2.1 Data Pre-processing

Data Pre-processing

Why Data Pre-processing?

Data in the real world is dirty. That is it is incomplete or noisy or inconsistent.

- Incomplete: means lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - e.g., occupation=“ ”
- Noisy: means containing errors or outliers
  - e.g., Salary=“-10”
- Inconsistent: means containing discrepancies in codes or names
  - e.g., Age=“42” Birthday=“03/07/1997”
  - e.g., Was rating “1,2,3”, now rating “A, B, C”
  - e.g., discrepancy between duplicate records

Why Is Data Dirty?

Data is dirty because of the below reasons.

- *Incomplete data may come from*
  - “Not applicable” data value when collected
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human / hardware / software problems
- *Noisy data (incorrect values) may come from*
  - Faulty data collection instruments
  - Human or computer error at data entry
  - Errors in data transmission
- *Inconsistent data may come from*
  - Different data sources
  - Functional dependency violation (e.g., modify some linked data)
- *Duplicate records also need data cleaning*

Why Is Data Pre-processing Important?

Data Pre-processing is important because:

- *If there is No quality data, no quality mining results!*
  - Quality decisions must be based on quality data
  - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- *Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse*

Multi-Dimensional Measure of Data Quality

A well-accepted multidimensional view has the following properties:

- Accuracy
- Completeness
- Consistency

- Timeliness
- Believability
- Value added
- Interpretability
- Accessibility

Broad categories:

The above properties are broadly categorized into:

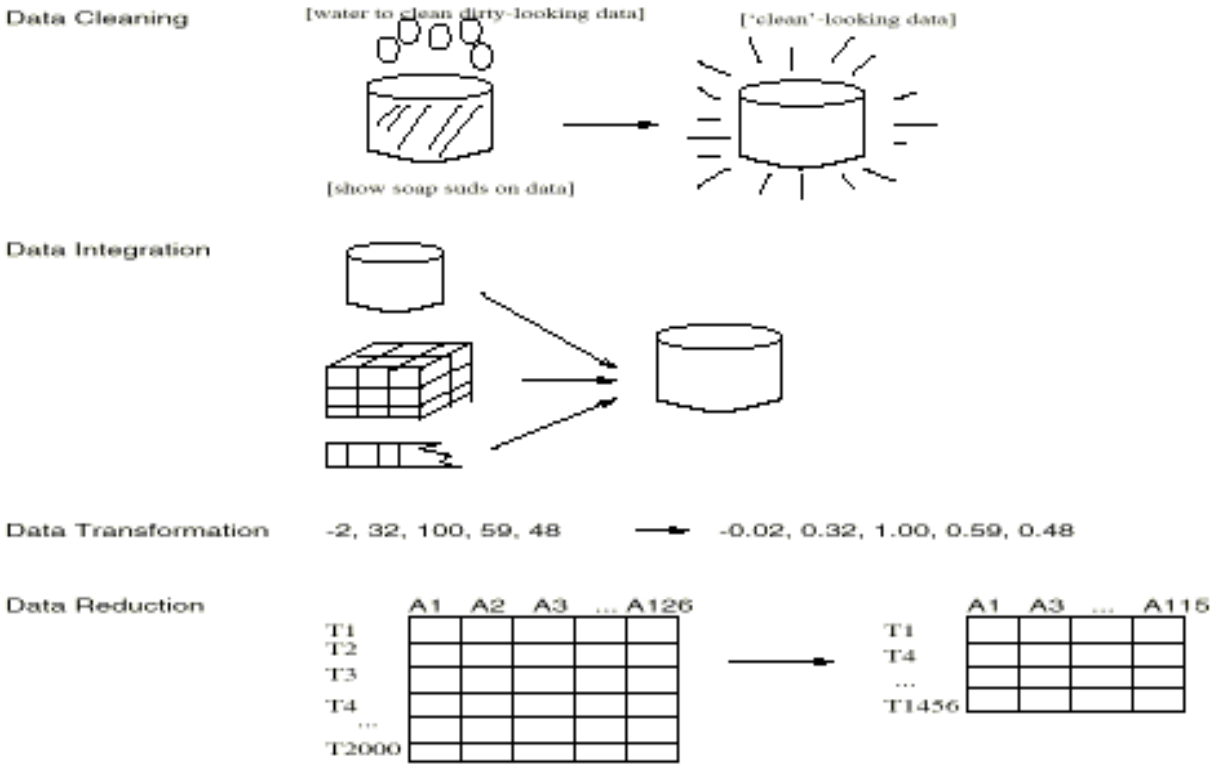
- Intrinsic
- Contextual
- Representational
- Accessibility

Major Tasks in Data Pre-processing

Major tasks in data pre-processing are data cleaning, data integration, data transformation, data reduction and data discretization.

- *Data cleaning*
  - Data Cleaning includes, filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.
- *Data integration*
  - Data Integration includes integration of multiple databases, data cubes, or files.
- *Data transformation*
  - Data Transformation includes normalization and aggregation.
- *Data reduction*
  - Data reduction is achieved by obtaining reduced representation of data in volume but produces the same or similar analytical results.
- *Data Discretization*
  - Data Discretization is part of data reduction but with particular importance, especially for numerical data.

Forms of Data Pre-processing



2.2 Data Cleaning

Data Cleaning

- *Importance of Data Cleaning*
  - "Data cleaning is one of the three biggest problems in data warehousing"—Ralph Kimball
  - "Data cleaning is the number one problem in data warehousing"—DCI survey

- *Data cleaning tasks are:*
  - Filling in missing values
  - Identifying outliers and smoothing out noisy data
  - Correcting inconsistent data
  - Resolving redundancy caused by data integration

Data Cleaning

Missing Data

Eg. Missing customer income attribute in the sales data

Methods of handling missing values:

a) Ignore the tuple

1) When the attribute with missing values does not contribute to any of the classes or has missing class label.
2) Effective only when more number of missing values are there for many attributes in the tuple.

3) Not effective when only few of the attribute values are missing in a tuple.

b) Fill in the missing value manually
1) This method is time consuming
2) It is not efficient
3) The method is not feasible

c) Use of a Global constant to fill in the missing value
1) This means filling with "Unknown" or "Infinity"
2) This method is simple
3) This is not recommended generally

d) Use the attribute mean to fill in the missing value
That is, take the average of all existing income values and fill in the missing income value.

e) Use the attribute mean of all samples belonging to the same class as that of the given tuple.
Say, there is a class "Average income" and the tuple with the missing value belongs to this class and then the missing value is the mean of all the values in this class.

f) Use the most probable value to fill in the missing value
This method uses inference based tools like Bayesian Formula, Decision tree etc.

Noisy Data

Apply data smoothing techniques like the ones given below:

a) Binning Methods

Simple Discretization Methods: Binning

- *Equal-width (distance) partitioning*
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well

- *Equal-depth (frequency) partitioning*
  - Divides the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

Binning Methods for Data Smoothing

- *Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34*

** Partition into equal-frequency (equi-depth) bins:*
- *Bin 1: 4, 8, 9, 15*
- *Bin 2: 21, 21, 24, 25*
- *Bin 3: 26, 28, 29, 34*

** Smoothing by bin means:*
- *Bin 1: 9, 9, 9, 9*
- *Bin 2: 23, 23, 23, 23*
- *Bin 3: 29, 29, 29, 29*

** Smoothing by bin boundaries:*
- *Bin 1: 4, 4, 4, 15*
- *Bin 2: 21, 21, 25, 25*
- *Bin 3: 26, 26, 26, 34*

b) Regression
- Method of mapping the data into an linear mathematical equation and converging them to a single line and finding the outlying data

c) Clustering
- Forming clusters and identifying the outlying data

Unsorted Data
21,25,34,8,4,15,21,24,28,29,9,26
Sorted Data
4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
Partition into equal frequency bins

a) Equal Width Partitioning / Distance Partitioning
Lowest value of the attribute = 4
Highest value of the attribute = 34
Width of the interval = (34 – 4) / 12 = 30 /12 = 2.5
Advantage: Straightforward
Disadvantage: Skewed data not handled

b) Equal Depth Partitioning / Frequency Partitioning
Fix N intervals,
Divide the sample into 3 intervals
Advantage: Good Data Scaling
Disadvantage: Managing categorical data is difficult

Inconsistent Data
a) Corrected manually using external references
b) Develop code functions that can correct data inconsistencies
c) Use available tools to correct data inconsistencies
- Known functional dependencies between attributes is used here

Data Cleaning as a Process

Handling missing data & noisy data is not all about data cleaning, but it is a big process.

1) First step in data cleaning is Discrepancy Detection

a) Using Meta Data
- Data about Data, - Attribute Value Domain,
- Acceptable value for an attribute, - Range of Attribute Values,
- Dependency between attributes

→ To detect discrepancy use code or tool

b) Using Field Overloading
- Eg. 20004 entered instead of 2004

c) Using Unique Rule

d) Using Consecutive Rule

e) Using Null Rule
- How to fill in field values that have blanks, ?, special characters...

