Unit-1 Part-B Data Preprocessing

1. Why preprocessing?

Real world data are generally

Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data

Noisy: containing errors or outliers

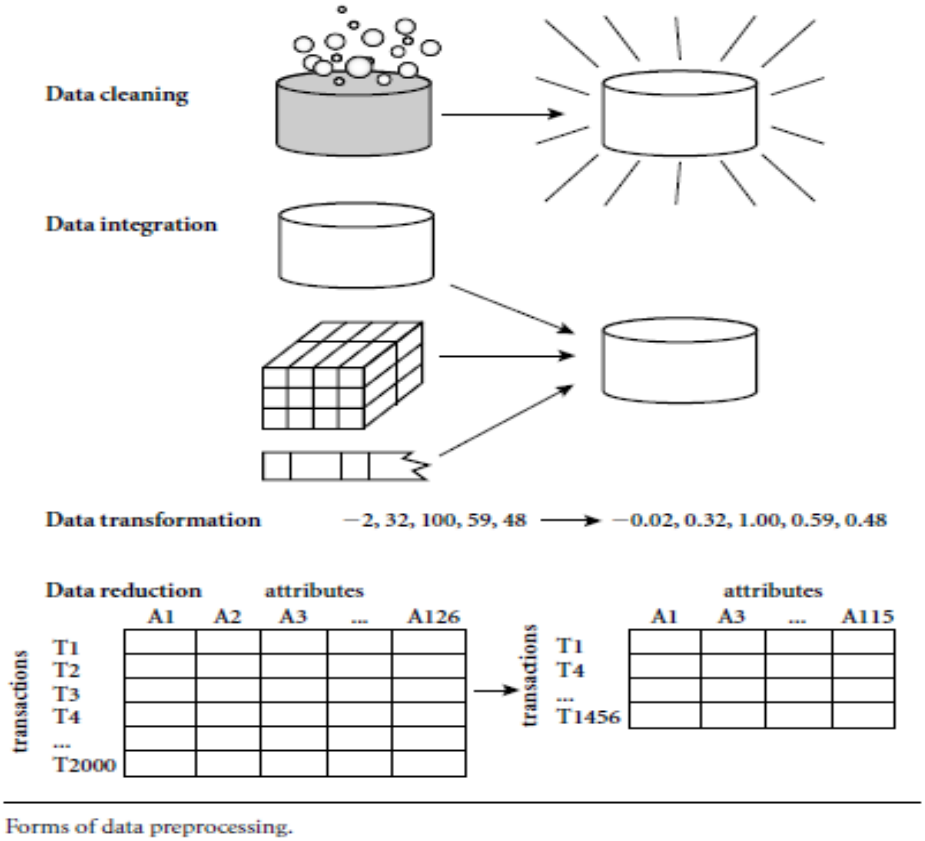
Inconsistent: containing discrepancies in codes or names.

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
  - Data warehouse needs consistent integration of quality data

2. Tasks in data preprocessing:

- **Data cleaning:** fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
- **Data integration:** Integration of multiple databases, data cubes, or files
- **Data transformation:** normalization and aggregation.
- **Data reduction:** Obtains reduced representation in volume but produces the same or similar analytical results
- **Data discretization:** part of data reduction, replacing numerical attributes with nominal ones

Forms of data preprocessing:



Forms of data preprocessing.

3. Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent. *Data cleaning* (or *data cleansing*) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

Missing Values

- Data is not always available
    - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
    - equipment malfunction
    - inconsistent with other recorded data and thus deleted
    - data not entered due to misunderstanding
    - certain data may not be considered important at the time of entry
    - not register history or changes of the data

How to Handle Missing Data?

Let's look at the following methods:

i. Ignore the tuple: usually done when class label is missing
ii. Fill in the missing value manually
iii. Use a global constant to fill in the missing value: ex. "unknown"
iv. Use the attribute mean to fill in the missing value
v. Use the attribute mean for all samples belonging to the same class to fill in the missing value
vi. Use the most probable value to fill in the missing value: inference-based such as Bayesian formula or decision tree

Noisy Data

"What is noise?" **Noise** is a random error or variance in a measured variable. some basic statistical description t echniques (e.g., boxplots and scatter plots), and methods of data visualization can be used to identify outliers, which may represent noise.

