

Data Mining: Concepts and Techniques

— Chapter 2 —

Jiawei Han, Micheline Kamber, and Jian Pei

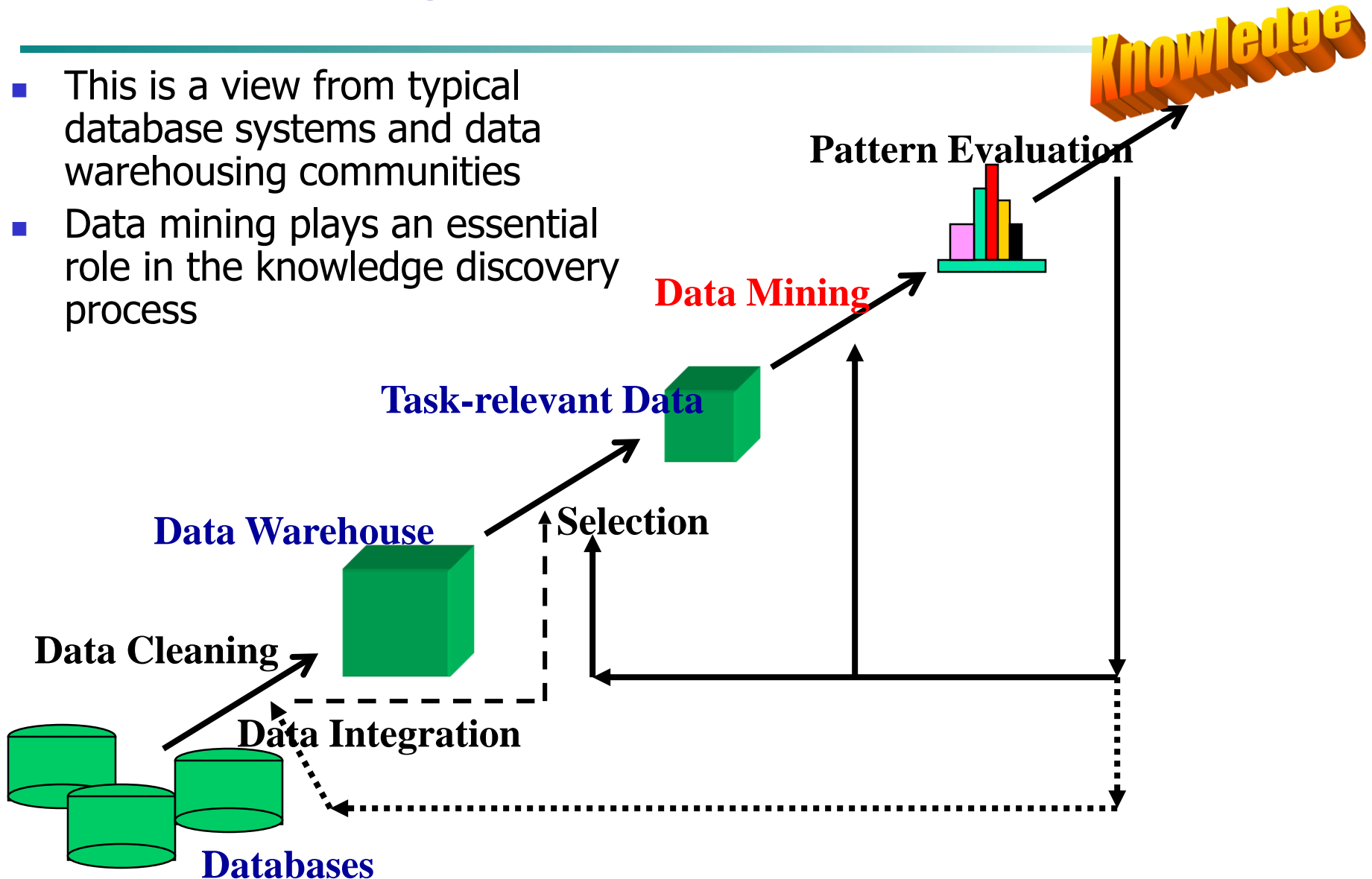
University of Illinois at Urbana-Champaign

Simon Fraser University

©2011 Han, Kamber, and Pei. All rights reserved.

Knowledge Discovery (KDD) Process

- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Why Data Preprocessing?

- Data in the real world is dirty
 - **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - e.g., occupation=""
 - **noisy**: containing errors or outliers
 - e.g., Salary="-10"
 - **inconsistent**: 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"

Why Is Data Dirty?

- 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
- Duplicate records also need data cleaning