Commercial tools to aid in discrepancy detection:

(i) Data Scrubbing Tool
- Uses simple domain knowledge to detect & correct errors

(ii) Data Auditing Tool
- Analyzes data to see if it satisfies rules & relationships
- Uses Statistical analysis to find correlations
- Uses clustering to find outliers

2) Second Step in Data Cleaning is Data Transformation
- Method of correcting the identified data discrepancies.

Commercial tools to aid in Data Transformation:

(i) Data Migration Tool:
- Simple Transformations
- Eg. Replace "Gender" by "Sex"

(ii) Extraction Transformation and Loading Tools (ETL):
- Users specify transformations through GUI

These two steps of Data Cleaning iterates until the user is satisfied.
This process is error prone and time consuming
Potter's Wheel – Publicly available Data Cleaning Tool.

2.3 Integration

Data Integration

Data Integration
- Combines data from multiple sources into a single store.
- Includes multiple databases, data cubes or flat files

Schema integration
- Integrates meta data from different sources
- Eg. A.cust_id = B.cust_no

Entity Identification Problem
- Identify real world entities from different data sources
- Eg. Pay_type filed in one data source can take the values 'H' or 'S', Vs in another data source it can take the values 1 or 2

Detecting and resolving data value conflicts:
- For the same real world entity, the attribute value can be different in different data sources
- Possible reasons can be - Different interpretations, different representation and different scaling
  - Eg. Sales amount represented in Dollars (USD) in one data source and as Pounds ($) in another data source.

Handling Redundancy in data integration:

- When we integrate multiple databases data redundancy occurs
- Object Identification – Same attributes / objects in different data sources may have different names.
- Derivable Data – Attribute in one data source may be derived from Attribute(s) in another data source
  Eg. Monthly_revenue in one data source and Annual revenue in another data source.
- Such redundant attributes can be detected using Correlation Analysis
- So, Careful integration of data from multiple sources can help in reducing or avoiding data redundancy and inconsistency which will in turn improve mining speed and quality.

Correlation Analysis – Numerical Data:

- Formula for Correlation Co-efficient =
  (Pearson's Product Moment Co-efficient)

$$r_{A,B} = \frac{\sum (A-\bar{A})(B-\bar{B})}{(n-1)\sigma_A \sigma_B} = \frac{\sum (AB) - n\bar{A}\bar{B}}{(n-1)\sigma_A \sigma_B}$$

$$\sigma = \sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$$

- Where, n = No. Of Tuples;
  $\bar{A}$ and $\bar{B}$ are respective means of A & B
  σ_A and σ_B are respective standard deviations of A & B
  $\Sigma(AB)$ is the sum of the cross product of A & B
- If the correlation co-efficient between the attributes A & B are positive then they are positively correlated.
- That is if A's value increases, B's value also increases.
- As the correlation co-efficient value increases, the stronger the correlation.
- If the correlation co-efficient between the attributes A & B is zero then they are independent attributes.
- If the correlation co-efficient value is negative then they are negatively Correlated.

Eg: Positive Correlation

A	B	A - A bar	B - B bar	Sqr (A - A bar)	Sqr (B - B bar)	AB
1	5	1	5	1	25	5
2	6	2	6	4	36	12
3	7	3	7	9	49	21
4	8	4	8	16	64	32
Sum				30	174	70

N	4
A bar	2.5
B bar	6.5
Sigma A	1.825741858
Sigma B	4.396968653
Correlation Co-efficient of A & B	0.2076137

Eg: Negative Correlation

A	B	A - A bar	B - B bar	Sqr (A - A bar)	Sqr (B - B bar)	AB
1	8	1	8	1	64	8
2	7	2	7	4	49	14
3	6	3	6	9	36	18
4	5	4	5	16	25	20
Sum				30	174	60

N	4
A bar	2.5
B bar	6.5
Sigma A	1.825741858
Sigma B	4.396968653
Correlation Co-efficient of A & B	-0.2076137

Eg: No Correlation

A	B	A - A bar	B - B bar	Sqr (A - A bar)	Sqr (B - B bar)	AB
1	4	1	4	1	16	4
2	3	2	3	4	9	6
3	3	3	3	9	9	9
4	4	4	4	16	16	16
Sum				30	50	35

N	4
A bar	2.5
B bar	3.5
Sigma A	1.825741858
Sigma B	2.357022604
Correlation Co-efficient of A & B	0

Correlation Analysis – Categorical Data:

- Applicable for data where values of each attribute are divided into different categories.
- Use Chi-Square Test (using the below formula)

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- If the value of X^2 is high, higher the attributes are related.
- The cells that contribute maximum to the value of X^2 are the ones whose Observed frequency is very high than its Expected frequency.
- The Expected frequency is calculated using the data distribution in the two categories of the attributes.
- Consider there are two Attributes A & B; the values of A are categorized into category Ai and Aj; the values of B are categorized into category Bi and Bj

- The expected frequency of Ai and Bj =
  Eij = (Count(Ai) * Count(Bj)) / N

	Play chess	*Not play chess*	*Sum (row)*
Like science fiction	***250(90)***	***200(360)***	***450***
Not like science fiction	***50(210)***	***1000(840)***	***1050***
Sum(col.)	***300***	***1200***	***1500***

- Eg. Consider a sample population of 1500 people who are surveyed to see if they Play Chess or not and if they Like Science Fiction books are not.
- The counts given within parenthesis are expected frequency and the remaining one is the observed frequency.
- For Example the Expected frequency for the cell (Play Chess, Like Science Fiction) is:
  = (Count (Play Chess) * Count (Like Science Fiction)) / Total sample population
  = (300 * 450) / 1500 = 90

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

- This shows that the categories Play Chess and Like Science Fiction are strongly correlated.

2.4 Transformation

Data Transformation

Smoothing:- Removes noise from the data
Aggregation:- Summarization, Data cube Construction
Generalization:- Concept Hierarchy climbing
Attribute / Feature Construction:- New attributes constructed from the given ones
Normalization:- Data scaled to fall within a specified range
- min-max normalization - z-score normalization
- normalization by decimal scaling

Data Transformation – Normalization:

Min-Max Normalization:

- Uses the formula:

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

Eg: Let income range \$12,000 to \$98,000 normalized to [0.0, 1,0].
Then \$73,600 is mapped to:

$$\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$$

Z-Score Normalization:

- Uses the formula:

$$v' = \frac{v - \mu_A}{\sigma_A}$$

- Where is μ mean and σ is standard deviation.
- Eg. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

Normalization by Decimal Scaling:

- Moves the decimal places of the values of the attribute A by a certain number of decimal places that depends upon the maximum absolute value of A.
- That is it uses the formula:

$$v' = \frac{v}{10^j}$$

- Where *j* is the smallest integer such that Max(|v'|) < 1
- Eg. A ranges from 986 to 917;
- Maximum absolute value of A is 986
- So divide each value of A by 1000 (j =3)
- Now normalized values of A ranges from -0.986 to 0.917

2.5 Reduction

Data Reduction

Why Data Reduction?

- A database of data warehouse may store terabytes of data
- Complex data analysis or mining will take long time to run on the complete data set

What is Data Reduction?

- Obtaining a reduced representation of the complete dataset
- Produces same result or almost same mining / analytical results as that of original.

Data Reduction Strategies:

1. Data cube Aggregation
2. Dimensionality reduction – remove unwanted attributes
3. Data Compression
4. Numerosity reduction – Fit data into mathematical models
5. Discretization and Concept Hierarchy Generation

1. Data Cube Aggregation:

- The lowest level of data cube is called as base cuboid.
- Single Level Aggregation - Select a particular entity or attribute and Aggregate based on that particular attribute.
  Eg. Aggregate along 'Year' in a Sales data.
- Multiple Level of Aggregation – Aggregates along multiple attributes – Further reduces the size of the data to analyze.
- When a query is posed by the user, use the appropriate level of Aggregation or data cube to solve the task
- Queries regarding aggregated information should be answered using the data cube whenever possible.

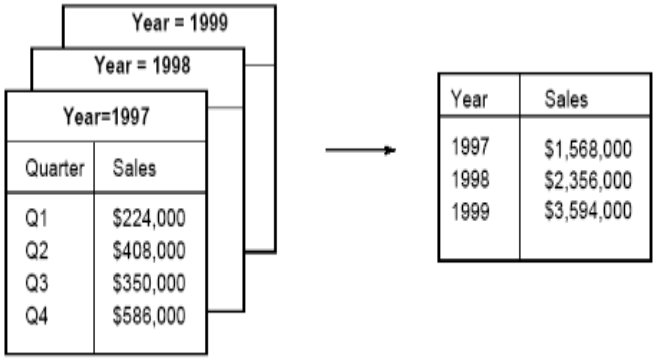


Figure 3.4: Sales data for a given branch of *AllElectronics* for the years 1997 to 1999. In the data on the left, the sales are shown per quarter. In the data on the right, the data are aggregated to provide the *annual_sales*.

