Module 2

Data Preprocessing-Need of data preprocessing, Data Cleaning- Missing values, Noisy data, Data Integration and Transformation, Data Reduction-Data cube aggregation, Attribute subset selection, Dimensionality reduction, Numerosity reduction, Discretization and concept hierarchy generation.

Data Preprocessing:

Today's real-world databases are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size and their likely origin from multiple, heterogenous sources. **Low-quality data will lead to low-quality mining results**.

There are several data preprocessing techniques.

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. This can improve the accuracy and efficiency of mining algorithms involving distance measurements.

These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a date field to a common format.

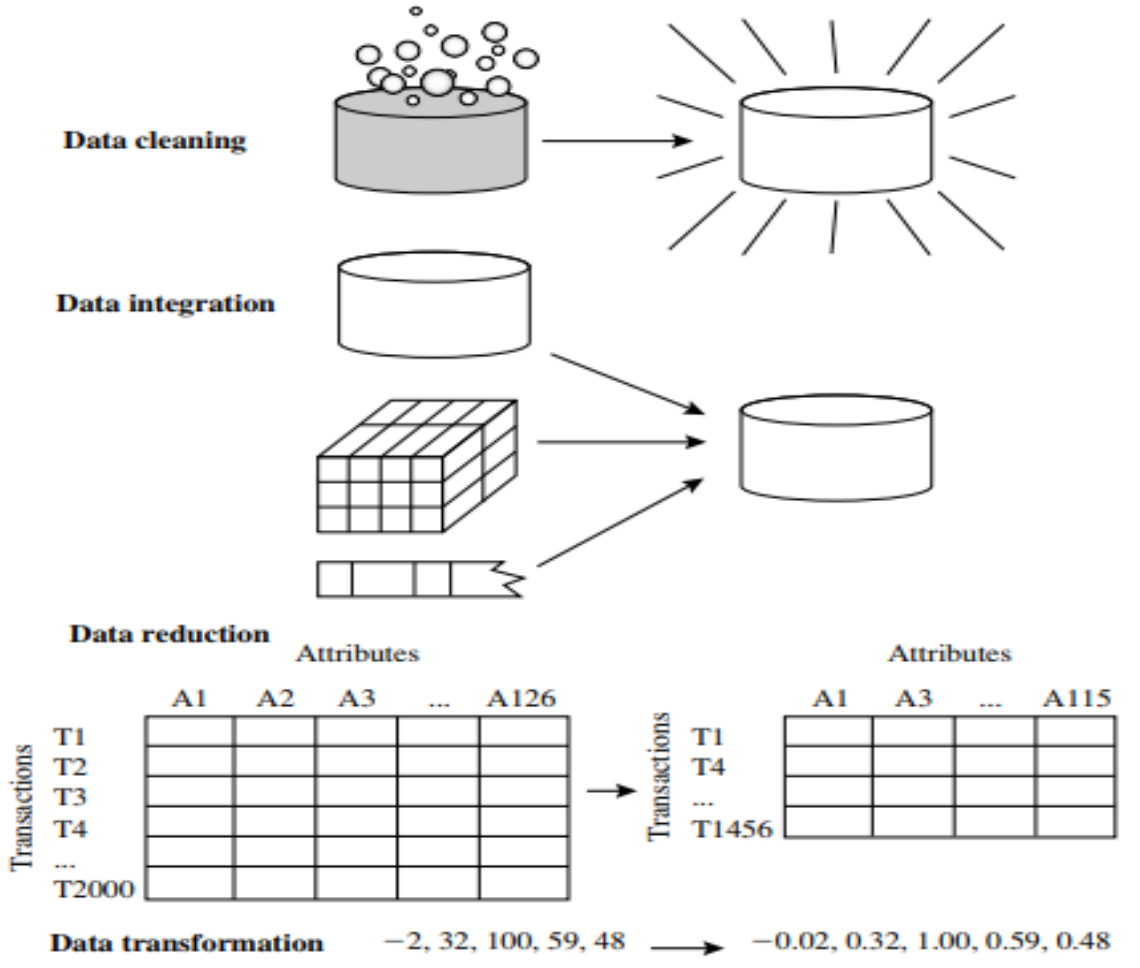
Data processing techniques, when applied before mining, can substantially **improve the overall quality of the patterns mined and/or the time required for the actual mining**.

Data Quality: Why Preprocess the Data?

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.

Major Tasks in Data Preprocessing

The major steps involved in data preprocessing, namely, data cleaning, data integration, data reduction, and data transformation.



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. In this section, you will study basic methods for data cleaning.

❖ Missing Values

How can you go about filling in the missing values for this attribute? Let's look at the following methods.

1. **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.
2. **Fill in the missing value manually:** In general, this approach is time consuming and may not be feasible given a large data set with many missing values.
3. **Use a global constant to fill in the missing value:** Replace all missing attribute values by the same constant such as a label like "Unknown" or $-\infty$. If missing values are replaced by, say, "Unknown," then the mining program may mistakenly think that they form an interesting concept, since they all have a value in common—that of "Unknown." Hence, although this method is simple, it is not recommended.
4. **Use the attribute mean to fill in the missing value**: For example, suppose that the data distribution regarding the income of AllElectronics customers is symmetric and that the mean income is $56,000. Use this value to replace the missing value for income.
5. **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.
6. **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. For example, using the other customer attributes in your data set, you may construct a decision tree to predict the missing values for income.

- **Noisy Data**

"What is noise?" **Noise is a random error or variance in a measured variable.** Let's look at the following data smoothing techniques.

1. **Binning:** Binning methods smooth a sorted data value by consulting its "neighborhood," that is, the values around it. The sorted values are distributed into a number of "buckets," or bins. Because binning methods consult the neighborhood of values, they perform local smoothing. Figure 3.2 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).