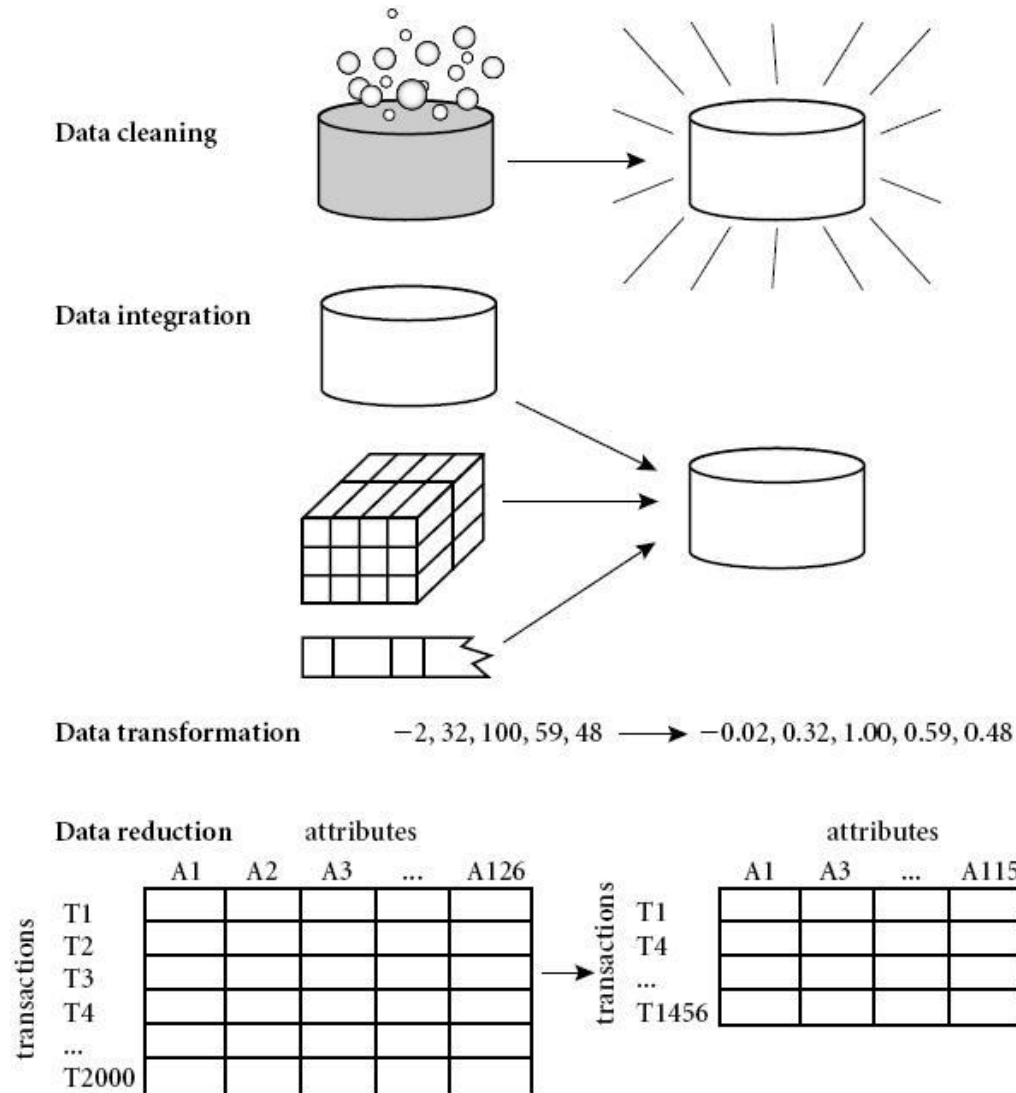
Why Is Data Preprocessing Important?

- 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

Major 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 but with particular importance, especially for numerical data

Forms of Data Preprocessing



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Mining Data Descriptive Characteristics

- Motivation
 - To better understand the data: central tendency, variation and spread
- Data dispersion characteristics
 - median, max, min, quantiles, outliers, variance, etc.
- Data Mining Point of view:
 - How to compute these efficiently in large databases

Measuring the Central Tendency

- Mean: An algebraic measure

- Weighted arithmetic mean:
- Trimmed mean: chopping extreme values

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \mu = \frac{\sum x}{N}$$

$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

- Median: A holistic measure

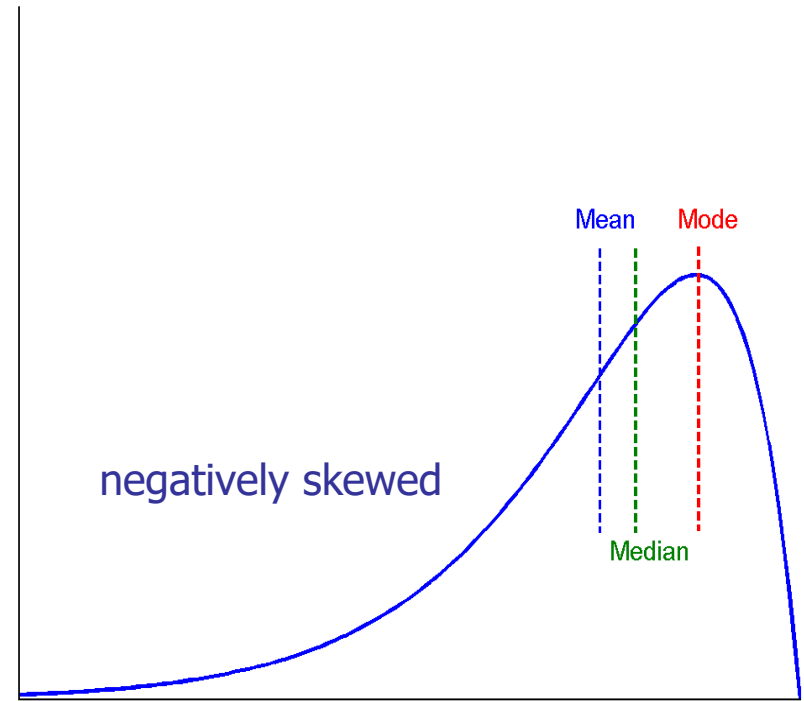
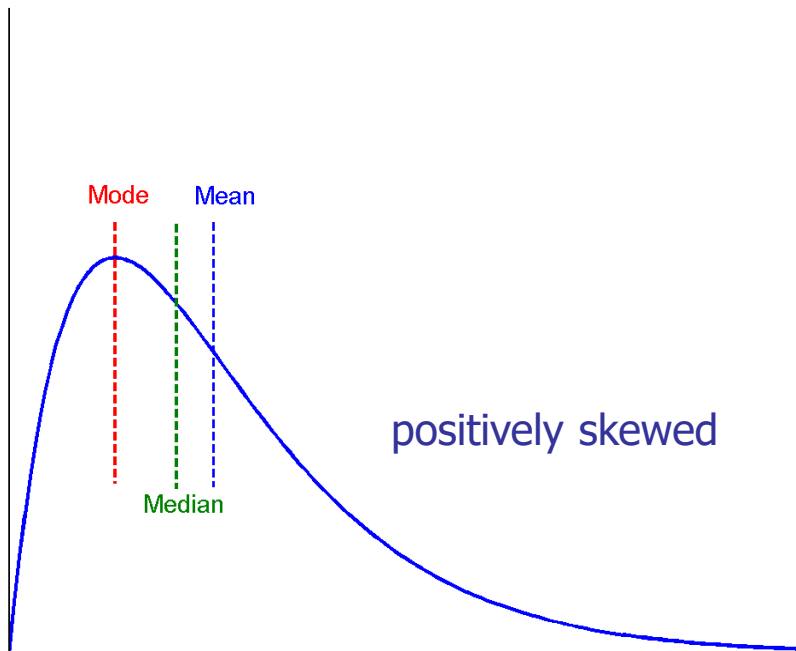
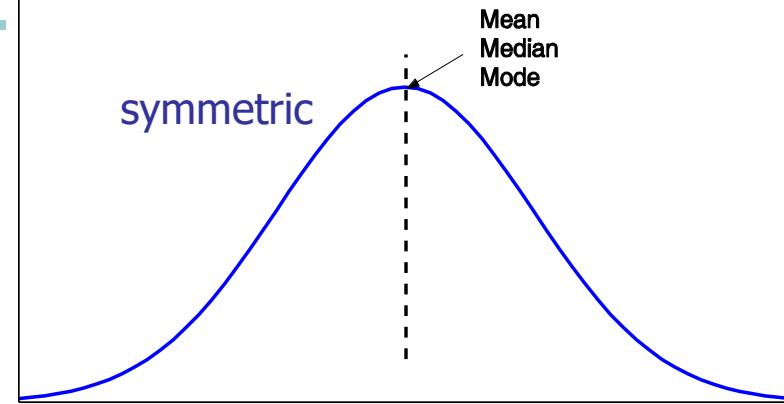
- Middle value if odd number of values, or average of the middle two values otherwise
- Estimated by interpolation for *grouped data*