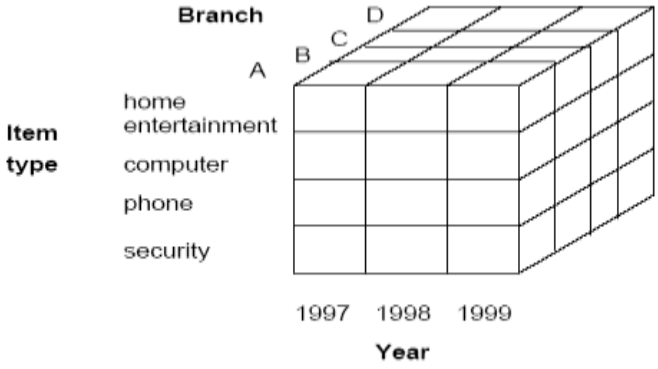


Figure 3.5: A data cube for sales at *AllElectronics*.

2. Attribute Subset Selection

Feature Selection: (attribute subset selection)

- The goal of attribute subset selection is to find the minimum set of Attributes such that the resulting probability distribution of data classes is as close as possible to the original distribution obtained using all Attributes.
- This will help to reduce the number of patterns produced and those patterns will be easy to understand

Heuristic Methods: (Due to exponential number of attribute choices)

- Step wise forward selection
- Step wise backward elimination
- Combining forward selection and backward elimination
- Decision Tree induction - Class 1 - A1, A5, A6; Class 2 - A2, A3, A4

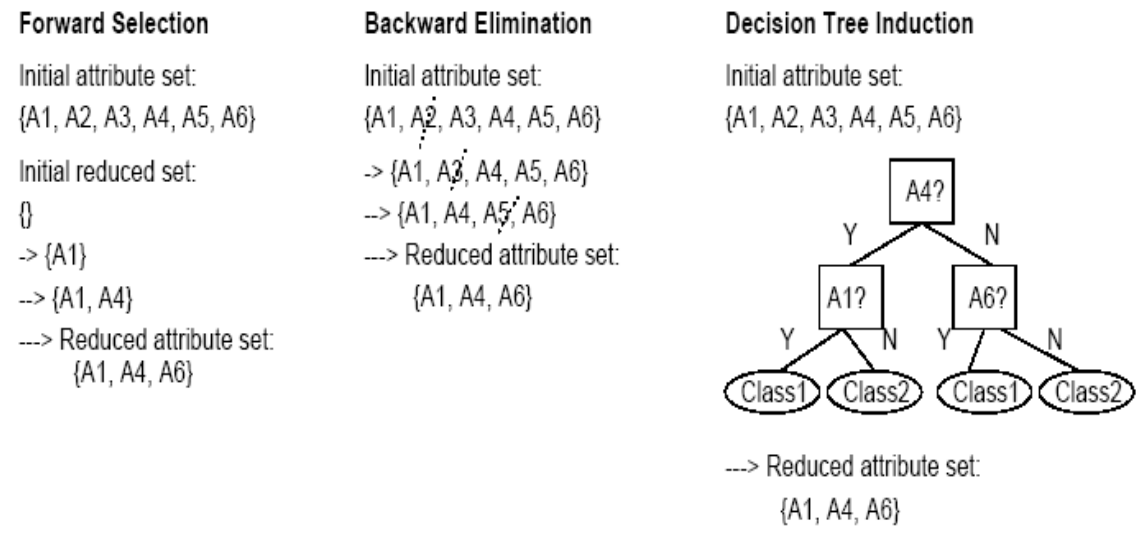


Figure 3.6: Greedy (heuristic) methods for attribute subset selection.

3. Data Compression

- Compressed representation of the original data.
- This data reduction is called as **Lossless** if the original data can be reconstructed from the compressed data without any loss of information.
- The data reduction is called as **Lossy** if only an approximation of the original data can be reconstructed.
- Two Lossy Data Compression methods available are:
  - Wavelet Transforms
  - Principal Components Analysis

3.1 Discrete Wavelet Transform (DWT):

- Is a linear Signal processing technique
- It transforms the data vector X into a numerically different vector X'.
- These two vectors are of the same length.
- Here each tuple is an n-dimensional data vector.
- $X = \{x1,x2,\ldots xn\} \rightarrow$ n attributes
- This wavelet transform data can be truncated.
- Compressed Approximation: Stores only small fraction of strongest of wavelet coefficients.
- Apply inverse of DWT to obtain the original data approximation.
- Similar to discrete Fourier transforms (Signal processing technique involving Sines and Cosines)
- DWT uses Hierarchical Pyramid Algorithm
  - Fast Computational speed o Halves the data in each iteration
  1. The length of the data vector should be an integer power of two. (Padding with zeros can be done if required)
  2. Each transform applies two functions:
     a. Smoothing – sum / weighted average
     b. Difference – weighted difference
  3. These functions are applied to pairs of data so that two sets of data of length L/2 is obtained.
  4. Applies these two transforms iteratively until a user desired data length is obtained.

3.2 Principal Components Analysis (PCA):

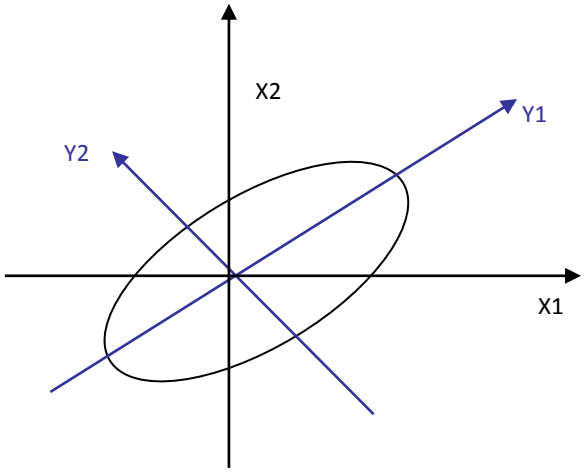
- Say, data to be compressed consists of N tuples and k attributes.
- Tuples can be called as Data vectors and attributes can be called as dimensions.
- So, data to be compressed consists of N data vectors each having k-dimensions.
- Consider a number c which is very very less than N. That is c << N.
- PCA searches for c orthogonal vectors that have k dimensions and that can best be used to represent the data.
- Thus data is projected to a smaller space and hence compressed.
- In this process PCA also combines the essence of existing attributes and produces a smaller set of attributes.
- Initial data is then projected on to this smaller attribute set.

Basic Procedure:

1. Input data Normalized. All attributes values are mapped to the same range.
2. Compute N Orthonormal vectors called as principal components. These are unit vectors perpendicular to each other.
   Thus input data = linear combination of principal components
3. Principal Components are ordered in the decreasing order of "Significance" or strength.
4. Size of the data can be reduced by eliminating the components with less "Significance" or the weaker components are removed. Thus the Strongest Principal Component can be used to reconstruct a good approximation of the original data.

- PCA can be applied to ordered & unordered attributes, sparse and skewed data.
- It can also be applied on multi dimensional data by reducing the same into 2 dimensional data.
- Works only for numeric data.

Principal Component Analysis



4. Numerosity Reduction

- Reduces the data volume by choosing smaller forms of data representations.
- Two types – Parametric, Non-Parametric.
- Parametric – Data estimated into a model
  – only the data parameters stored and not the actual data.
  - Stored data includes outliers also
  - Eg. Log-Linear Models
- Non-Parametric – Do not fits data into models

- Eg. Histograms, Clustering and Sampling

4.1 Regression and Log-Linear Models:

- Linear Regression - data are modeled to fit in a straight line.
- That is data can be modeled to the mathematical equation:

$$Y = \alpha + \beta X,$$

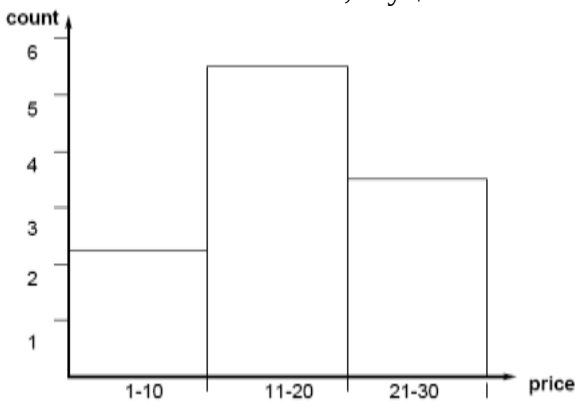
- Where X is called the "Response Variable" and Y is called "Predictor Variable".
- Alpha and beta are called the regression coefficients.
- Alpha is the Y-intercept and Beta is the Slope of the equation.
- These regression coefficients can be solved by using "method of least squares".
- Multiple Regression – Extension of linear regression
  - Response variable Y is modeled as a multidimensional vector.
- Log-Linear Models: Estimates the probability of each cell in a base cuboid for a set of discretized attributes.
- In this higher order data cubes are constructed from lower ordered data cubes.

4.2 Histograms:

- Uses binning to distribute the data.
- Histogram for an attribute A:
  - Partitions the data of A into disjoint subsets / buckets.
  - Buckets are represented in a horizontal line in a histogram.
  - Vertical line of histogram represents frequency of values in bucket.
  - Singleton Bucket – Has only one attribute value / frequency pair
- Eg. Consider the list of prices (in $) of the sold items.

  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.

  - consider a bucket of uniform width, say $10.