Given a numeric attribute such as, say, *price*, how can we "smooth" out the data to remove the noise? Let's look at the following data smoothing techniques.

a) 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.

- first sort data and partition into (equal-frequency) bins
- then one can smooth by bin means, smooth by bin median, smooth by bin boundaries
- In **smoothing by bin means**, each value in a bin is replaced by the mean value of the bin
- **In smoothing by bin medians,** 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

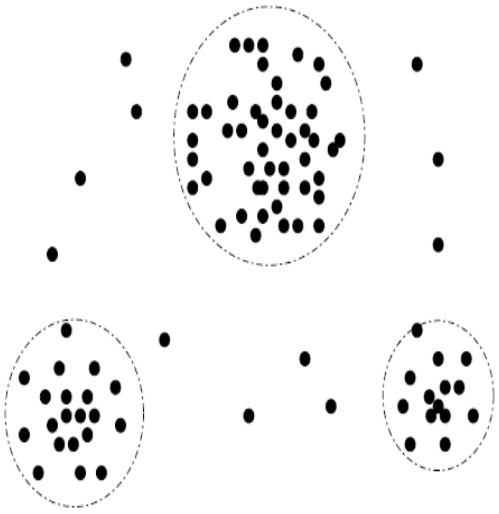
Smoothing by bin means:
Bin 1: 9, 9, 9
Bin 2: 22, 22, 22
Bin 3: 29, 29, 29

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

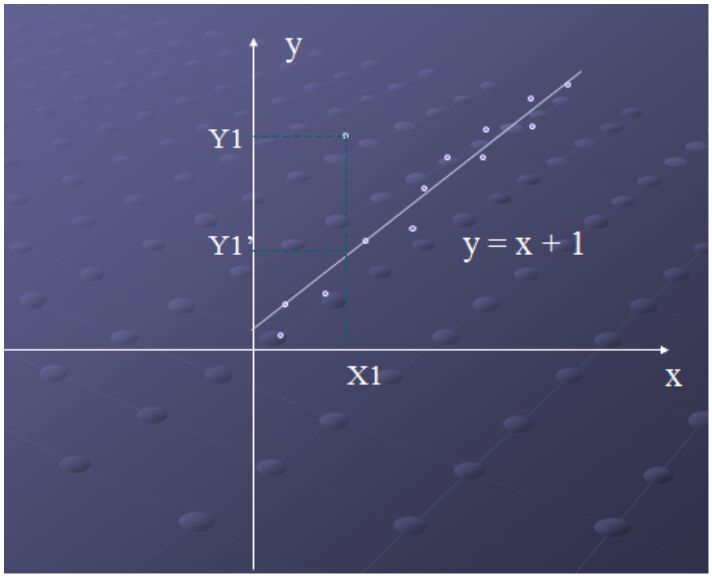
Binning methods for data smoothing.

Regression: Data smoothing can also be done by regression, a technique that conforms data values to a function. *Linear regression* involves finding the "best" line to fit two attributes (or variables) so that one attribute can be used to predict the other. *Multiple linear regression* is an extension of linear regression, where more than two attributes are involved and the data are fit to a multidimensional surface.

Outlier analysis: Outliers may be detected by clustering, for example, where similar values are organized into groups, or "clusters." Intuitively, values that fall outside of the set of clusters may be considered outliers.



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.



Linear Regression

4. Data Integration

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.

Challenges in data integration:

i) Entity Identification Problem

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

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?

ii) Redundancy and Correlation Analysis

Redundancy is another important issue in data integration. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.
Some redundancies can be detected by **correlation analysis**. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data.

- For nominal data, we use the **χ^2 (*chi-square*)** test.
- For numeric attributes, we can use the ***correlation coefficient*** **and** ***covariance***,

both of which access how one attribute's values vary from those of another.

Correlation Coefficient for Numeric 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, a_i and b_i 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, σ_A and σ_B are the respective standard deviations of A and B

If $r_{A,B}$ 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.