- Mode

- Value that occurs most frequently in the data
- Unimodal, bimodal, trimodal

Symmetric vs. Skewed Data

- Median, mean and mode of symmetric, positively and negatively skewed data



Measuring the Dispersion of Data

- Quartiles, outliers and boxplots

- **Quartiles:** Q_1 (25th percentile), Q_3 (75th percentile)
- **Inter-quartile range:** $IQR = Q_3 - Q_1$
- **Five number summary:** min, Q_1 , median, Q_3 , max
- **Boxplot:** ends of the box are the quartiles; median is marked; add whiskers, and plot outliers individually
- **Outlier:** usually, a value higher/lower than $1.5 \times IQR$

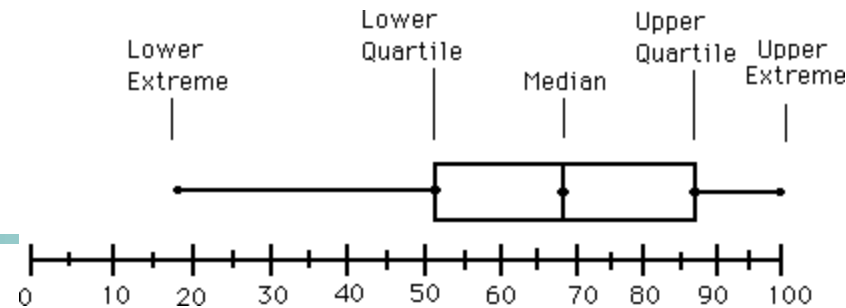
- Variance and standard deviation (*sample: s , population: σ*)

- **Variance:** (algebraic, scalable computation)

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{n-1} \left[\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \right] \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^n x_i^2 - \mu^2$$

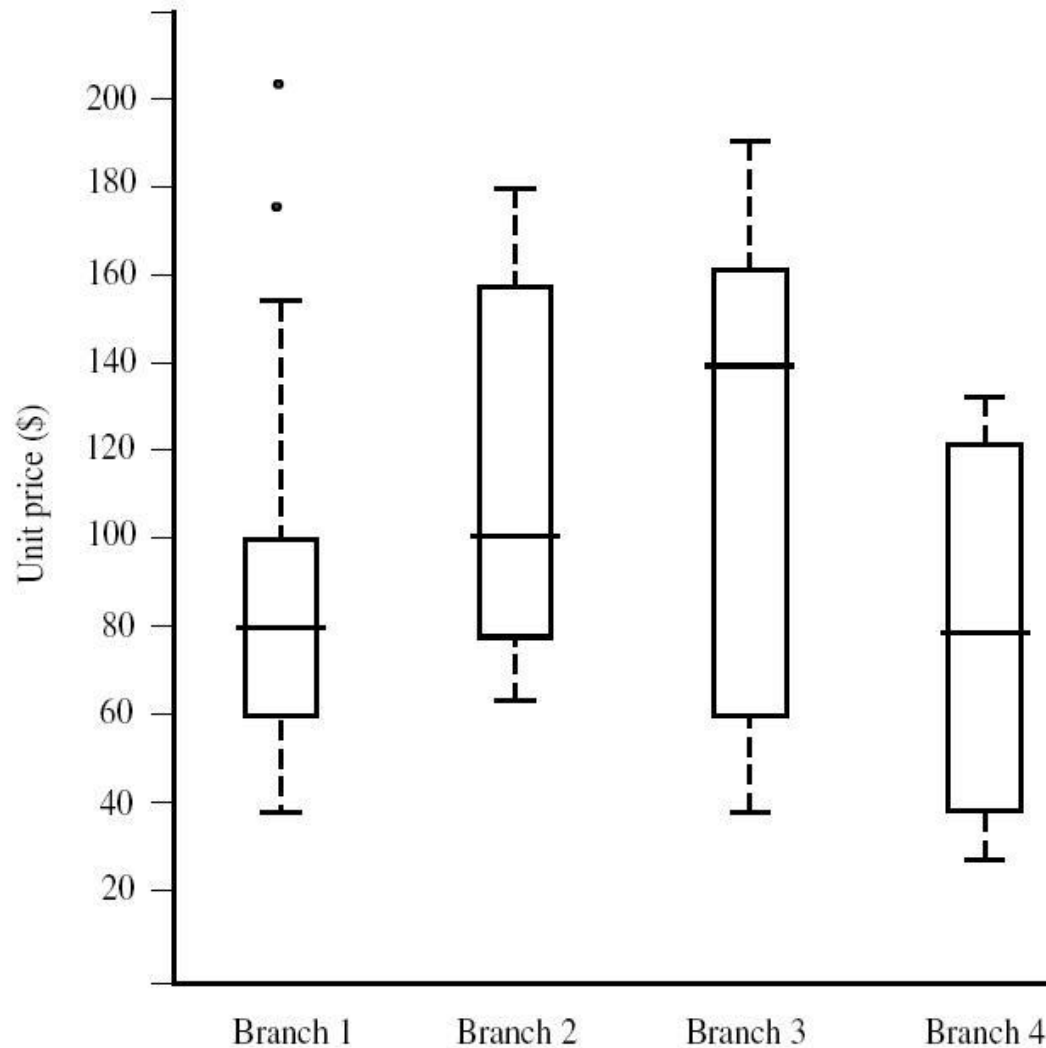
- **Standard deviation** s (*or* σ) is the square root of variance s^2 (*or* σ^2)