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

Binning methods for data smoothing.

   a. 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.
   b. Similarly, **smoothing by bin medians** can be employed, in which each bin value is replaced by the bin median.

c. **In smoothing by bin boundaries**, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value. In general, the larger the width, the greater the effect of the smoothing. Alternatively, bins may be equal width, where the interval range of values in each bin is constant.

2. **Clustering :** 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.

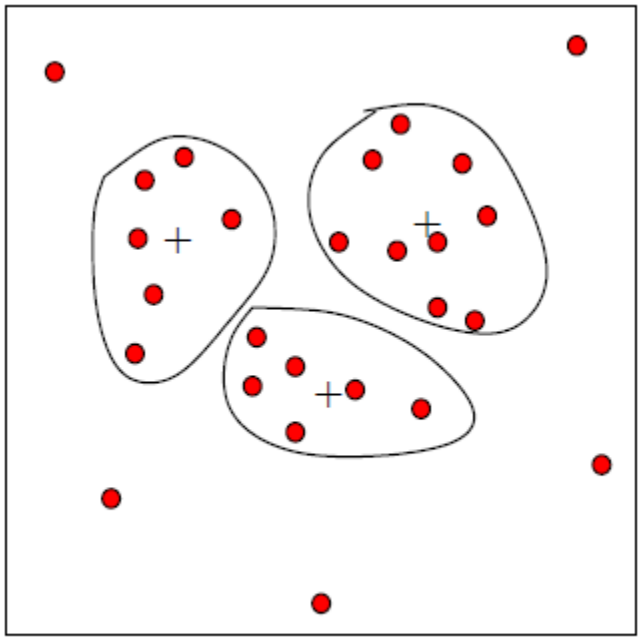


Figure 3.3: Outliers may be detected by clustering analysis.

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

(e.g., mislabeled characters). Patterns whose surprise content is above a threshold are output to a list. A human can then sort through the patterns in the list to identify the actual garbage ones.

This is much faster than having **to manually search through the entire database.** The garbage patterns can then be removed from the (training) database. The garbage patterns can be excluded from use in subsequent data mining.

4. **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.

❖ Inconsistent data

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

There may also be inconsistencies due to data integration, where a given attribute can have different names in different databases. Redundancies may also result.

Data Integration and Transformation

Data mining often requires **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.

Data Integration

Data integration, which **combines data from multiple sources into a coherent data store**, as in data warehousing. These sources may include multiple databases, data cubes, or flat files.

There are a number of issues to consider during data integration. **Schema integration 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, how can the data analyst or the computer be sure that *customer_id* in one database and *cust_number* in another refer to the same attribute? 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**.

Redundancy is another important issue in data integration. An attribute (such as annual revenue, for instance) 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 numeric attributes, we can evaluate the correlation between two attributes, A and B, by computing the **correlation coefficient** (also known as Pearson's product moment coefficient, named after its inventer, Karl Pearson). This is

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

where n is the number of tuples, ai and bi are the respective values of A and B in tuple i, $\bar{A}$ and $\bar{B}$ are the respective mean values of A and B, σA and σB are the respective standard deviations of A and B. Note that −1 ≤ rA,B ≤ +1. **If rA,B is greater than 0, then A and B are positively correlated, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that A (or B) may be removed as a redundancy. If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, where the values of one attribute increase as the values of the other attribute decrease. This means that each attribute discourages the other.**

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).

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. One university may adopt a quarter system, offer three courses on database systems, and assign grades from A+ to F, whereas another may adopt a semester syst em, offer two courses on databases, and assign grades from 1 to 10. It is difficult to work out precise course-to-grade transformation rules between the two universities, making information exchange difficult.

Data Transformation

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

1. **Smoothing,** which works to remove noise from the data. Techniques include binning, regression, and clustering.
2. **Aggregation**, where summary or aggregation operations are applied to the data. For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for data analysis at multiple granularities.
3. **Generalization of the data,** where low-level or "primitive"(raw) data are replaced by higher level concepts through the use of concept heirarchies. (for ex. where attributes such as street can be generalized to higher-level concepts, like city or country)
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. **Attribute construction (or feature construction),** where **new attributes are constructed and added from the given set of attributes** to help the mining process.

Normalization

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. There are many methods for data normalization. We study **min-max normalization,**

z-score normalization, and normalization by decimal scaling. For our discussion, let A be a numeric attribute with n observed values, v_1, v_2,..., v_n.

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, v_i , of A to $v_i{}'$ in the range [new min_A,new max_A] 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

.

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

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}, \tag{3.9}$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. The mean and standard deviation were discussed in Section 2.2, where $\bar{A} = \frac{1}{n}(v_1 + v_2 + \cdots + v_n)$ and σ_A is computed as the square root of the variance of A (see Eq. (2.6)). 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.

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 , of A is normalized to v/ by computing

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

where j is the smallest integer such that $max(|v_i'|) < 1$.

Decimal scaling. 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 1000 (i.e., j = 3) so that −986 normalizes to −0.986 and 917 normalizes to 0.917.

In **attribute construction**, new attributes are constructed from the given attributes and added in order to help to improve accuracy and understanding of structure in high dimensional data. For example, we may wish to add the attribute area based on the attribute's height and width. Attribute construction can help alleviate the fragmentation problem when decision tree algorithms are used for classification, where an attribute is repeatedly tested along a path in derived decision tree. By combining attributes, attribute

construction can discover missing information about the relationships between data attributes that can be useful for knowledge discovery.