Covariance of Numeric Data

In probability theory and statistics, correlation and covariance are two similar measures for assessing how much two attributes change together. Consider two numeric attributes *A* and *B*, and a set of *n* observations $\{(a_1, b_1),\ldots,(a_n, b_n)\}$. The mean values of *A* and *B*, respectively, are also known as the **expected values** on *A* and *B*, that is,

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n}$$

and

$$E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}.$$

The **covariance** between A and B is defined as

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

If we compare Eq. for $r_{A,B}$ (correlation coefficient) with Eq. for covariance, we see that

$$r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B},$$

where σ_A and σ_B are the standard deviations of A and B, respectively. It can also be shown that

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}.$$

For two attributes A and B that tend to change together, if A is larger than $\bar{A}$ (the expected value of A), then B is likely to be larger than $\bar{B}$ (the expected value of B). Therefore, the covariance between A and B is *positive*. On the other hand, if one of the attributes tends to be above its expected value when the other attribute is below its expected value, then the covariance of A and B is *negative*.

If A and B are *independent* (i.e., they do not have correlation), then $E(A \cdot B) = E(A) \cdot E(B)$. Therefore, the covariance is $Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B} = E(A) \cdot E(B) - \bar{A}\bar{B} = 0$.

Stock Prices for *AllElectronics* and *HighTech*

Time point	AllElectronics	HighTech
t1	6	20
t2	5	10
t3	4	14
t4	3	5
t5	2	5

Covariance analysis of numeric attributes. Consider Table which presents a simplified example of stock prices observed at five time points for *AllElectronics* and *HighTech*, a high-tech company. If the stocks are affected by the same industry trends, will their prices rise or fall together?

$$E(AllElectronics) = \frac{6+5+4+3+2}{5} = \frac{20}{5} = \$4$$

and

$$E(HighTech) = \frac{20+10+14+5+5}{5} = \frac{54}{5} = \$10.80.$$

Thus, using Eq. we compute

$$Cov(AllElectroncis, HighTech) = \frac{6 \times 20 + 5 \times 10 + 4 \times 14 + 3 \times 5 + 2 \times 5}{5} - 4 \times 10.80$$
$$= 50.2 - 43.2 = 7.$$

Therefore, given the positive covariance we can say that stock prices for both companies rise together.

iii) Tuple Duplication

In addition to detecting redundancies between attributes, duplication should also be detected at the tuple level (e.g., where there are two or more identical tuples for a given unique data entry case). The use of denormalized tables (often done to improve performance by avoiding joins) is another source of data redundancy. Inconsistencies often arise between various duplicates, due to inaccurate data entry or updating some but not all data occurrences.

iv) Data Value Conflict Detection and Resolution

Data integration also involves the *detection and resolution of data value conflicts*. For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding. For instance, a *weight* attribute may be stored in metric units in one system and British imperial units in another.

For a hotel chain, the *price* of rooms in different cities may involve not only different currencies but also different services (e.g., free breakfast) and taxes. When exchanging information between schools, for example, each school may have its own curriculum and grading scheme.

5. Data Reduction

- Imagine that you have selected data from the *AllElectronics* data warehouse for analysis.
- The data set will likely be huge! Complex data analysis and mining on huge amounts of data can take a long time, making such analysis impractical or infeasible.
- **Data reduction** techniques can be applied to obtain a reduced representation of the data set that ismuch 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

Data reduction strategies include

I. ***Dimensionality reduction*****,**
II. ***Numerosity reduction*****,**
III. ***Data compression*****.**

i. Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. Dimensionality reduction methods include

a) Attribute subset selection is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed.

b) P*rincipal components analysis*, which transform or project the original data onto a smaller space.

ii. Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation.

These techniques may be **parametric or nonparametric**.

Parametric methods-- a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Regression and log-linear models are examples.

Nonparametric methods for storing reduced representations of the data include *histograms*, *clustering*, *sampling*, and *data cube aggregation.*

iii. *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**. Dimensionality reduction and numerosity reduction techniques can also be considered forms of data compression.

a) Attribute Subset Selection

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant.

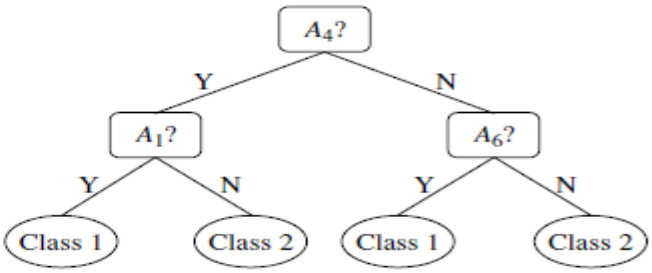
For example, if the task is to classify customers based on whether or not they are likely to purchase a popular new CD at *AllElectronics* when notified of a sale, attributes such as the customer's telephone number are likely to be irrelevant, unlike attributes such as *age* or *music taste*.

Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the mining algorithm employed. This can result in discovered patterns of poor quality. In addition, the added volume of irrelevant or

redundant attributes can slow down the mining process.

Attribute subset selection 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.

"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. 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

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.

i.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.
ii. Stepwise backward elimination: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.
iii. 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.
iv. Decision tree induction: Decision tree induction constructs a flowchart like structure where each internal (nonleaf) node denotes a test on an attribute, each branch corresponds to an outcome of the test, and each external (leaf) node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes.

b) Principal Component Analysis (PCA)

Principal Component Analysis (PCA) also called as Karhunen-Loeve (K-L) method

Procedure

- Take the whole dataset consisting of *d+1 dimensions* and ignore the labels such that our new dataset becomes *d dimensional.*
- Compute the *mean* for every dimension of the whole dataset.
- Compute the *covariance matrix* of the whole dataset.
- Compute *eigenvectors* and the corresponding *eigenvalues*.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ *dimensional* matrix **W.**
- Use this $d \times k$ *eigenvector matrix* to transform the samples onto the new subspace

The principal components (new set of axes) give important information about variance. Using the strongest components one can reconstruct a good approximation of the original signal.

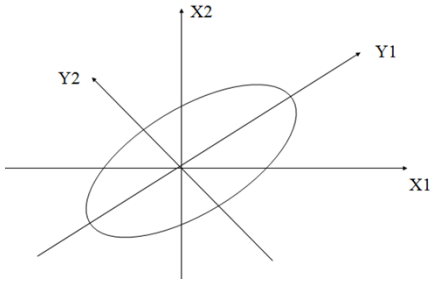


Figure : Principal components analysis. Y1 and Y2 are the first two principal components for the given data

d) 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

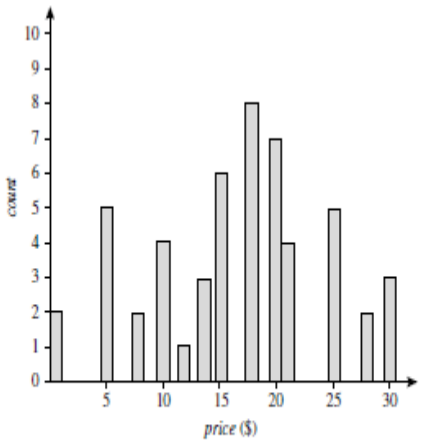
$$y = wx+b$$

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.
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)

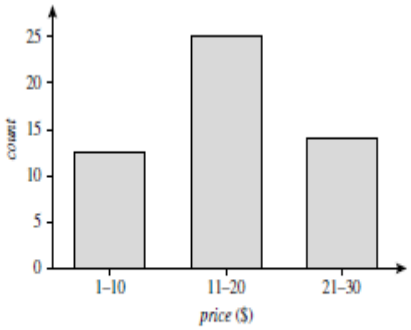
e) Histograms

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

Histograms. 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*.



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



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

"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).
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. Singleton buckets are useful for storing high-frequency outliers.