Boxplot Analysis



- **Five-number summary** of a distribution:
Minimum, Q1, M, Q3, Maximum
- **Boxplot**
 - Data is represented with a box
 - The ends of the box are at the first and third quartiles, i.e., the height of the box is IQR
 - The median is marked by a line within the box
 - Whiskers: two lines outside the box extend to Minimum and Maximum

Boxplot Analysis



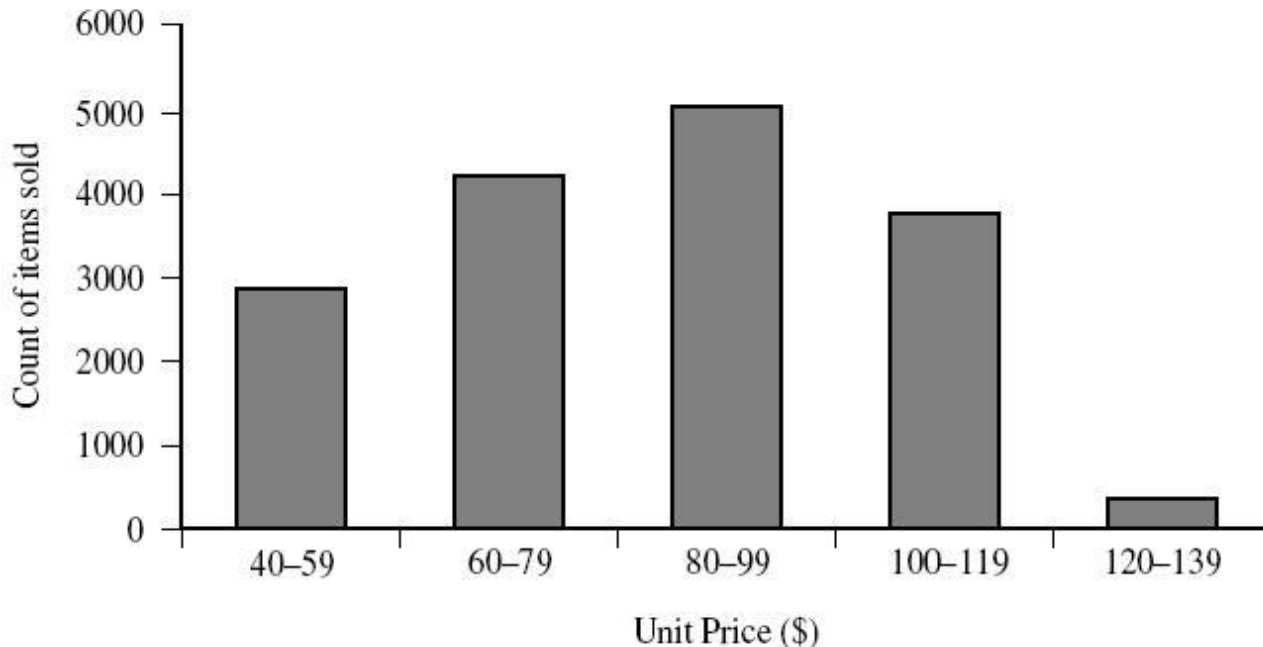
Example

A set of unit price data for items sold at a branch of *AllElectronics*.

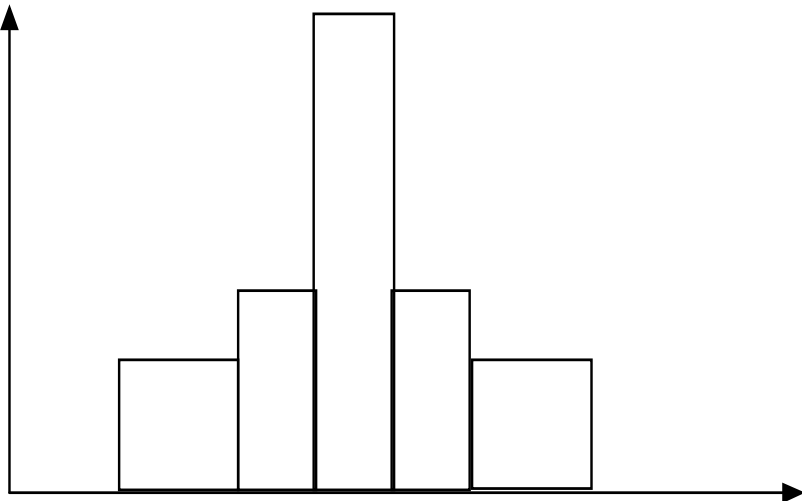
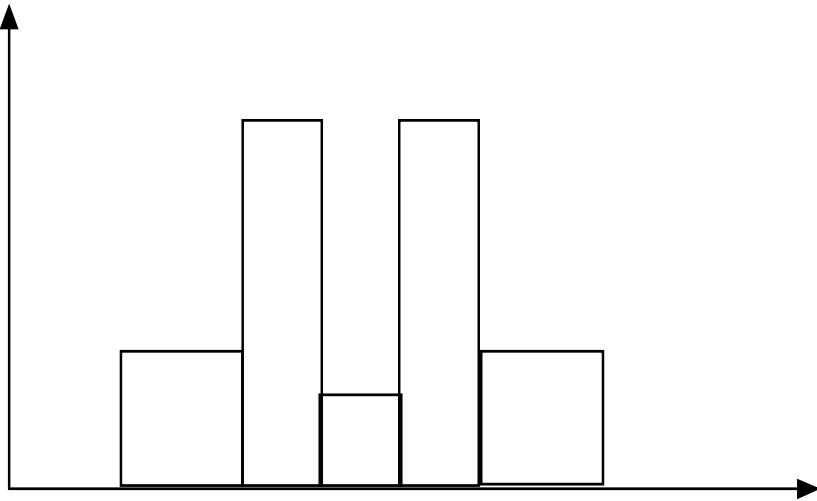
<i>Unit price (\$)</i>	<i>Count of items sold</i>
40	275
43	300
47	250
..	..
74	360
75	515
78	540
..	..
115	320
117	270
120	350