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.**

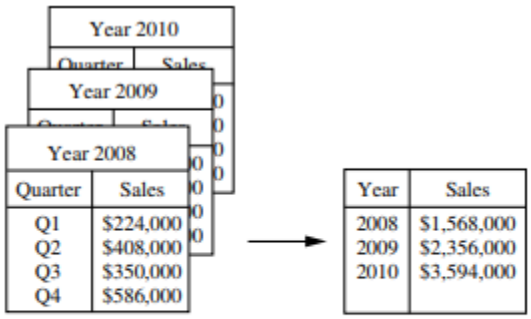
Data reduction strategies include the following

1. **Data Cube Aggregation**, where **aggregation operations are applied to the construction of a data cube.**
2. **Dimension Reduction**, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
3. **Data compression,** where encoding mechanisms are used to reduce the data set size.
4. **Numerosity reduction**, where the the original data volume is replaced 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.
5. **Discretization and concept hierarchy generation**, where raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are a powerful tool for data mining.

➲ Data Cube Aggregation

Imagine that you have collected the data for your analysis. These data consist of the AllElectronics 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**. This aggregation is illustrated in Figure 3.10. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.



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.

Data cubes store multidimensional aggregated information. 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.

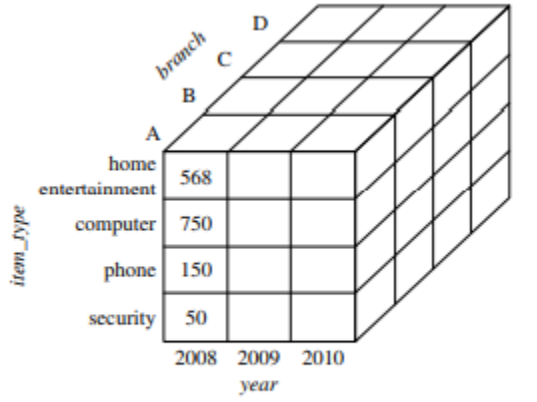


Figure 3.11 A data cube for sales at *AllElectronics*.

Each cell holds an aggregate data value, corresponding to the data point in multidimensional space. (For readability, only some cell values are shown.) 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 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. 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.

➲ Dimensionality Reduction

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. For example, if the task is to classify customers based on whether or not they are likely **to purchase a popular new CD at AllElectronics** when notified of a sale, attributes such as the customer's **telephone number are likely to be irrelevant, unlike attributes such as age or music taste.** Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time-consuming task, especially when the 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.

Dimensionality Reduction reduces the data set size by removing such attributes or dimensions from it. The methods of attribute subset selection are applied.

The goal of attribute subset selection is to **find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes.** Mining on a reduced set of attributes has an additional benefit: **It reduces the number of attributes appearing in the discovered patterns, helping to make the patterns easier to understand.**

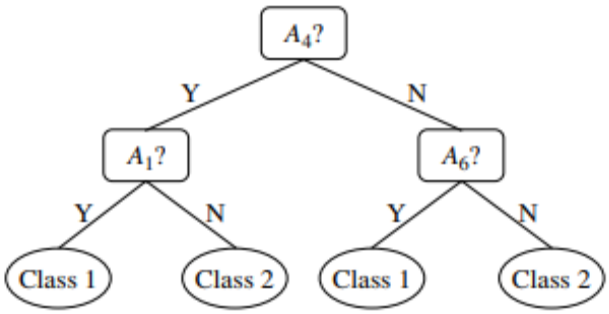
"How can we find a 'good' subset of the original attributes?" For n attributes, there are 2^n possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as n and the number of data classes increase. Therefore, heuristic methods that explore 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.

Basic heuristic methods of attribute subset selection include the techniques that follow, some of which are illustrated in Figure 3.6.

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.

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: $\{\}$ $\Rightarrow \{A_1\}$ $\Rightarrow \{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\}$ $\Rightarrow \{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  => Reduced attribute set: $\{A_1, A_4, A_6\}$

Greedy (heuristic) methods for attribute subset selection.

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.

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.

If the mining task is classification, and **the mining algorithm itself is used to determine the attribute subset,** then this is called a **wrapper approach;** otherwise, it is

a filter approach. In general, the wrapper approach leads to greater accuracy since it optimizes the evaluation measure of the algorithm while removing attributes. However, it requires much more computation than a filter approach.

➲ Data Compression

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. We focus on two popular and effective methods of lossy data compression: **wavelet transforms and principal component analysis.**

1.Wavelet Transforms

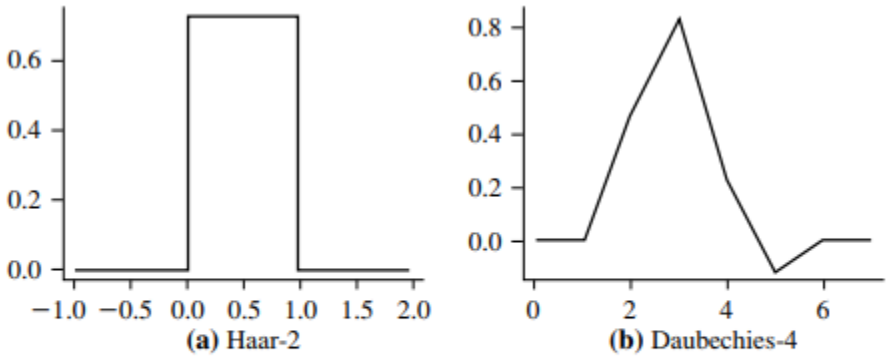
The discrete wavelet transform (DWT) is a linear signal processing technique that, when applied to a data vector X, transforms it to a numerically different vector, $X^{\prime}$, of wavelet coefficients. The two vectors are of the same length. When applying this technique to data reduction, we consider each tuple as an n-dimensional data vector, that is, X = (x1,x2,...,xn), depicting n measurements made on the tuple from n database attributes.