f) Clustering

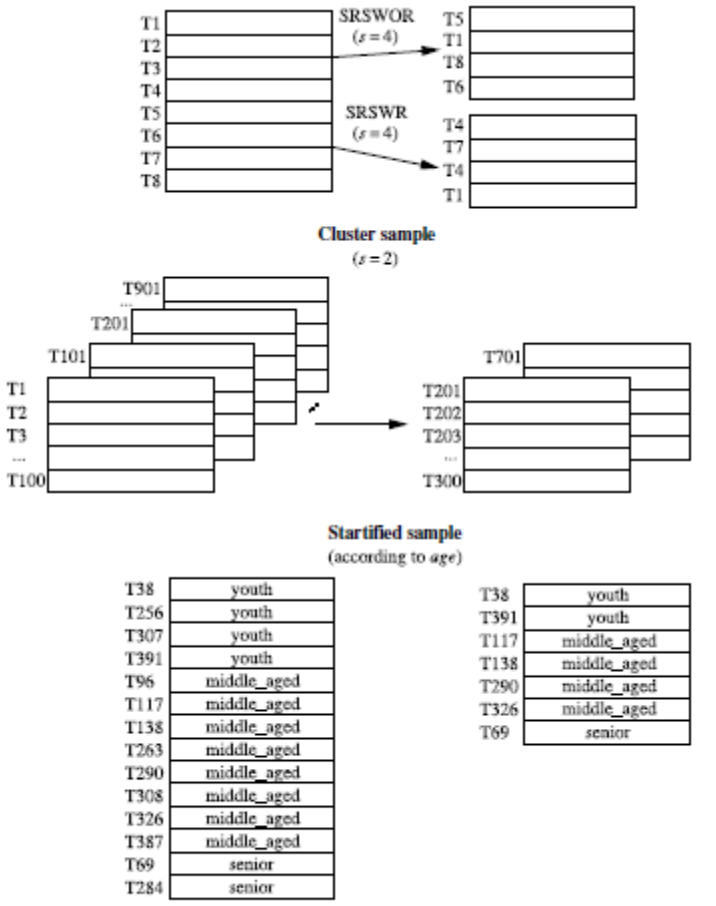
Clustering techniques consider data tuples as objects. They partition the objects into groups, or *clusters*, so that objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based 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.
In data reduction, the cluster representations of the data are used to replace the actual data.

g)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.

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.
Simple random sample with replacement (SRSWR) of size s: This is similar to SRSWOR, except that each time a tuple is drawn from D, it is recorded and then *replaced*. That is, after a tuple is drawn, it is placed back in D so that it may be drawn again.
Cluster sample: If the tuples in D are grouped into M mutually disjoint "clusters," then an SRS of s clusters can be obtained, where $s < M$. For example, tuples in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the tuples.

Stratified sample: If D is divided intomutually 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.

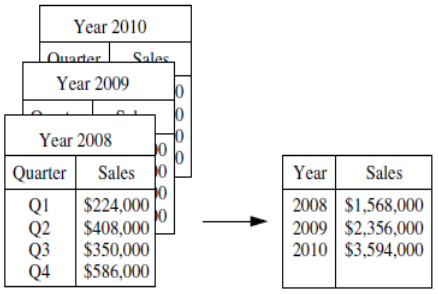


Sampling can be used for data reduction.

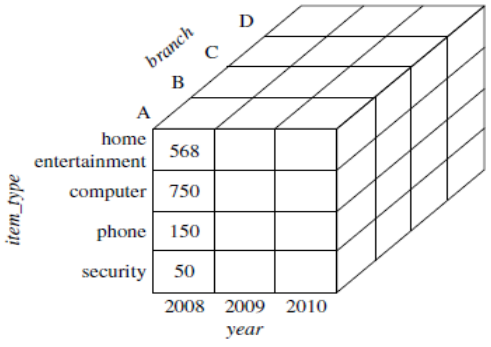
h) Data Cube Aggregation

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

Data cubes store multidimensional aggregated information. For example, Figure(left) shows a data cube for multidimensional analysis of sales data with respect to annual sales per item type for each *AllElectronics* branch. Each cell holds an aggregate data value, corresponding to the data point in multidimensional space.



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.



A data cube for sales at *AllElectronics*.

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.

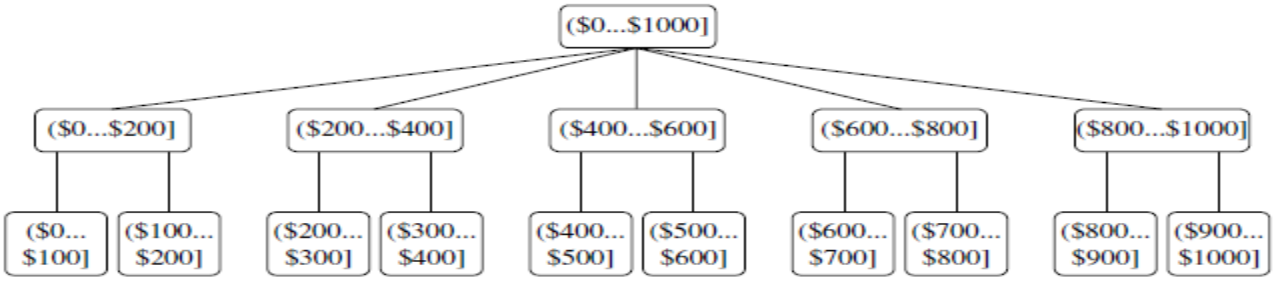
6. Data Transformation

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.
Data discretization, a form of data transformation.