Histogram Analysis

- Graph displays of basic statistical class descriptions
 - Frequency histograms
 - A univariate graphical method
 - Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data



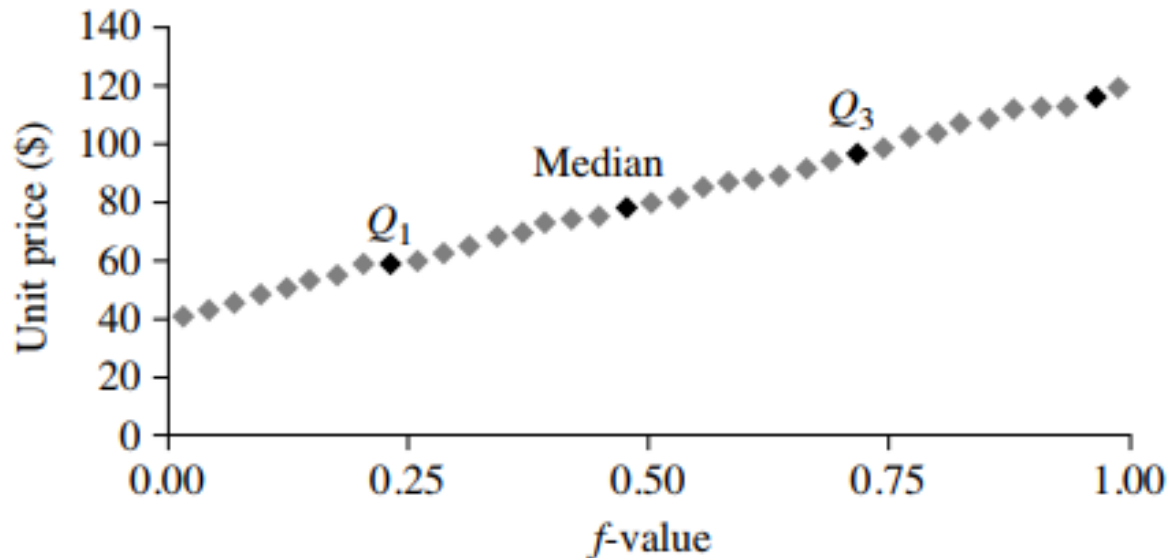
Histograms Often Tell More than Boxplots



- The two histograms shown in the left may have the same boxplot representation
 - The same values for: min, Q1, median, Q3, max
- But they have rather different data distributions

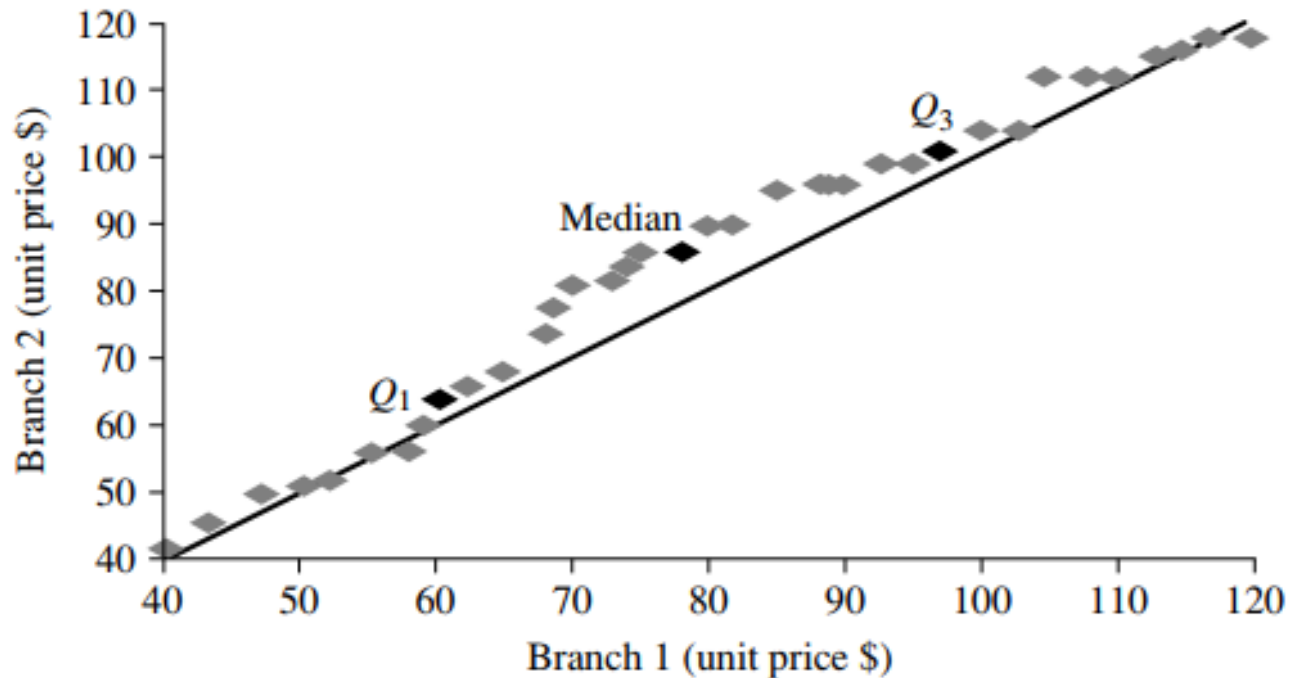
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots **quantile** information
 - For a data x_i data sorted in increasing order, f_i indicates that approximately 100 f_i % of the data are below or equal to the value x_i