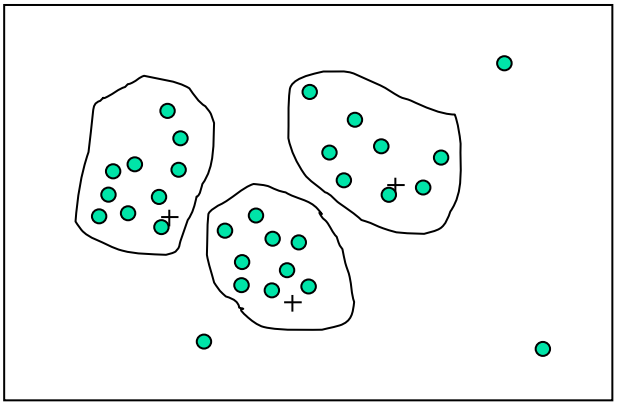
- Methods of determining the bucket / partitioning the attribute values:
  - Equi-Width: Width of each bucket is a constant
  - Equi-Depth: Frequency of each bucket is a constant
  - V-Optimal: Histogram with the least variance
    - Histogram Variance = Weighted sum of values in each bucket
    - Bucket Weight = Number of values in the bucket.
  - MaxDiff: Find the difference between pair of adjacent values. Buckets are formed between the pairs where the difference between the pairs is greater than or equal to b-1, b (Beta) is user specified.
  - V-Optimal & MaxDiff are most accurate and practical.
- Histograms can be extended for multiple attributes – Multidimensional histograms – can capture dependency between attributes.
- Histograms of up to five attributes are found to be effective so far.

- Singleton buckets are useful for storing outliers with high frequency.

4.3 Clustering:

- Considers data tuples as objects.
- Partition objects into clusters.
- Objects within a cluster are similar to one another and the objects in different clusters are dissimilar.
- Quality of a cluster is represented by its 'diameter'
  - maximum distance between any two objects in a cluster.
- Another measure of cluster quality = Centroid Distance = Average distance of each cluster object from the cluster centroid.
- The cluster representation of the data can be used to replace the actual data
- Effectiveness depends on the nature of the data.
- Effective for data that can be categorized into distinct clusters.
- Not effective if data is 'smeared'.
- Can also have hierarchical clustering of data
- For faster data access in such cases we use multidimensional index trees.
- There are many choices of clustering definitions and algorithms available.
-

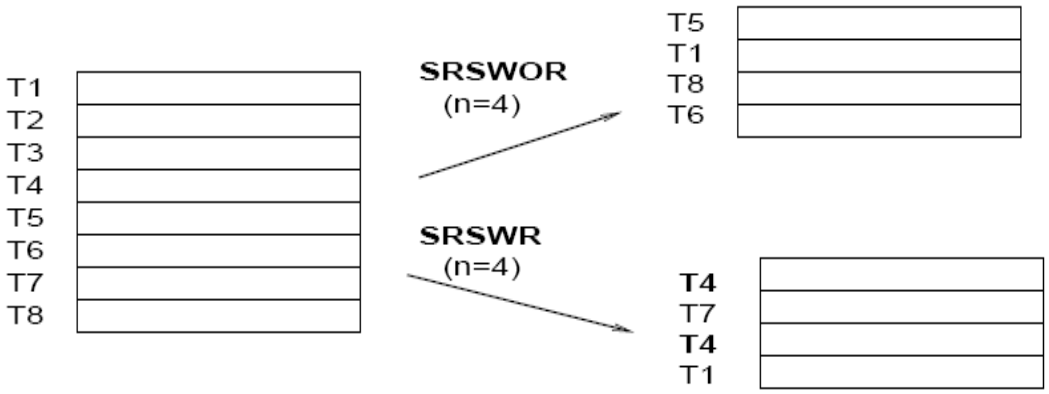
Diagram for clustering



4.4 Sampling:

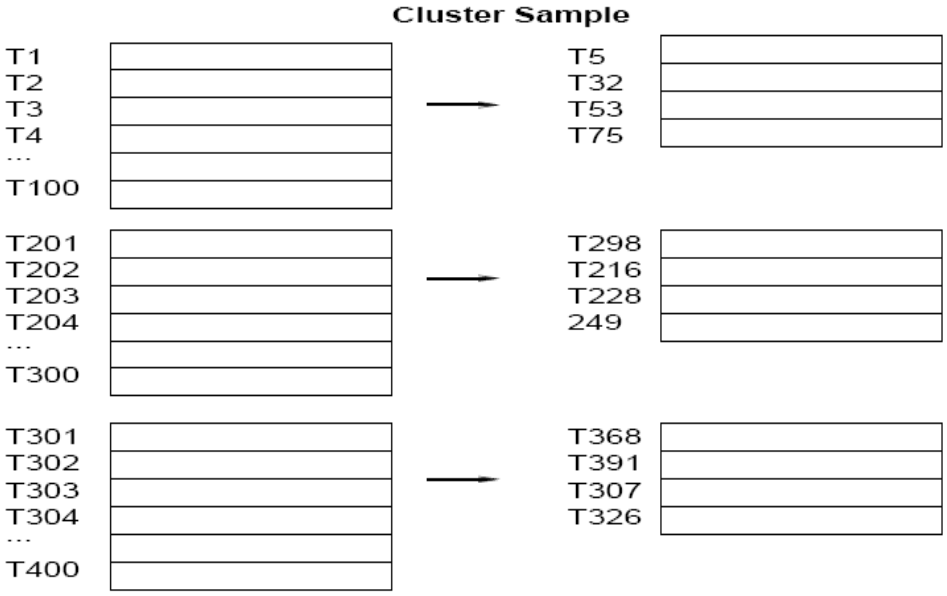
- Can be used as a data reduction technique.
- Selects random sample or subset of data.
- Say large dataset D contains N tuples.

1. Simple Random Sample WithOut Replacement (SRSWOR) of size n:
   - Draw n tuples from the original N tuples in D, where n<N.
   - The probability of drawing any tuple in D is 1/N. That is all tuples have equal chance
2. Simple Random Sample With Replacement (SRSWR) of size n:
   - Similar to SRSWOR, except that each time when a tuple is drawn from D it is recorded and replaced.
   - After a tuple is drawn it is placed back in D so that it can be drawn again.



3. Cluster Sample:

- Tuples in D are grouped into M mutually disjoint clusters.
- Apply SRS (SRSWOR / SRSWR) to each cluster of tuples.
- Each page of data fetching of tuples can be considered as a cluster.



4. Stratified Sample:

- D is divided into mutually disjoint strata.
- Apply SRS (SRSWOR / SRSWR) to each Stratum of tuples.
- In this way the group having the smallest number of tuples is also represented.

Stratified Sample
(according to age)

T38	young
T256	young
T307	young
T391	young
T96	middle-aged
T117	middle-aged
T138	middle-aged
T263	middle-aged
T290	middle-aged
T308	middle-aged
T326	middle-aged
T387	middle-aged
T69	senior
T284	senior

T38	young
T391	young
T117	middle-aged
T138	middle-aged
T290	middle-aged
T326	middle-aged
T69	senior

2.6 Discretization Concept Hierarchies

Data Discretization and Concept Hierarchy Generation:

Data Discretization Technique:

- Reduces the number of attribute values
- Divides the attribute values into intervals
- Interval labels are used to replace the attribute values
- Result – Data easy to use, concise, Knowledge level representation of data

Types of Data Discretization Techniques:

1. Supervised Discretization
   a. Uses Class information of the data
2. Unsupervised Discretization
   a. Does not uses Class information of the data
3. Top-down Discretization (splitting)
   a. Identifies 'Split-Points' or 'Cut-Points' in data values
   b. Splits attribute values into intervals at split-points
   c. Repeats recursively on resulting intervals
   d. Stops when specified number of intervals reached or some stop criteria is reached.
4. Bottom-up Discretization (merging)
   a. Divide the attribute values into intervals where each interval has a distinct attribute value.
   b. Merge two intervals based on some merging criteria
   c. Repeats recursively on resulting intervals
   d. Stops when specified number of intervals reached or some stop criteria is reached.

- Discretization results in – Hierarchical Partitioning of Attributes = Called as Concept Hierarchy
- Concept Hierarchy used for Data Mining at multiple levels of abstraction.
- Eg. For Concept Hierarchy – Numeric values for the attribute Age can be replaced with the class labels 'Youth', 'Middle Aged' and 'Senior'
- Discretization and Concept Hierarchy are pre-processing steps for Data Mining
- For a Single Attribute multiple Concept Hierarchies can be produced to meet various user needs.
- Manual Definition of concept Hierarchies by Domain experts is a tedious and time consuming task.
- Automated discretization methods are available.
- Some Concept hierarchies are implicit at the schema definition level and are defined when the schema is being defined by the domain experts.
- **Eg of Concept Hierarchy using attribute 'Age'**
- Interval denoted by (Y,X] → Value Y (exclusive) and Value X (inclusive)

Discretization and Concept Hierarchy Generation for Numeric Data:

- Concept Hierarchy generation for numeric data is difficult and tedious task as it has wide range of data values and has undergoes frequent updates in any database.
- Automated Discretization Methods:
  - Binning o Histogram analysis
  - Entropy based Discretization Method
  - X2 – Merging (Chi-Merging)
  - Cluster Analysis
  - Discretization by Intuition Partitioning

- These methods assumes the data is in the sorted order

Binning:

- Top-Down Discretization Technique Used
- Un Supervised Discretization Technique – No Class Information Used
- User specified number of bins is used.
- Same technique as used for Smoothing and Numerosity reduction
- Data Discretized using Equi-Width or Equi-Depth method
- Replace each bin value by bin mean or bin median.
- Same technique applied recursively on resulting bins or partitions to generate Concept Hierarchy
- Outliers are also fitted in separate bins or partitions or intervals

Histogram Analysis:

- Un Supervised Discretization; Top-Down Discretization Technique.
- Data Values split into buckets – Equi-Width or Equi-Frequency
- Repeats recursively on resulting buckets to generate multi-level Concept Hierarchies.
- Stops when user specified numbers of Concept Hierarchy Levels are generated.