"How can this technique be useful for data reduction if the wavelet transformed data are of the same length as the original data?" 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.** 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.**

The DWT is closely related to the discrete Fourier transform (DFT), a signal processing technique involving sines and cosines. In general, however, the DWT achieves better lossy compression. That is, if the same number of coefficients is retained for a DWT and a DFT of a given data vector, the DWT version will provide a more accurate approximation of the original data. Hence, for an equivalent approximation, the DWT requires less space than the DFT. Unlike the DFT, wavelets are quite localized in space, contributing to the conservation of local detail.

There is only one DFT, yet **there are several families of DWTs.** Figure 3.4 shows some wavelet families. Popular wavelet transforms include the Haar-2, Daubechies-4, and Daubechies-6.



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

The general procedure for applying a discrete wavelet transform uses **a hierarchical pyramid algorithm that halves the data at each iteration, resulting in fast computational speed.** The method is as follows:

1. The length, L, of the input data vector must be an integer power of 2. This condition can be met by padding the data vector with zeros as necessary (L ≥ n).

2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference, which acts to bring out the detailed features of the data.

3. The two functions are applied to pairs of data points in X, that is, to all pairs of measurements .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 highfrequency content of it, respectively.

4. 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.

5. Selected values from the data sets obtained in the previous iterations are designated the wavelet coefficients of the transformed data.

Equivalently, a matrix multiplication can be applied to the input data in order to obtain the wavelet coefficients, where the matrix used depends on the given 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. Although we do not have room to discuss it here, 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 realworld applications, including the compression of fingerprint images, computer vision, analysis of time-series data, and data cleaning.

2. Principal Components Analysis

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

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. Unlike attribute subset selection , which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA “combines” the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

The basic procedure is as follows:

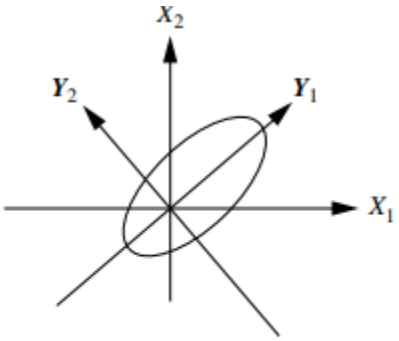
1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.

2. PCA computes 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.

3. The principal components are sorted in order of decreasing “significance” or strength. The principal components essentially serve as a new set of axes for the data providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, Figure shows the first two principal components, Y1 and Y2, for the given set of data originally mapped to the axes X1 and X2. This information helps identify groups or patterns within the data.

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



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

➲ Numerosity 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.

- **Regression and Log-Linear Models: Parametric Data Reduction**

Regression and log-linear models can be used to approximate the given data. In **linear regression**, **the data are modeled to fit a straight line.** For example, a random

variable, Y (called a response variable), can be modeled as a linear function of another random variable, x (called a predictor variable), with the equation

y = 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. **This allows a higher-dimensional data space to be constructed from lower-dimensional spaces. Log-linear models are therefore also useful for dimensionality reduction (since the lower-dimensional points together typically occupy less space than the original data points) and data smoothing (since aggregate estimates in the lower-dimensional space are less subject to sampling variations than the estimates in the higher-dimensional space).**

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 : Non - Parametric method for numerosity reduction**

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 3.3 Histograms. The following data are a list of AllElectronics 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, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

Figure 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.

“How are the buckets determined and the attribute values partitioned?” 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).

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

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

V-optimal and MaxDiff are most accurate and practical histograms.

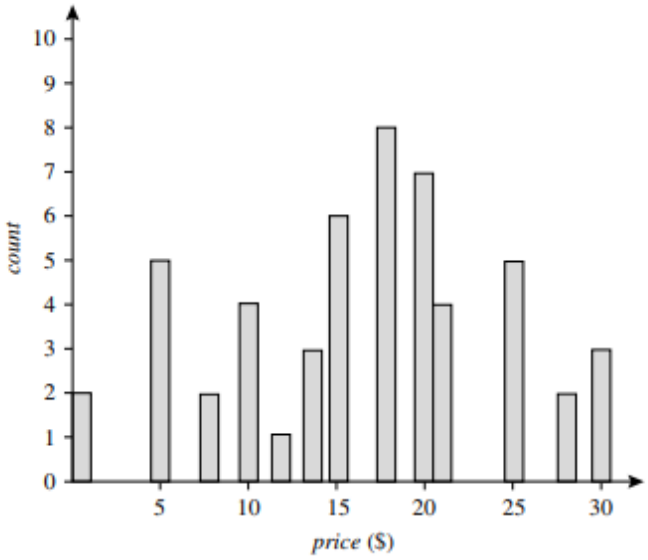


Figure 3.7 A histogram for *price* using singleton buckets—each bucket represents one price–value/frequency pair.