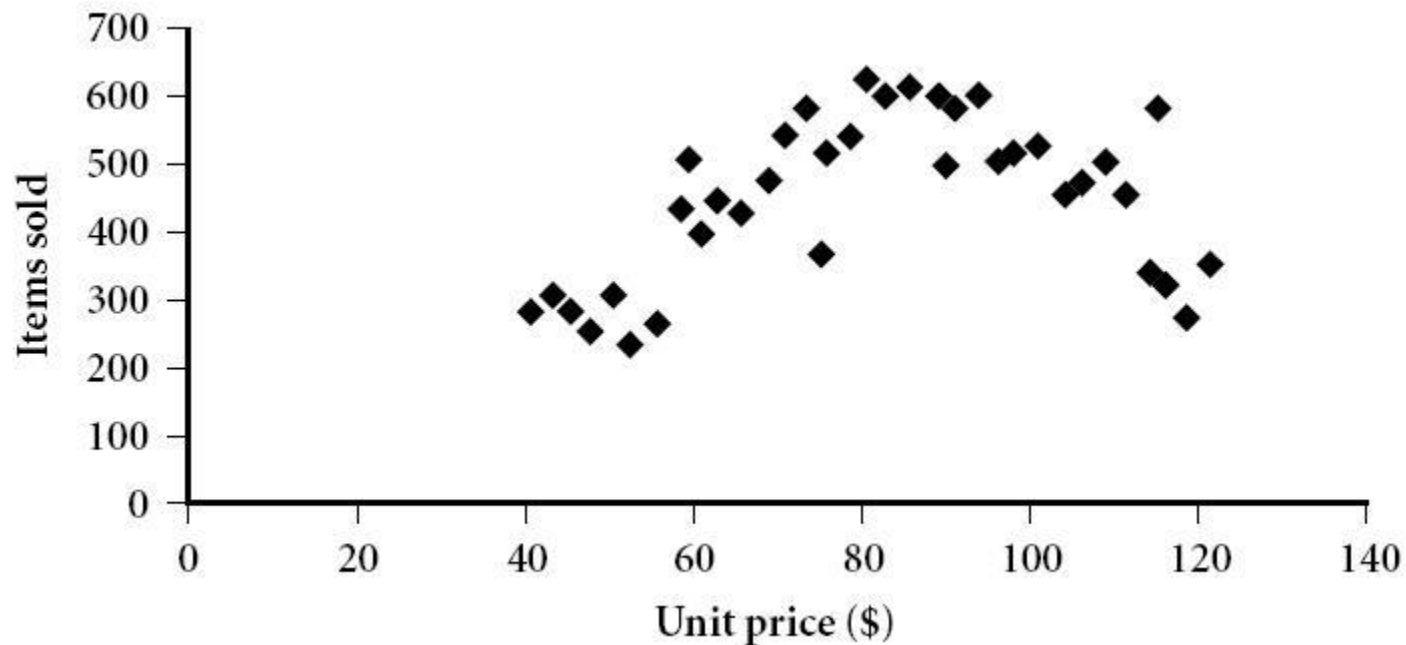
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Allows the user to view whether there is a shift in going from one distribution to another