Entropy-Based Discretization:

- Supervised Discretization Method; Top-Down Discretization Method
- Calculates and determines split-points; The value of the attribute A that has minimum entropy is the split point; Data divided into partitions at the split points
- Repeats recursively on resulting partitions to produce Concept Hierarchy of A
- Basic Method:
  - Consider a database D which has many tuples and A is one of the attribute.

- This attribute A is the Class label attribute as it decides the class of the tuples.
- Attribute value of A is considered as Split-point - Binary Discretization.
  - Tuples with data values of A<= Split-point = D1
  - Tuples with data values of A > Split-point = D2
- Uses Class information. Consider there are two classes of tuples C1 and C2. Then the ideal partitioning should be that the first partition should have the class C1 tuples and the second partition should have the class C2 tuples. But this is unlikely.
- First partition may have many tuples of class C1 and few tuples of class C2 and Second partition may have many tuples of class C2 and few tuples of class C1.
- To obtain a perfect partitioning the amount of Expected Information Requirement is given by the formula:
- **Formula for Info(D) and Entropy(D1)**
- $$I(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} Entropy(D_2) \qquad Entropy(D_1) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$
- Consider that there are m classes and p_i is the probability of class *i* in D1.
- Select a split-point so that it has minimum amount of Expected Information Requirement.
- Repeat this recursively on the resulting partitions to obtain the Concept Hierarchy and stop when the number of intervals exceed the max-intervals (user specified)

X2 Merging (Chi-Merging):

- Bottom-up Discretization; Supervised Discretization
- Best neighboring intervals identified and merged recursively.
- Basic concept is that adjacent intervals should have the same class distribution. If so they are merged, otherwise remain separate.
  - Each distinct value of the numeric attribute = one interval
  - Perform X2 test on each pair of adjacent intervals.
  - Intervals with least X2 value are merged to form a larger interval
  - Low X2 value → Similar class distribution
  - Merging done recursively until pre-specified stop criteria is reached.
- Stop criteria determined by 3 conditions:
  - Stops when X2 value of every pair of adjacent intervals exceeds a pre-specified significance level – set between 0.10 and 0.01
  - Stops when number of intervals exceeds a pre-specified max interval (say 10 to 15)
  - Relative Class frequencies should be consistent within an interval. Allowed level of inconsistency within an interval should be within a pre-specified threshold say 3%.

Cluster Analysis:

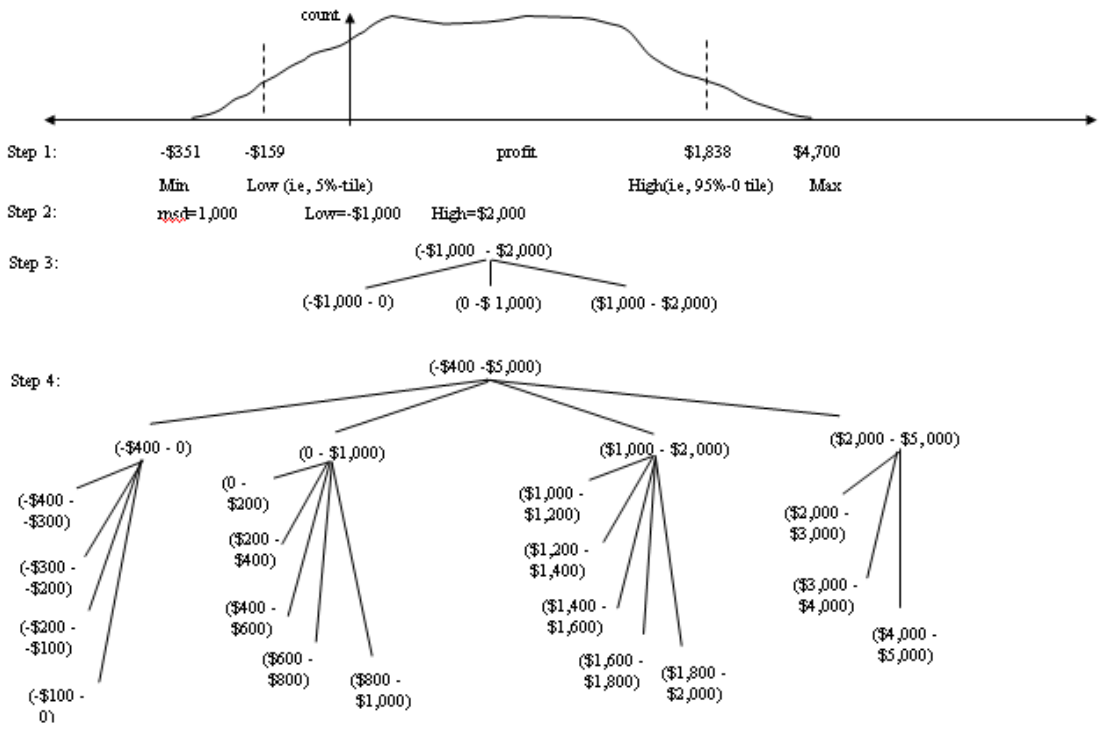
- Uses Top-Down Discretization or Bottom-up Discretization
- Data values of an attribute are partitioned into clusters
- Uses the closeness of data values → Produces high quality discretization results.
- Each cluster is a node in the concept hierarchy

- Each cluster further sub-divided into sub-clusters in case of Top-down approach to create lower level clusters or concepts.
- Clusters are merged in Bottom-up approach to create higher level cluster or concepts.

Discretization by Intuitive Partitioning:

- Users like numerical value intervals to be uniform, easy-to-use, 'Intuitive', Natural.
- Clustering analysis produces intervals such as ($53,245.78,$62,311.78].
- But intervals such as ($50,000,$60,000] is better than the above.
- 3-4-5 Rule:
    - Partitions the given data range into 3 or 4 or 5 equi-width intervals
    - Partitions recursively, level-by-level, based on value range at most significant digit.
    - Real world data can be extremely high or low values which need to be considered as outliers. Eg. Assets of some people may be several magnitudes higher than the others
    - Such outliers are handled separately in a different interval
    - So, majority of the data lies between 5% and 95% of the given data range.
    - Eg. Profit of an ABC Ltd in the year 2004.
    - Majority data between 5% and 95% - (-$159,876,$1,838,761]
    - MIN = -$351,976; MAX = $4,700,896; LOW = -$159,876; HIGH = $1,838,761;
    - Most Significant digit – msd = $1,000,000;
    - Hence LOW' = -$1,000,000 & HIGH' = $2,000,000
    - Number of Intervals = ($2,000,000 – (-$1,000,000))/$1,000,000 = 3.

Example of 3-4-5 Rule



- Hence intervals are: (-$1,000,000,$0], ($0,$1,000,000], ($1,000,000,$2,000,000]
- LOW' < MIN => Adjust the left boundary to make the interval smaller.
- Most significant digit of MIN is $100,000 => MIN' = -$400,000
- Hence first interval reduced to (-$400,000,$0]
- HIGH' < MAX => Add new interval ($2,000,000,$5,000,000]
- Hence the Top tier Hierarchy intervals are:
- (-$400,000,$0],($0,$1,000,000],($1,000,000,$2,000,000],($2,000,000,$5,000,000]
- These are further subdivided as per 3-4-5 rule to obtain the lower level hierarchies.
- Interval (-$400,000,$0] is divided into 4 equi-width intervals
- Intervals ($0,$1,000,000] & is divided into 5 Equi-width intervals
- Interval ($1,000,000,$2,000,000] is divided into 5 Equi-width intervals
- Interval ($2,000,000, $5,000,000] is divided into 3 Equi-width intervals.

Concept Hierarchy Generation for Categorical Data:

Categorical Data = Discrete data; Eg. Geographic Location, Job type, Product Item type

Methods Used:

1. Specification of partial ordering of attributes explicitly at the schema level by users or Experts.
2. Specification of a portion of a hierarchy by explicit data grouping.
3. Specification of the set of attributes that form the concept hierarchy, but not their partial ordering.
4. Specification of only a partial set of attributes.

1. Specification of a partial set of attributes at the schema level by the users or domain experts:

- Eg. Dimension 'Location' in a Data warehouse has attributes 'Street', 'City', 'State' & 'Country'.
- Hierarchical definition of these attributes obtained by ordering these attributes as:
- State < City < State < Country at the schema level itself by user or expert.