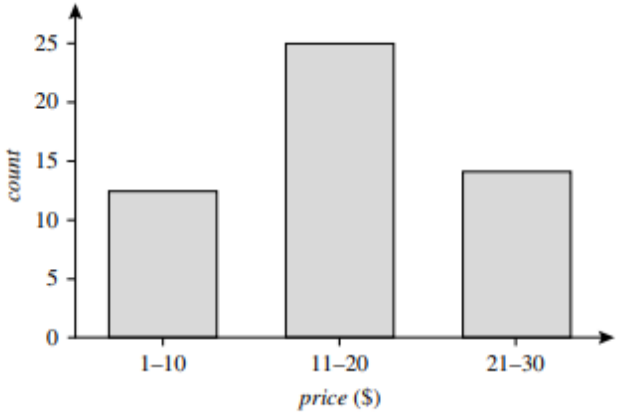
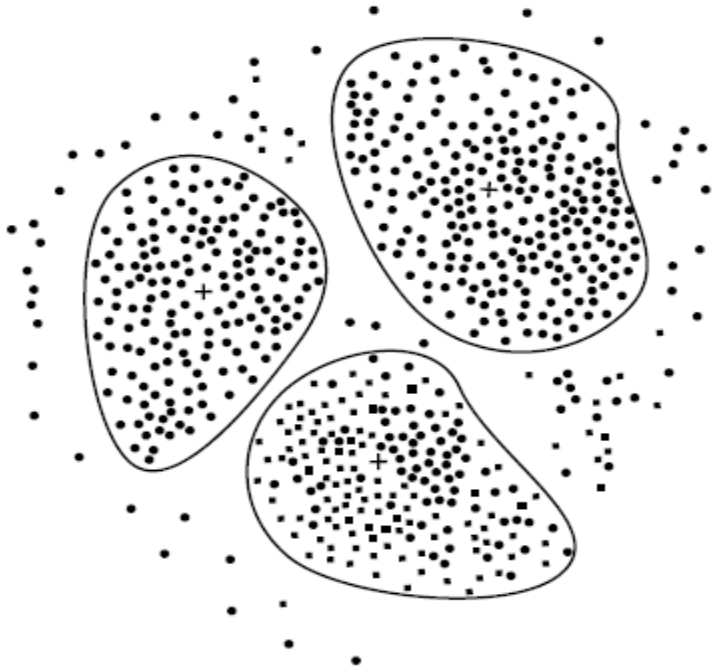


Figure 3.8 An equal-width histogram for *price*, where values are aggregated so that each bucket has a uniform width of $10.

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. More studies are needed regarding the

effectiveness of multidimensional histograms for high dimensionalities. **Singleton buckets are useful for storing high-frequency outliers.**

- **Clustering**



A 2-D plot of customer data with respect to customer locations in a city, showing three data clusters. Each cluster centroid is marked with a "+", representing the average point in space for that cluster. Outliers may be detected as values that fall outside of the sets of clusters.

Figure showed a 2-D plot of customer data with respect to customer locations in a city. Three data clusters are visible.

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.

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

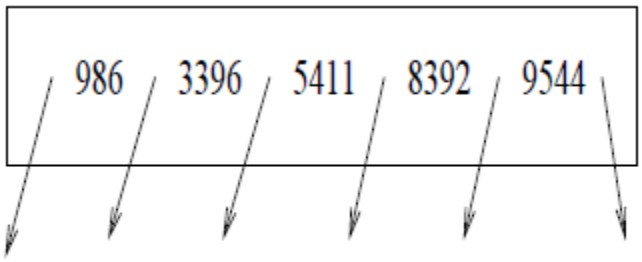


Figure 3.10: The root of a B+-tree for a given set of data.

An index tree can therefore store aggregate and detail data at varying levels of resolution or abstraction. It provides a hierarchy of clustering of the data set, where each cluster has a label that holds for the data contained in the cluster. If we consider each child of a parent node as a bucket, then an index tree can be considered as a hierarchical histogram. For example, consider the root of a B+-tree as shown in Figure with pointers to the data keys 986, 3396, 5411, 8392, and **9544. Suppose that the tree** contains 10,000 tuples with keys ranging from 1 to 9,999. The data in the tree can be approximated by an equi-depth histogram of 6 buckets for the key ranges 1 to 985, 986 to 3395, 3396 to 5410, 5411 to

8392, 8392 to 9543, and 9544 to 9999. Each bucket contains roughly 10,000/6 items. Similarly, each bucket is subdivided into smaller buckets, allowing for aggregate data at a finer-detailed level.

The use of multidimensional index trees as a form of data reduction relies on an ordering of the attribute values in each dimension. Multidimensional index trees include R-trees, quad-trees, and their variations. They are well-suited for handling both sparse and skewed data.

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. Let's look at the most common ways that we could sample D for data reduction, as illustrated in Figure 3.9.

1. **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, that is, all tuples are equally likely to be sampled.

2. **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.

3. **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.

4. **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.