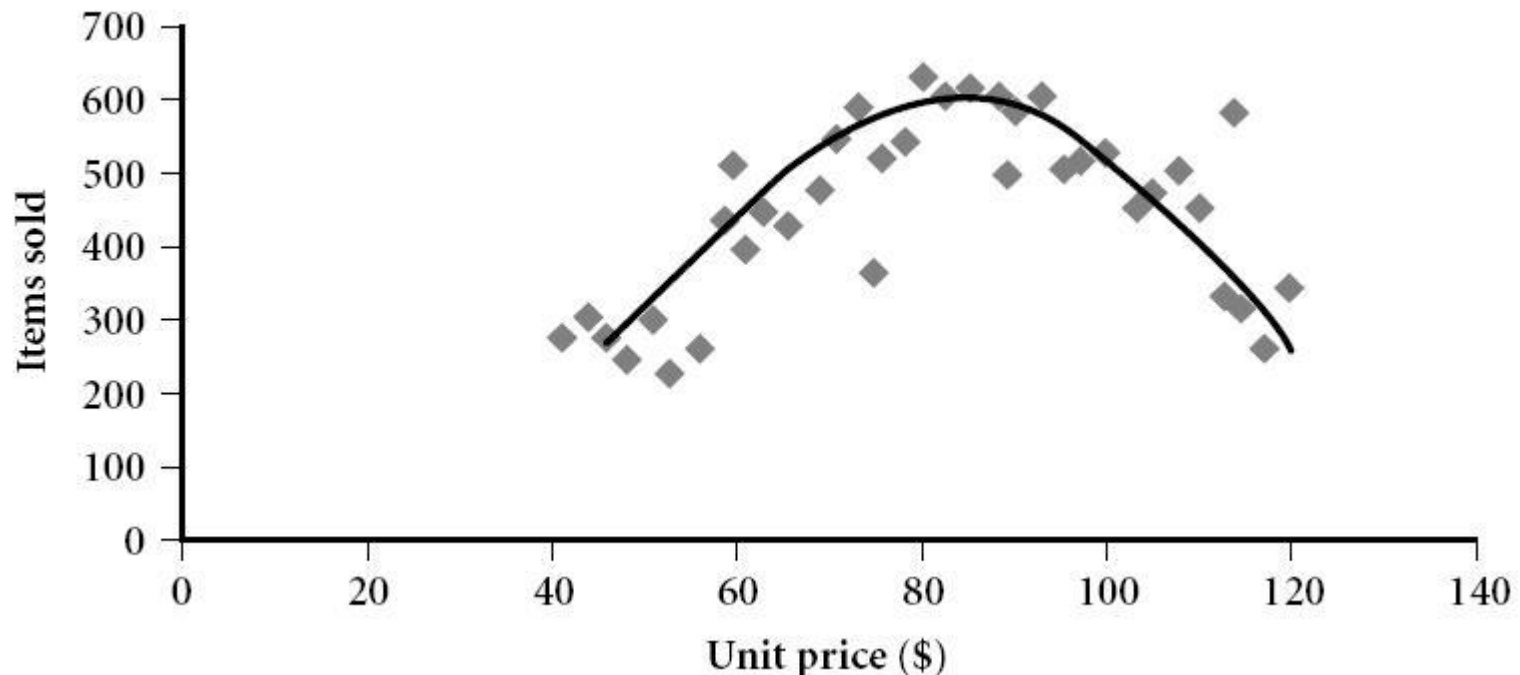
Scatter plot

- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane

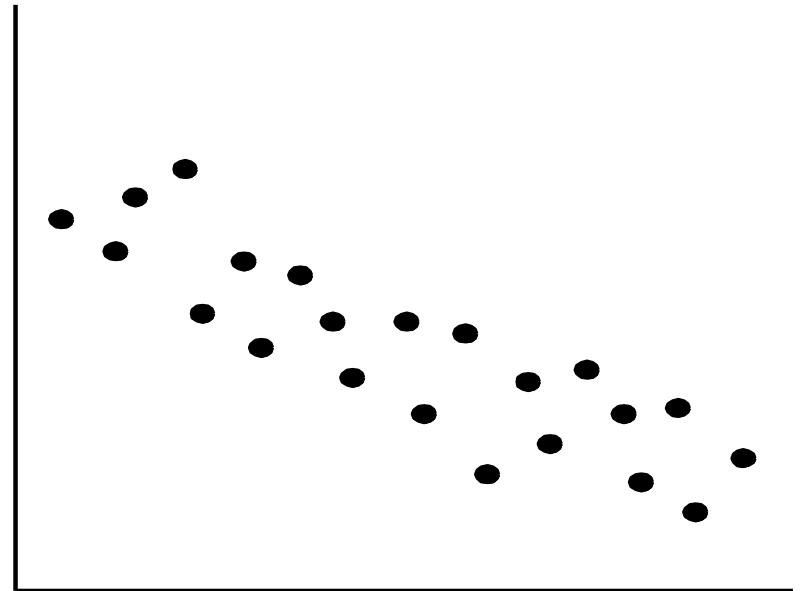
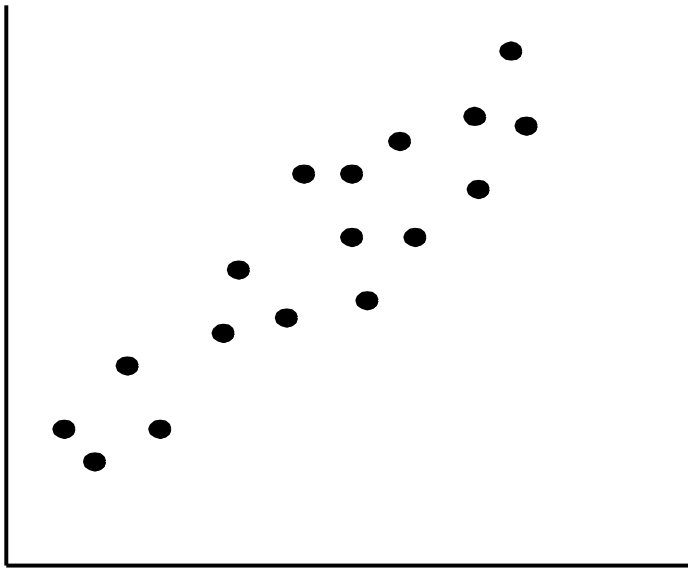


Loess Curve

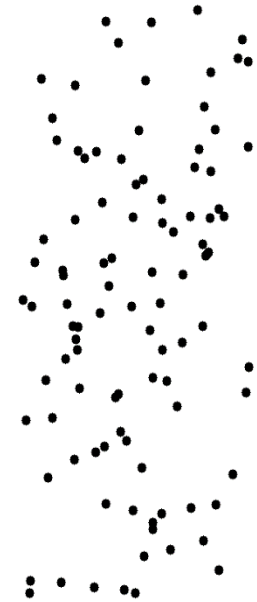
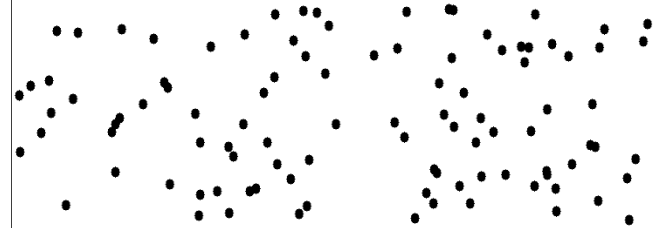
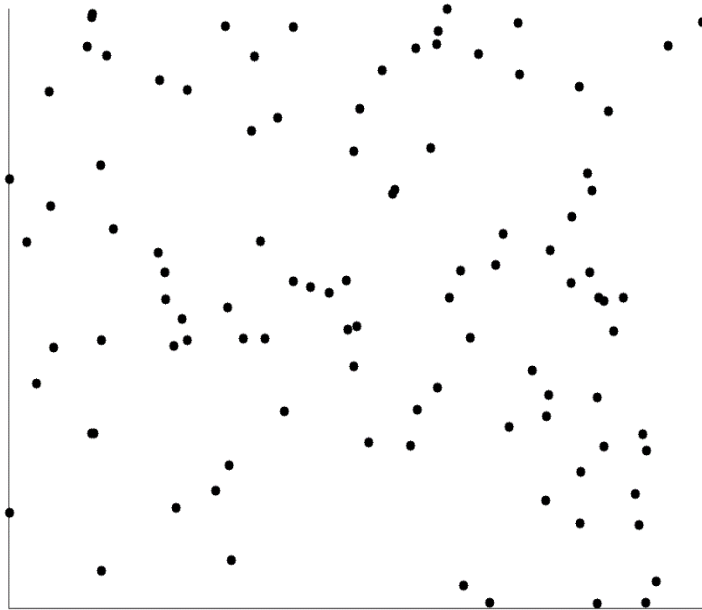
- Adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence
- Loess curve is fitted by setting two parameters: a smoothing parameter, and the degree of the polynomials that are fitted by the regression