2. Specification of a portion of the hierarchy by explicit data grouping:

- Manual definition of concept hierarchy.
- In real time large databases it is unrealistic to define the concept hierarchy for the entire database manually by value enumeration.
- But we can easily specify intermediate-level grouping of data - a small portion of hierarchy.
- For Eg. Consider the sttribute State where we can specify as below:
- {Chennai, Madurai, Trichy} C (Belongs to) Tamilnadu
- {Bangalore, Mysore, Mangalore} C (Belongs to) Karnataka

3. Specification of a set of attributes but not their partial ordering:

- User specifies set of attributes of the concept hierarchy; but omits to specify their ordering
- Automatic concept hierarchy generation or attribute ordering can be done in such cases.
- This is done using the rule that counts and uses the distinct values of each attribute.

- The attribute that has the most distinct values is placed at the bottom of the hierarchy
- And the attribute that has the least distinct values is placed at the top of the hierarchy
- This heuristic rule applies for most cases but it fails of some.
- Users or experts can examine the concept hierarchy and can perform manual adjustment.
- Eg. Concept Hierarchy for 'Location' dimension:
- Country (10); State (508), City (10,804), Street (1,234,567)
- Street < City < State < Country
- In this case user need not modify the generated order / concept hierarchy.
- But this heuristic rule may fail for the 'Time' dimension.
- Distinct Years (100); Distinct Months (12); Distinct Days-of-week (7)
- So in this case the attribute ordering or the concept hierarchy is:
- Year < Month < Days-of-week
- This is not correct.

4. Specification of only partial set of attributes:

- User may have vague idea of the concept hierarchy
- So they just specify only few attributes that form the concept hierarchy.
- Eg. User specifies just the Attributes Street and City.
- To get the complete concept hierarchy in this case we have to link these user specified attributes with the data semantics specified by the domain experts.
- Users have the authority to modify this generated hierarchy.
- The domain expert may have defined that the attributes given below are semantically linked
- Number, Street, City, State, Country.
- Now the newly generated concept hierarchy by linking the domain expert specification and the users specification will be that:
- Number < Street < City < State < Country
- Here the user can inspect this concept hierarchy and can remove the unwanted attribute 'Number' to generate the new Concept Hierarchy as below:
- Street < City < State < Country

2.8 Concept Description: Data Generalization and Summarization Based Characterization

Data generalization and summarization-based characterization

Data and objects in databases often contain detailed information at primitive concept levels. For example, the item relation in a sales database may contain attributes describing low level item information such as item ID, name, brand, category, supplier, place made, and price. It is useful to be able to summarize a large set of data and present it at a high conceptual level. For example, summarizing a large set of items relating to Christmas season sales provides a general description of such data, which can be very helpful for sales and marketing managers. This requires an important functionality in data mining: data generalization.

Data generalization is a process which abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels. Methods for the

e3cient and 3exible generalization of large data sets can be categorized according to two approaches: (1Ñ the data cube approach, and (2Ñ the attribute-oriented induction approach.

Data cube approach for data generalization

In the data cube approach (or OLAP approachÑ to data generalization, the data for analysis are stored in a multidimensional database, or data cube. Data cubes and their use in OLAP for data generalization were described in detail. In general, the data cube approach \materializes data cubes" by 3rst identifying expensive computations required for frequently-processed queries. These operations typically involve aggregate functions, such as count(Ñ, sum(Ñ, average(Ñ, and max(Ñ. The computations are performed, and their results are stored in data cubes. Such computations may be performed for various levels of data abstraction. These materialized views can then be used for decision support, knowledge discovery, and many other applications.

A set of attributes may form a hierarchy or a lattice structure, de3ning a data cube dimension. For example, date may consist of the attributes day, week, month, quarter, and year which form a lattice structure, and a data cube dimension for time. A data cube can store pre-computed aggregate functions for all or some of its dimensions. The precomputed aggregates correspond to speci3ed group-by's of di-erent sets or subsets of attributes.

Generalization and specialization can be performed on a multidimensional data cube by roll-up or drill-down operations. A roll-up operation reduces the number of dimensions in a data cube, or generalizes attribute values to higher level concepts. A drill-down operation does the reverse. Since many aggregate functions need to be computed repeatedly in data analysis, the storage of precomputed results in a multidimensional data cube may ensure fast response time and o-er 3exible views of data from di-erent angles and at di-erent levels of abstraction.

The data cube approach provides an e3cient implementation of data generalization, which in turn forms an important function in descriptive data mining. However, as we pointed out in Section 5.1, most commercial data cube implementations con3ne the data types of dimensions to simple, nonnumeric data and of measures to simple, aggregated numeric values, whereas many applications may require the analysis of more complex data types. More-over, the data cube approach cannot answer some important questions which concept description can, such as which dimensions should be used in the description, and at what levels should the generalization process reach. Instead, it leaves the responsibility of these decisions to the users.

In the next subsection, we introduce an alternative approach to data generalization called attribute-oriented induction, and examine how it can be applied to concept description. Moreover, we discuss how to integrate the two approaches, data cube and attribute-oriented induction, for concept description.

Attribute-oriented induction

The attribute-oriented induction approach to data generalization and summarization-based

characterization was ɜrst proposed in 1989, a few years prior to the introduction of the data cube approach. The data cube approach can be considered as a data warehouse-based, precomputation-oriented, materialized view approach. It performs o−-line aggregation before an OLAP or data mining query is submitted for processing. On the other hand, the attribute-oriented approach, at least in its initial proposal, is a relational database query-oriented, generalization-based, on-line data analysis technique. However, there is no inherent barrier distinguishing the two approaches based on on-line aggregation versus o−-line precomputation. Some aggregations in the data cube can be computed on-line, while o−-line precomputation of multidimensional space can speed up attribute-oriented induction as well. In fact, data mining systems based on attribute-oriented induction, such as DBMiner, have been optimized to include such o−-line precomputation.

Let's ɜrst introduce the attribute-oriented induction approach. We will then perform a detailed analysis of the approach and its variations and extensions.

The general idea of attribute-oriented induction is to ɜrst collect the task-relevant data using a relational database query and then perform generalization based on the examination of the number of distinct values of each attribute in the relevant set of data. The generalization is performed by either attribute removal or attribute generalization (also known as concept hierarchy ascensionÑ. Aggregation is performed by merging identical, generalized tuples, and accumulating their respective counts. This reduces the size of the generalized data set. The resulting generalized relation can be mapped into di−erent forms for presentation to the user, such as charts or rules.

1. Attribute removal is based on the following rule: If there is a large set of distinct values for an attribute of the initial working relation, but either (1Ñ there is no generalization operator on the attribute (e.g., there is no concept hierarchy deɜned for the attributeÑ, or (2Ñ its higher level concepts are expressed in terms of other attributes, then the attribute should be removed from the working relation.

   What is the reasoning behind this rule? An attribute-value pair represents a conjunct in a generalized tuple, or rule. The removal of a conjunct eliminates a constraint and thus generalizes the rule. If, as in case 1, there is a large set of distinct values for an attribute but there is no generalization operator for it, the attribute should be removed because it cannot be generalized, and preserving it would imply keeping a large number of disjuncts which contradicts the goal of generating concise rules. On the other hand, consider case 2, where the higher level concepts of the attribute are expressed in terms of other attributes. For example, suppose that the attribute in question is street, whose higher level concepts are represented by the attributes hcity, province or state, countryi. The removal of street is equivalent to the application of a generalization operator. This rule corresponds to the generalization rule known as dropping conditions in the machine learning literature on learning-from-examples.

2. Attribute generalization is based on the following rule: If there is a large set of distinct values for an attribute in the initial working relation, and there exists a set of

generalization operators on the attribute, then a generalization operator should be selected and applied to the attribute.

This rule is based on the following reasoning. Use of a generalization operator to generalize an attribute value within a tuple, or rule, in the working relation will make the rule cover more of the original data tuples, thus generalizing the concept it represents. This corresponds to the generalization rule known as climbing generalization trees in learning-from-examples.

Both rules, attribute removal and attribute generalization, claim that if there is a large set of distinct values for an attribute, further generalization should be applied. This raises the question: how large is \a large set of distinct values for an attribute" considered to be?

Depending on the attributes or application involved, a user may prefer some attributes to remain at a rather low abstraction level while others to be generalized to higher levels. The control of how high an attribute should be generalized is typically quite subjective. The control of this process is called attribute generalization control. If the attribute is generalized \too high", it may lead to over-generalization, and the resulting rules may not be very informative. On the other hand, if the attribute is not generalized to a \su3ciently high level", then under-generalization may result, where the rules obtained may not be informative either. Thus, a balance should be attained in attribute-oriented generalization.

There are many possible ways to control a generalization process. Two common approaches are described below.

The 3rd technique, called attribute generalization threshold control, either sets one generalization thresh-old for all of the attributes, or sets one threshold for each attribute. If the number of distinct values in an attribute is greater than the attribute threshold, further attribute removal or attribute generalization should be performed. Data mining systems typically have a default attribute threshold value (typically ranging from 2 to 8Ñ, and should allow experts and users to modify the threshold values as well. If a user feels that the gen-eralization reaches too high a level for a particular attribute, she can increase the threshold. This corresponds to drilling down along the attribute. Also, to further generalize a relation, she can reduce the threshold of a particular attribute, which corresponds to rolling up along the attribute.