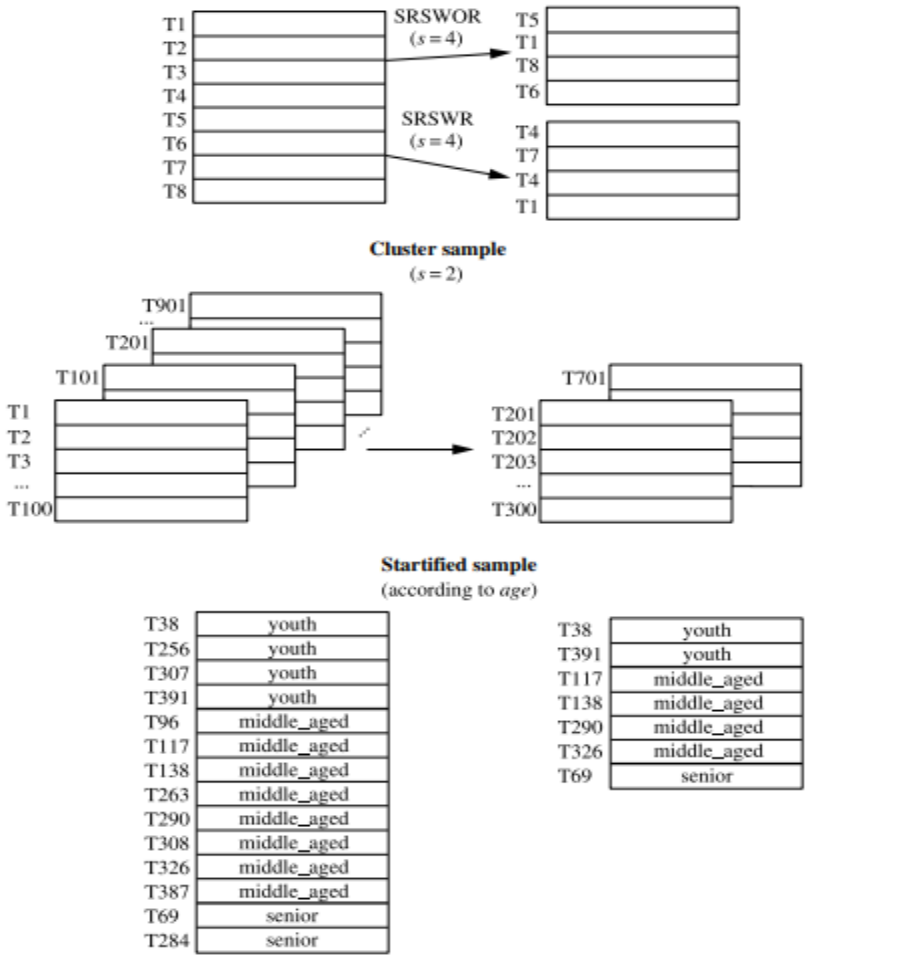


Figure 3.9 Sampling can be used for data reduction.

An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, s, as opposed to N, the data set size. Hence, sampling complexity is potentially sublinear to the size of the data. Other data reduction techniques can require at least one complete pass through D. For a fixed sample size, sampling complexity increases only linearly as the number of data

dimensions, n, increases, whereas techniques using histograms, for example, increase exponentially in n.

When applied to data reduction, sampling is most commonly used to estimate the answer to an aggregate query. It is possible (using the central limit theorem) to determine a sufficient sample size for estimating a given function within a specified degree of error. This sample size, 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.

➲ Discretization and concept hierarchy generation

Discretization techniques can be used to reduce the number of values for a continuos attribute, by dividing the range of attribute into intervals. Interval labels (e.g., 0–10, 11–20, etc.) can then be used to replace actual data values.

Techniques to **generate decision trees for classification** can be applied to discretization. Such techniques **employ a top-down splitting approach.** Unlike the other methods mentioned so far, decision tree approaches to discretization are supervised, that is, they make use of class label information.

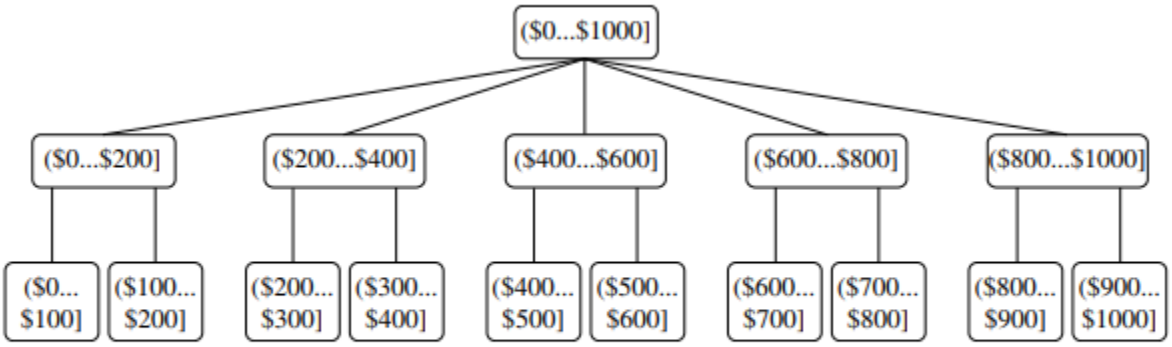
For example, we may have a data set of patient symptoms (the attributes) where each patient has an associated diagnosis class label. Class distribution information is used in the calculation and determination of split-points (data values for partitioning an attribute range). Intuitively, the main idea 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.

A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. A concept hierarchy for given numeric attribute defines a discretization of the attribute. Concept hierarchies can also be

used to to reduce the data by collecting and replacing low-level-concepts(such as numeric values for the attribute *age*) by higher level concepts (such as *young,middle-aged* or *senior*).

Figure shows a concept hierarchy for the attribute price. More than one concept hierarchy can be defined for the same attribute to accommodate the needs of various users.

Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, 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.



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

Discretization and Concept Hierarchy Generation for Numeric Data

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

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

1.Binning

Binning is a top-down splitting technique based on a specified number of bins. We already discussed binning methods for data smoothing. These methods are also 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.

2.Histogram Analysis

Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information. Histograms were introduced before . 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 (e.g., earlier in Figure for price, where each bucket has a width of $10). 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, or the minimum number of values for each partiton at each level.

A concept hierarchy for ***price,*** generated is shown in Figure 3.13.