Data Transformation Strategies

In *data transformation*, the data are transformed or consolidated into forms appropriate for mining.
Strategies for data transformation include the following:

i. **Smoothing**, which works to remove noise from the data. Techniques include binning, regression, and clustering.
ii. **Attribute construction** (or *feature construction*), where new attributes are constructed and added from the given set of attributes to help the mining process.
iii. **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 abstraction levels.
iv. **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.
v. **Discretization**, where the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth, adult*, *senior*). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute. Below 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.
vi. **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.



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

Data Transformation by Normalization:

The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or from kilograms to pounds for *weight*, may lead to very different results. In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or "weight." To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*. This involves transforming the data to fall within a smaller or common range such as [-1, 1] or [0.0, 1.0].

Methods for data normalization

1. Min-max normalization
2. z-score normalization
3. Normalization by decimal scaling

1. Min-max normalization

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 $[new\_min_A, new\_max_A]$ 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. 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 $\frac{73,600-12,000}{98,000-12,000}(1.0-0)+0 = 0.716$.

2. z-score normalization

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

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

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

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 $\frac{73,600-54,000}{16,000} = 1.225$.

3. Normalization by decimal scaling

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 $max(|v'_i|) < 1$.

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.

7. Data Discretization

Data discretization, a form of data transformation.

Discretization by Binning

Binning is a top-down splitting technique based on a specified number of bins.

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. 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*.

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.

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.

Measures of **correlation** can be used for discretization. *ChiMerge* is a $\chi 2$-based discretization method. The discretization methods that we have studied up to this point have all employed a top-down, splitting strategy. This contrasts with ChiMerge, which employs a bottom-up approach by finding the best neighboring intervals and then merging them to form larger intervals, recursively. As with decision tree analysis, ChiMerge is supervised in that it uses class information.

8. Concept Hierarchy Generation for Nominal Data

We now look at data transformation for nominal data. In particular, we study concept hierarchy generation for nominal attributes. Nominal attributes have a finite (but possibly large) number of distinct values, with no ordering among the values. Examples include *geographic_location*, *job_category*, and *item_type*.

Manual definition of concept hierarchies can be a tedious and time-consuming task for a user or a domain expert. Fortunately, many hierarchies are implicit within the database schema and can be automatically defined at the schema definition level. The concept hierarchies can be used to transform the data into multiple levels of granularity.

For example, data mining patterns regarding sales may be found relating to specific regions or countries, in addition to individual branch locations.

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,

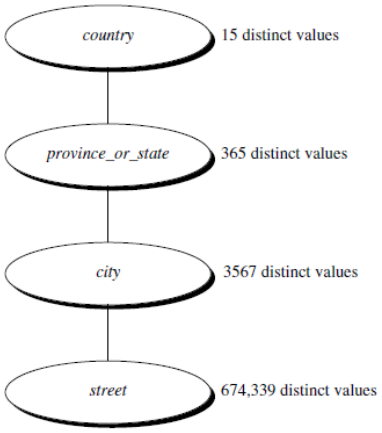
suppose that a relational database contains the following group of attributes: *street, city, province or state*, and *country*. A hierarchy can be defined by specifying the total ordering among these attributes at the schema level such as *street* < *city* <*province or state* < *country*.

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. we can easily specify explicit groupings for a small portion of intermediate-level data. For example, after specifying that *province* and *country*

For example, after specifying that *day, month, year* form a hierarchy at the schema level, a user could define some intermediate levels manually, such as "{*Month*} ⊂ *Quarter*"

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 systemcan then try to automatically generate the attribute ordering so as to construct a meaningful concept hierarchy.
Concept hierarchy can be automatically generated based on the **number of distinct values per attribute** in the given attribute set.



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*.