The second technique, called generalized relation threshold control, sets a threshold for the generalized relation. If the number of (distinctÑ tuples in the generalized relation is greater than the threshold, further generalization should be performed. Otherwise, no further generalization should be performed. Such a threshold may also be preset in the data mining system (usually within a range of 10 to 30Ñ, or set by an expert or user, and should be adjustable. For example, if a user feels that the generalized relation is too small, she can increase the threshold, which implies drilling down. Otherwise, to further generalize a relation, she can reduce the threshold, which implies rolling up.

These two techniques can be applied in sequence: 3rst apply the attribute threshold control technique to generalize each attribute, and then apply relation threshold control to further reduce the size of the generalized relation.

Notice that no matter which generalization control technique is applied, the user should be allowed to adjust the generalization thresholds in order to obtain interesting concept descriptions. This adjustment, as we saw above, is similar to drilling down and rolling up, as discussed under OLAP operations. However, there is a methodological distinction between these OLAP operations and attribute-oriented induction. In OLAP, each step of drilling down or rolling up is directed and controlled by the user; whereas in attribute-oriented induction, most of the work is performed automatically by the induction process and controlled by generalization thresholds, and only minor adjustments are made by the user after the automated induction.

In many database-oriented induction processes, users are interested in obtaining quantitative or statistical in-formation about the data at di−erent levels of abstraction. Thus, it is important to accumulate count and other aggregate values in the induction process. Conceptually, this is performed as follows. A special measure, or numerical attribute, that is associated with each database tuple is the aggregate function, count. Its value for each tuple in the initial working relation is initialized to 1. Through attribute removal and attribute generalization, tuples within the initial working relation may be generalized, resulting in groups of identical tuples. In this case, all of the identical tuples forming a group should be merged into one tuple. The count of this new, generalized tuple is set to the total number of tuples from the initial working relation that are represented by (i.e., were merged intoÑ the new generalized tuple. For example, suppose that by attribute-oriented induction, 52 data tuples from the initial working relation are all generalized to the same tuple, T . That is, the generalization of these 52 tuples resulted in 52 identical instances of tuple T . These 52 identical tuples are merged to form one instance of T , whose count is set to 52. Other popular aggregate functions include sum and avg. For a given generalized tuple, sum contains the sum of the values of a given numeric attribute for the initial working relation tuples making up the generalized tuple. Suppose that tuple T contained sum(units soldÑ as an aggregate function. The sum value for tuple T would then be set to the total number of units sold for each of the 52 tuples. The aggregate avg (averageÑ is computed according to the formula, avg = sum♪count.

Generalized descriptions resulting from attribute-oriented induction are most commonly displayed in the form of a generalized relation, such as the generalized relation.

2.9 Mining Association Rules in Large Databases

Mining Association Rules from large databases:

Basic Concepts and a Road Map:

Frequent Patterns:

- Patterns that appear in a dataset frequently.
- Eg. Milk and Bread are purchased together mostly

Frequent Sequential Patterns:

- Sequence of patterns that appear frequently in a dataset.
- Eg. Purchase of Computer followed by purchase of Digital Camera followed by purchase of Memory Card – This sequence of pattern found mostly.

Frequent Structural Patterns:

- Structural forms such as graphs, trees and la ttices appear frequently in a dataset.

Frequent Pattern Mining:

- Discovery of interesting associations & correlations between item sets in transactional and relational databases.

Market Basket Analysis – A Motivating Example:

- Analyzes customer buying habits by finding associations between different items that customers place in their "Shopping Baskets".
- Discovery of such associations helps retailers to:
  - Develop Marketing Strategy o Increase Sales o Plan Shelf Space
  - Decide on which items to put on sale at reduced prices.
    - Eg. Sale on Printers allowed when Computers are purchased.
- These patterns can be represented in the form of association rules.
  - For Eg. The customers who purchase computers also tend to buy printers at the same time is represented in association rule as below:
    - Computer => Printers [Support = 2%; Confidence = 60%]
- Support and Confidence are two measures of rule interestingness.
- Support – reflects usefulness of discovered rule.
- Confidence – reflects certainty of discovered rule.
- Support = 2% => 2% of transactions satisfies the rule "Printers are purchased when computers are sold"
- Confidence = 60% => 60% of customers who purchased a computer also purchased printers.
- Association rules are interesting if they satisfy:
  - Minimum Support Threshold → Set by users or domain experts
  - Minimum Confidence Threshold → Set by users or domain experts

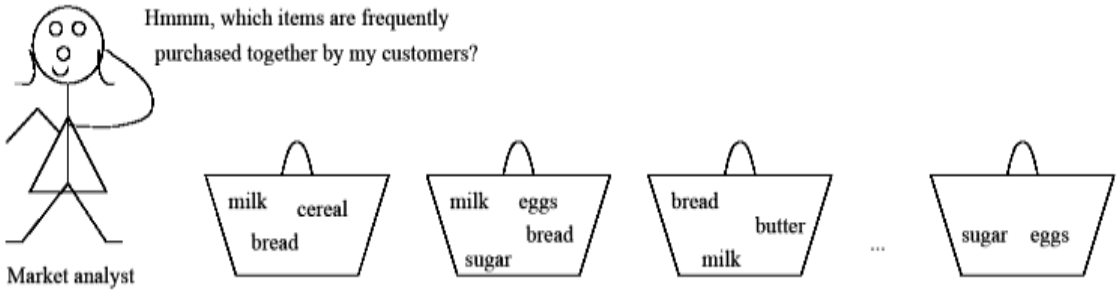


Figure 6.1: Market basket analysis.

Basic Concepts:

- Let I = {I1,I2,…In} = set of items
- D = {T1,T2,…Tn} = Set of Transactions
- Each Tj is a set of items such that T is the subset of I
- Each transaction has a unique identifier TID
- Let A = set of items
- Then T is said to contain A if and only if A is a subset of T
- Association Rule is of the form:
  - $A \Rightarrow B$, where $A \subset \mathcal{I}$, $B \subset \mathcal{I}$ and $A \cap B = \phi$
- The Association rule A => B has support **s** in the transaction set D; where s is the % of transactions in D that contains $A \cup B$.
- The Association rule A => B has confidence **c** in the transaction set D; where c is the % of transactions in D containing A which also contains B.

- That is:

$$support(A \Rightarrow B) = Prob\{A \cup B\}$$
$$confidence(A \Rightarrow B) = Prob\{B|A\}.$$

- Association rules that satisfy both a minimum support threshold (min-sup) and minimum confidence threshold (min-conf) are called Strong Association rules.
- A set of items is referred to as an item set.
- An item set that contains k items is called a k-item set.
- The set {Computer, Printer} is a 2-itemset
- Frequency / Support Count of an Item set:
  - Number of transactions that contain the item set.
- Frequent Item set:
  - If an item set satisfies minimum support (min-support) then it is a frequent item set.

How are association rule mined from large databases?

- Step1: Find all frequent item sets
- Step 2: Generate strong association rules from the frequent item sets.

Association Rule Mining – A road map:

- One form of association rule mining is "Market Basket Analysis"
- Many kinds of association rules exist.
- Association rules classified based on the following criteria:
  - 1) Based on the type of values handled in the rule:
    - a) Boolean Association Rule:
      - Association between presence or absence of items.
      - Eg. Computer => Printer [Support = 2%; Confidence = 60%]
    - b) Quantitative Association Rule:
      - Association between quantitative attributes
      - Eg. In the below Age and Income are quantitative attributes.

$$age(X, \text{“}30-34\text{”}) \wedge income(X, \text{“}42K-48K\text{”}) \Rightarrow buys(X, \text{“}high\ resolution\ TV\text{”})$$

  - 2) Based on the dimensions of data involved in the rule:
    - a) Single-Dimensional Association Rule:
      - If attributes in an association rule refer to only one dimension
      - Eg. $buys(X, \text{“}computer\text{”}) \Rightarrow buys(X, \text{“}financial\_management\_software\text{”})$
    - b) Multi-Dimensional Association Rule:
      - Attributes in the association rule refer to two or more dimensions.
      - Eg. In the below it involves 3 dimensions Age, Income & Buys.

$$age(X, \text{“}30-34\text{”}) \wedge income(X, \text{“}42K-48K\text{”}) \Rightarrow buys(X, \text{“}high\ resolution\ TV\text{”})$$

  - 3) Based on the levels of abstractions involved in the rule:
    - a) Multi-Level Association Rule
      - Association rule refers to items at different levels of abstraction.
      - Eg. $age(X, \text{“}30-34\text{”}) \Rightarrow buys(X, \text{“}laptop\ computer\text{”})$
      - $age(X, \text{“}30-34\text{”}) \Rightarrow buys(X, \text{“}computer\text{”})$

- In the above, Computer is at the higher level of abstraction and Laptop Computer at the lower level of abstraction.

▪ b) Single-Level Association Rule

- Association rule refers to items at same level of abstraction

o 4) Based on the nature of association involved in the rule:

▪ Association mining extended to Correlation Analysis where the absence or presence of correlated items can be identified.