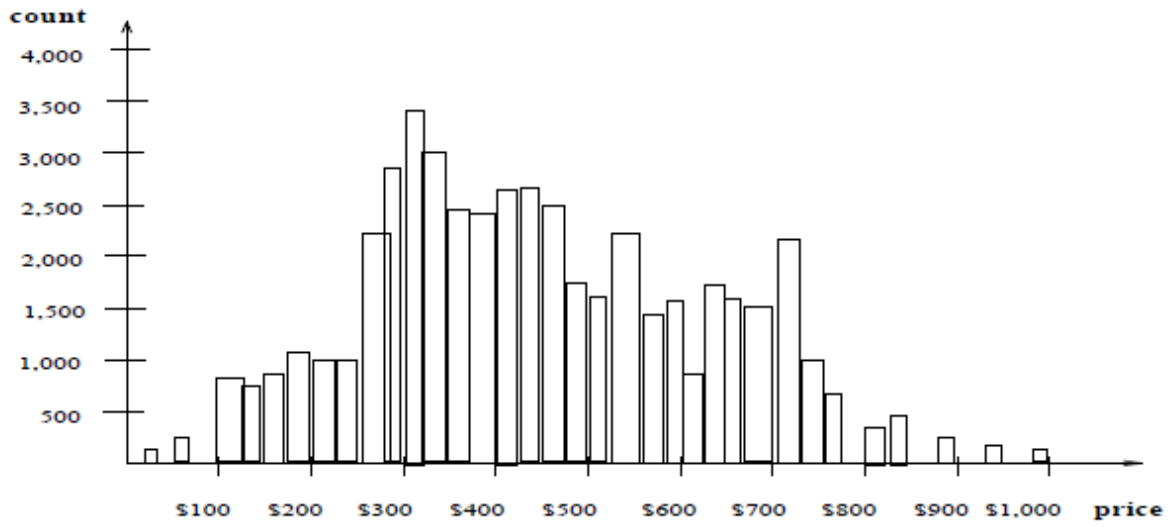


Figure 3.13: Histogram showing the distribution of values for the attribute *price*.

3. 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.

4.Entropy Based Discretization

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

1. Each value of A can be considered a potential interval boundary or threshold T. For ex. a value *v* of A can partition S into two subset satisfying the conditions A<*v* and A≥ *v* thereby creating a binary discretization.
2. Given S,the threshold value selected is the one that maximizes ther information gain resulting from subsequent partitioning. The information gain is

$$I(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2),$$

   Where S1 and S2 are samples satisfing conditions A<*v* and A≥ *v* .
   The entropy function Ent() is based on class distribution of the samples in the set. The entropy of S1 =

$$-\sum_{i=1}^{m} p_i \log_2(p_i),$$

   where pi is the probability of class i in S1. Determined by dividing the number of samples of class i in S1 by total number of samples in S1. The value of Ent(S2) can be calculated similarly.
3. The process of determining a threshold value is recursively applied to each partiton obtained until some stopping criterion is met. Such as
   Ent(s) – I(S,T) > **δ**

Entrpoy based discretization can reduce data size.Entropy based discretization uses class information. This makes it more likely that the interval bodies arfe defined to occur in places that may help improve classification accuracy.

5.Segmentaion by Natural Partitioning

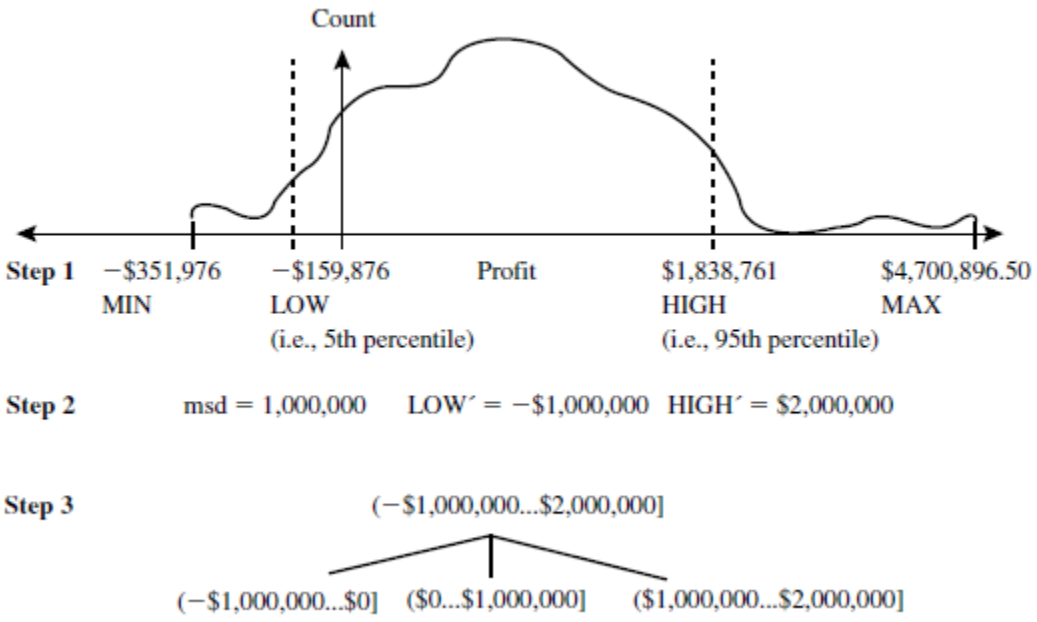
Although binning, histogram analysis, clustering and entropy-based discretization are useful in the generation of numerical hierarchies, many users would like to see

