per quarter. On the right, the data are aggregated to provide the annual sales.

For example, Figure 3.11 shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch.

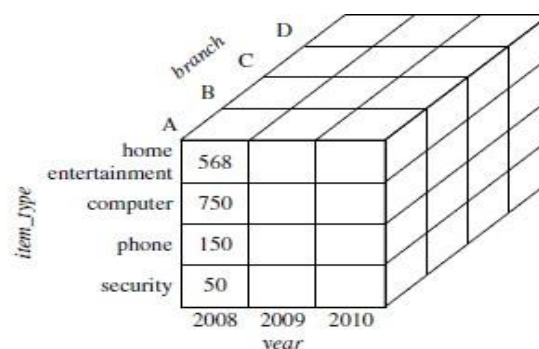


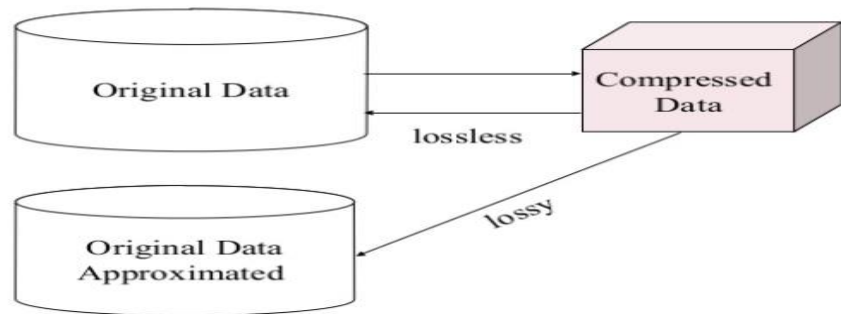
Figure 3.11 A data cube for sales at *AllElectronics*.

- 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 in Figure 3.11, 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*.
-

Data Compression

- **Data Compression** specifically refers to a **data reduction** method by which files are shrunk at the bit level.
- **Data Compression** works by using formulas or algorithms to **reduce** the number of bits needed to represent the **data**.
- The data reduction is *lossless* if the original data can be reconstructed from the compressed data without any loss of information; otherwise, it is *lossy*.

Data Compression



String Compression:

- There are extensive theories & well-tuned algorithms
- Typically lossless
- But only limited manipulation is possible

Audio/video Compression:

- Typically lossy compression with progressive refinement
- Some time small fragments of signal can be constructed without reconstruction of the whole.
- Dimensionality Reduction and Numerosity Reduction techniques can also be consider form of data compression.

Data Discretization and Concept Hierarchy Generation for Numerical Data

- **Data discretization** transforms numeric data by mapping values to interval or concept labels. Such methods can be used to automatically generate *concept hierarchies* for the data, which allows for mining at multiple levels of granularity.
 - 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.

- A concept hierarchy for a given numeric attribute defines a discretization of the attribute. i.e organize attributes or attributes values into different level of abstraction.
- *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.
- Concept hierarchies can be used to reduce the data collecting and replacing low-level concepts(such as numeric value for the attribute age) by higher level concepts (such as young, middle-aged, or senior).
- Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert.
- Several discretization methods can be used to automatically generate or dynamically refine concept hierarchies for numerical attributes.
- Concept hierarchy generation is also form of data reduction.
- Discretization techniques include:
 - ❖ Binning
 - ❖ Histogram Analysis
 - ❖ Cluster Analysis
 - ❖ Decision Tree Analysis
 - ❖ Correlation Analysis.

Discretization by Binning:

- Binning is a top-down splitting technique based on a specified number of bins.
- Binning methods for data smoothing are also used as discretization methods for data reduction and concept hierarchy generation.
- For example, attribute values can be discretized by applyig 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.