Apriori Algorithm:

- This is a basic algorithm for:
  - Finding Frequent Item sets
  - Generate Strong Association Rules from Frequent Item sets.

Finding Frequent Item sets:

- This algorithm uses prior knowledge of frequency item set properties.
- Uses Level-wise search – Iterative approach
- Here k-item sets are used to explore k+1 item sets
- 1) First set of frequent 1-Item sets are found. This set is denoted by L1.
- 2) L1 is used to find L2 = Frequent 2-Item sets
- 3) L2 is used to find L3 and so on.
- 4) Stop when no more frequent k-item sets can be found.
- Finding of each Lk requires one full scan of the database.
- At every level the algorithm prunes the sets that are unlikely to be frequent sets.
- Each pass (Except the first pass) has 2 phases:
  - (i) Candidate Generation Process
  - (ii) Pruning Process
- This Algorithm uses the Apriori Property:
  - "All non-empty subsets of a frequent itemset must also be frequent"

(i)Candidate Generation Process:

- To find Lk, the candidate set Ck has to be generated by joining Lk-1 with itself.
- The members of Lk-1 are joinable if they have (k-2) items in common.
- $L_{k-1} \bowtie L_{k-1} = \{A \bowtie B | A, B \in L_{k-1}, |A \cap B| = k-2\}$
- Eg. L3 = { {1,2,3}, {1,2,5}, {1,3,5}, {2,3,5}, {2,3,4} }
- Then C4 = { {1,2,3,5}, {2,3,4,5} }
- {1,2,3,5} is generated from {1,2,3} & {1,2,5}
- {2,3,4,5} is generated from {2,3,4} & {2,3,5}

(ii)Pruning Process:

- This process removes the items from Ck which does not satisfy the Apriori Property.
- First find all k-1subsets of each item of Ck
- Then check if these subsets are in Lk-1. If for any item of Ck any of its subsets are not in Lk-1 then remove that item from Ck.
- **Prune (Ck)**

```
For all c in Ck
        For all (k-1) subset d of c do
                If d not in Lk-1 then
                        Ck = Ck \ {c}
                End If
        End do
End do
```

- Eg: C4 = { {1,2,3,5}, {2,3,4,5} }

- 3 subsets of {1,2,3,5} = {1,2,3}, {2,3,5}, {1,3,5}
- All these subsets are in L3 hence do not remove {1,2,3,5} from C4
- 3 subsets of {2,3,4,5} = {2,3,4}, {2,4,5}, {3,4,5}
- The subsets {2,4,5} & {3,4,5} are not in L3. Hence remove {2,3,4,5} from C4.
- Thus this algorithm uses Candidate Generation and Pruning Process at every iteration.
- Move upwards from level 1 to level k where no candidate set remains after pruning.

Algorithm:

Initialize k=1; C1 = all the 1-itemsets;
Read the database to count the support of C1 to determine L1;
L1 = {Frequent 1-itemsets};
k=2; // k is pass number //
While (Lk-1 <> Empty) do
Begin
    Ck = Generate Candidate item sets with the given Lk-1;
    Prune (Ck);
    For all transactions t in T do
        Increment the count of all candidates in Ck that are contained in t;
    End do;
    Lk = All candidates in Ck with minimum support;
    k = k + 1
End;
L = Uk Lk = L1 U L2 U…ULk;

Example:

A Transactional database where |D| = 4

AllElectronics database

TID	List of item_ID's
T100	I1, I2, I5
T200	I2, I3, I4
T300	I3, I4
T400	I1, I2, I3, I4

k = 1; Let minimum support = 2 = 50% → User specified

Scan D for count of each candidate →

C_1

Itemset	Sup.
{I1}	2
{I2}	3
{I3}	3
{I4}	3
{I5}	1

Compare candidate support with minimum support count →

L_1

Itemset	Sup.
{I1}	2
{I2}	3
{I3}	3
{I4}	3

k = 2; After pruning C2 is same.

Generate C_2 candidates from L_1 →

C_2

Itemset
{I1,I2}
{I1,I3}
{I1,I4}
{I2,I3}
{I2,I4}
{I3,I4}

Scan D for count of each candidate →

C_2

Itemset	Sup.
{I1,I2}	2
{I1,I3}	1
{I1,I4}	1
{I2,I3}	2
{I2,I4}	2
{I3,I4}	3

Compare candidate support with minimum support count →

L_2

Itemset	Sup.
{I1,I2}	2
{I2,I3}	2
{I2,I4}	2
{I3,I4}	3

k = 3
Generate C3 candidates from L2

C3 = { {I1, I2, I3}, {I1, I2, I4}, {I2, I3, I4} }

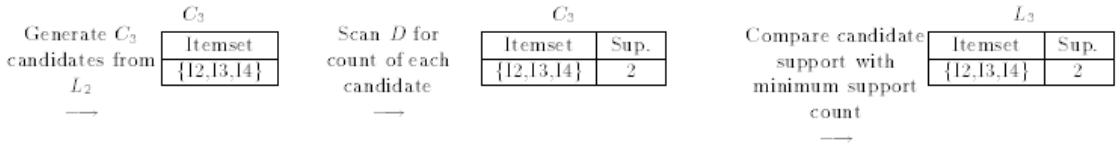
Pruning Step:

{I1, I2, I3} Subsets are {I1, I2}, {I1, I3}, {I2, I3} → Not all subsets in L2

{I1, I2, I4} Subsets are {I1, I2}, {I1, I4}, {I2, I4} → Not all subsets in L2

{I2, I3, I4} Subsets are {I2, I3}, {I2, I4}, {I3, I4} → All subsets are in L2

After Pruning: C3 = { {I2, I3, I4} }



k = 4; L3 has 1 element => C4 is empty => Algorithm stops

L = L1 U L2 U L3 = { {I1}, {I2}, {I3}, {I4}, {I1, I2}, {I2, I3}, {I2, I4}, {I3, I4}, {I2, I3, I4} }

Generating Association Rules from Frequent Item sets:

- Generate strong association rules from the generated frequent item sets (Previous step).
- Strong Association rules satisfy both minimum support and minimum confidence.
- Confidence Equation is :

  Confidence (A => B) = Prob (B/A) = Support (A U B) / Support (A)

- Support (A U B) = Number of transactions containing the itemsets AUB
- Support (A) = Number of transactions containing the item set A
- Based on this equation, Association rules can be generated as follows:
  - 1) For each frequent item set, l generate all non-empty subsets of l
  - 2) For every non-empty subset s of l, output the rule
    - "s => (l-s)" If Support (l) / Support (s) >= min_conf
    - min-conf = Minimum Confidence threshold
- Eg: Consider the frequent item set l = {I2, I3, I4}
  - 1) Non-empty subsets of l are:
    - {I2, I3}, {I2, I4}, {I3, I4}, {I2}, {I3} and {I4}
  - The resulting Association rules are:

$$I2 \wedge I3 \Rightarrow I4, \quad confidence = 2/2 = 100\%$$
$$I2 \wedge I4 \Rightarrow I3, \quad confidence = 2/2 = 100\%$$
$$I3 \wedge I4 \Rightarrow I2, \quad confidence = 2/3 = 67\%$$
$$I2 \Rightarrow I3 \wedge I4, \quad confidence = 2/3 = 67\%$$
$$I3 \Rightarrow I2 \wedge I4, \quad confidence = 2/3 = 67\%$$
$$I4 \Rightarrow I2 \wedge I3, \quad confidence = 2/3 = 67\%$$

  - Let the minimum confidence threshold = 70%; then the strong Association rules are the first and second rules.

Review Questions

Two Marks:

1. Why do we go for data preprocessing?
2. Why is data dirty?
3. Why is data preprocessing important?
4. What are the multi-dimensional measures of data quality?
5. List the various tasks in data preprocessing.
6. List down the data transformation processes. Write about normalization by decimal scaling.
7. What are Min-Max Normalization and Z-score Normalization?
8. Why data reduction? What is data reduction?
9. What is data discretization technique? List the types of data discretization techniques.
10. What are the various concept hierarchy generation methods for categorical data?
11. What are frequent patterns? What are frequent sequential patterns? What are frequent structural patterns? What is frequent pattern mining?
12. Write about Support and Confidence for Association rule mining.
13. List the criteria upon which association rules are classified.

Sixteen Marks:

1. (i) What are the various data cleaning tasks and explain on the methods of handling those? Describe about the data cleaning process steps. (8)
   (ii) What are the various data integration tasks and explain on the methods of handling those? (8)
2. (i) Explain about the various data reduction strategies. (8)
   (ii) Detail on the different data compression techniques. (8)
3. Write brief notes on numerosity reduction. (16)
   (i) Regression and log-linear models
   (ii) Histograms
   (iii) Clustering
   (iv) Sampling
4. Explain about the various discretization and concept hierarchy generation techniques for numeric data. (16)
5. Explain the Apriori algorithm with a suitable example for finding frequent itemsets and how to generate strong association rules from frequent itemsets. (16)

Assignment Topic:

1. Write about “Data Generalization and Summarization Based Characterization”.