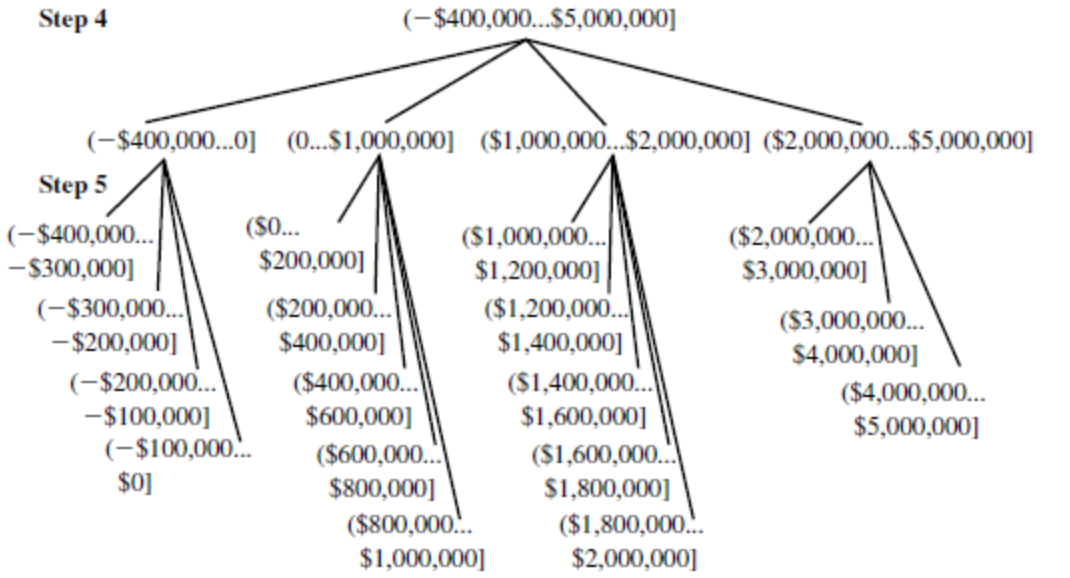
numerical ranges partitioned into relatively uniform, easy-to-read intervals that appear intuitive or “natural". For example, annual salaries broken into ranges like [$50,000, $60,000) are often more desirable than ranges like [$51263.98, $60872.34), obtained by some sophisticated clustering analysis.

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

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

This rule can be recursively applied to each interval, creating a concept hierarchy for the given numeric attribute.





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

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

Ex: Suppose that profits at different branches of *AllElectronics* for the year 2004 cover a wide range, from -$351,976.00 to $4,700,896.50. A user desires the automatic generation of a concept hierarchy for *profit*. For improved readability, we use the notation (*l*...*r*] to represent the interval (*l*, *r*]. For example, (-$1,000,000...$0] denotes the range from -$1,000,000 (exclusive) to $0 (inclusive).

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

1. Based on the above information, the minimum and maximum values are *MIN* = -$351,976:00, and *MAX* = $4,700,896:50. The low (5th percentile) and high (95th percentile) values to be considered for the top or first level of discretization are *LOW* = -$159,876, and *HIGH* = $1,838,761.

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

LOW' = -$1,000,000, rounding *HIGH* up to the million dollar digit, we get *HIGH'* = +$2,000,000.

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

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

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

5. Recursively, each interval can be further partitioned according to the 3-4-5 rule to form the next lower level of the hierarchy: The first interval, (-$400,000. . . $0], is partitioned into 4 subintervals: (-$400,000. . .-$300,000], (-$300,000. . .-$200,000],(-$200,000. . .-$100,000], and (-$100,000. . . $0]. The second interval, ($0. . . $1,000,000], is partitioned into 5 subintervals: ($0 .... $200,000],($200,000. . . $400,000],($400,000. . . $600,000],($600,000. . . $800,000], and ($800,000. . . $1,000,000]. The third interval, ($1,000,000. . . $2,000,000], is partitioned into 5 subintervals: ($1,000,000. . . $1,200,000],($1,200,000. . . $1,400,000],($1,400,000. . . $1,600,000], ($1,600,000 . . . $1,800,000], and ($1,800,000 . . . $2,000,000].

The last interval, ($2,000,000. . . $5,000,000], is partitioned into 3 subintervals: ($2,000,000. . . $3,000,000], ($3,000,000. . . $4,000,000], and ($4,000,000 . . . $5,000,000].

Similarly, the 3-4-5 rule can be carried on iteratively at deeper levels, as necessary

Concept Hierarchy Generation for Categorical Data

We now look at data transformation for categorical data. In particular, we study concept hierarchy generation for categorical attributes. 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.

We study four methods for the generation of concept hierarchies for nominal data, as follows.

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.

“Without knowledge of data semantics, how can a hierarchical ordering for an arbitrary set of nominal attributes be found?” 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**.

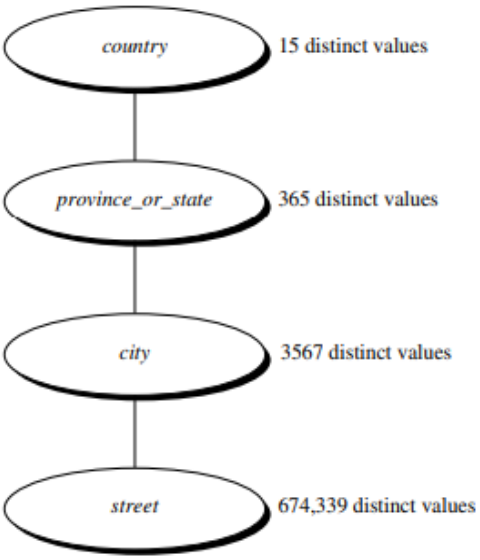
This heuristic rule works well in many cases. Some local-level swapping or adjustments may be applied by users or experts, when necessary, after examination of the generated hierarchy. Let’s examine an example of this method.

Example 3.7**.** 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. 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. 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.



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. Users, however, should have the option to override this feature, as necessary.

Example 3.8. Suppose that a data mining expert (serving as an administrator) has pinned together the five attributes number, street, city, province or state, and country, because they are closely linked semantically regarding the notion of location. If a user were to specify only the attribute city for a hierarchy defining location, the system can automatically drag in all five semantically related attributes to form a hierarchy. The user

may choose to drop any of these attributes (e.g., number and street) from the hierarchy, keeping city as the lowest conceptual level.

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.