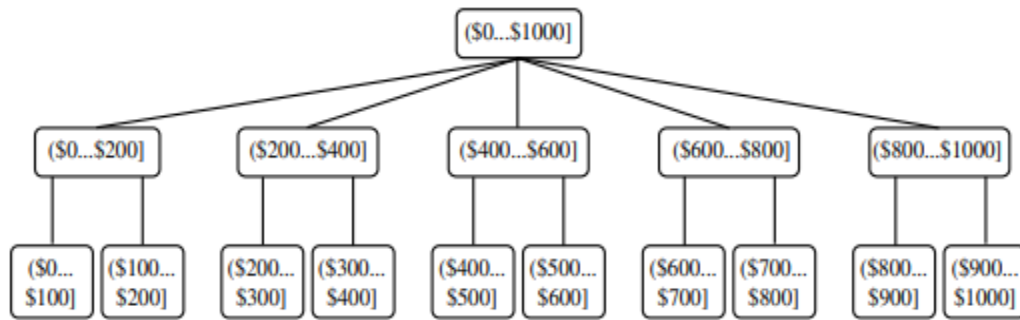
Discretization by Histogram Analysis:

- 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*. Various partitioning rules can be used to define histograms.
-

-
- In an *equal-width* histogram, for example, the values are partitioned into equal-size partitions or ranges.
 - 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 pre specified 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, as described next.

Discretization by Cluster, Decision Tree, and Correlation Analyses:

- Clustering, decision tree analysis, and correlation analysis can be used for data discretization.
 - 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.
 - Techniques to generate decision trees for classification can be applied to discretization. Such techniques employ a top-down splitting approach.
 - Measures of correlation can be used for discretization. *ChiMerge* is χ^2 - based discretization method. This method employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively.
-



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

Concept Hierarchy Generation for Categorical Data

- Nominal (categorical) 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*.
- In concept hierarchy, attributes such as *street* can be generalized to higher-level concepts, like *city* or *country*.
- 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.
- 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.
- For example, data mining patterns regarding sales may be found relating to specific regions or countries, in addition to individual branch locations

Methods for the generation of concept hierarchies for nominal data:

1.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*.

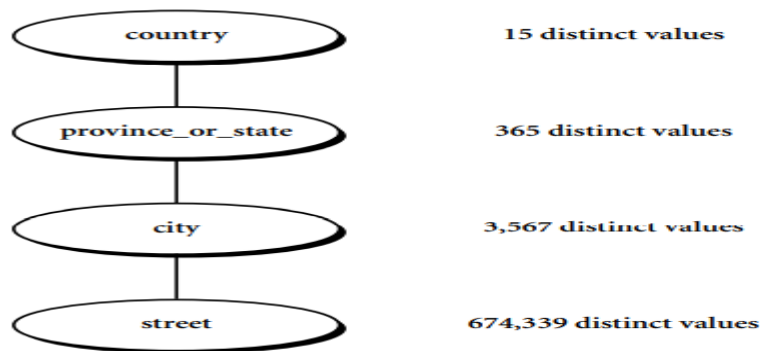
2.Specification of a portion of a hierarchy by explicit data grouping: This is essentially the manual definition of a portion of a concept hierarchy. In a large database, it is unrealistic to define an entire concept hierarchy by explicit value enumeration. On the contrary, we can easily specify explicit 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, such as “{*Alberta, Saskatchewan, Manitoba*} □ *prairies- Canada*” and “{*British Columbia, prairies-Canada*} □ *Western -Canada*.”

3.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.

Example 3.7 Concept hierarchy generation based on the number of distinct values per attribute.

Suppose a user selects a set of location-oriented attributes—*street*, *country*, *province_or_state*, and *city*—from the *AllElectronics* database, but does not specify the hierarchical ordering among the attributes.

A concept hierarchy for *location* can be generated automatically, as illustrated in Figure 3.13. First, sort the attributes in ascending order based on the number of distinct values in each attribute. This results in the following (where the number of distinct values per attribute is shown in parentheses): *country* (15), *province_or_state* (365), *city* (3567), and *street* (674,339). Second, generate the hierarchy from the top down according to the sorted order, with the first attribute at the top level and the last attribute at the bottom level. Finally, the user can examine the generated hierarchy, and when necessary, modify it to reflect desired semantic relationships among the attributes. In this example, it is obvious that there is no need to modify the generated hierarchy. ■



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

- 4. 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 form a complete hierarchy.