Positively and Negatively Correlated Data



Not Correlated Data



Graphic Displays of Basic Statistical Descriptions

- Histogram
- Boxplot
- Quantile plot: each value x_i is paired with f_i indicating that approximately 100 f_i % of data are $\leq x_i$
- Quantile-quantile (q-q) plot: graphs the quantiles of one univariant distribution against the corresponding quantiles of another
- Scatter plot: each pair of values is a pair of coordinates and plotted as points in the plane
- Loess (local regression) curve: add a smooth curve to a scatter plot to provide better perception of the pattern of dependence

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Cleaning

- Importance
 - “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
 - Fill in missing values
 - Identify outliers and smooth out noisy data
 - Correct inconsistent data
 - Resolve redundancy caused by data integration

Missing Data

- 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
- Missing data may need to be inferred.

How to Handle Missing Data?

- Ignore the tuple: not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
 - a global constant : e.g., “unknown”, a new class?!
 - the attribute mean
 - the attribute mean for all samples belonging to the same class: smarter
 - the most probable value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may be due to
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention
- Other data problems which requires data cleaning
 - duplicate records
 - incomplete data
 - inconsistent data

How to Handle Noisy Data?

- Binning

- first sort data and partition into (equal-frequency) bins
- then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.

- Regression

- smooth by fitting the data into regression functions

- Clustering

- detect and remove outliers

- Combined computer and human inspection

- detect suspicious values and check by human (e.g., deal with possible outliers)

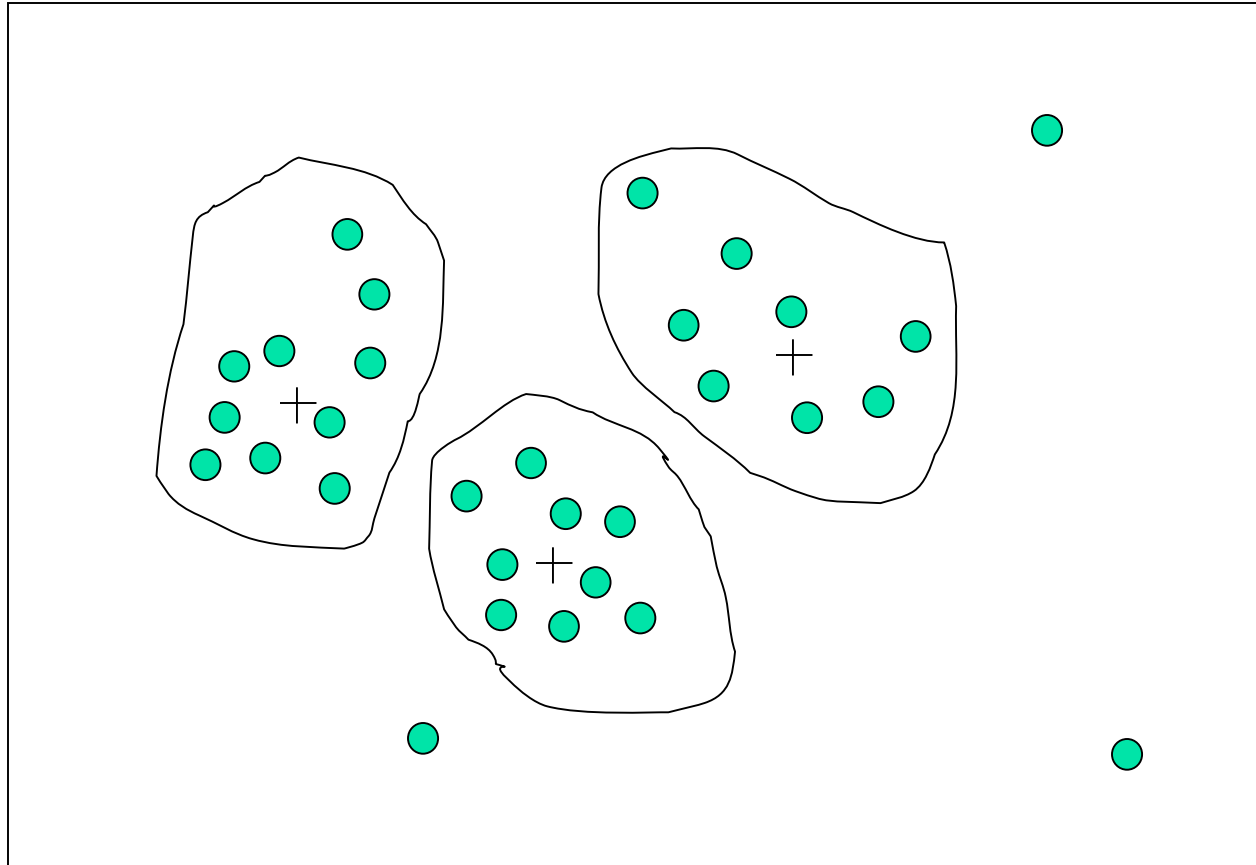
Simple Discretization Methods: Binning

- **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

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

Cluster Analysis



Data Cleaning as a Process

■ Data discrepancy detection

- Use metadata (e.g., domain, range, dependency, distribution)
- Check uniqueness rule, consecutive rule and null rule
- Use commercial tools
 - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
 - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)

■ Data migration and integration

- Data migration tools: allow transformations to be specified
- ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Integration

- **Data integration:**
 - Combines data from multiple sources into a coherent store
- Schema integration: e.g., $A.cust-id \equiv B.cust-\#$
 - Integrate metadata from different sources
- Entity identification problem:
 - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
 - For the same real world entity, attribute values from different sources are different
 - Possible reasons: different representations, different scales, e.g., metric vs. British units

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
 - *Object identification*: The same attribute or object may have different names in different databases
 - *Derivable data*: one attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Correlation Analysis (Numerical Data)

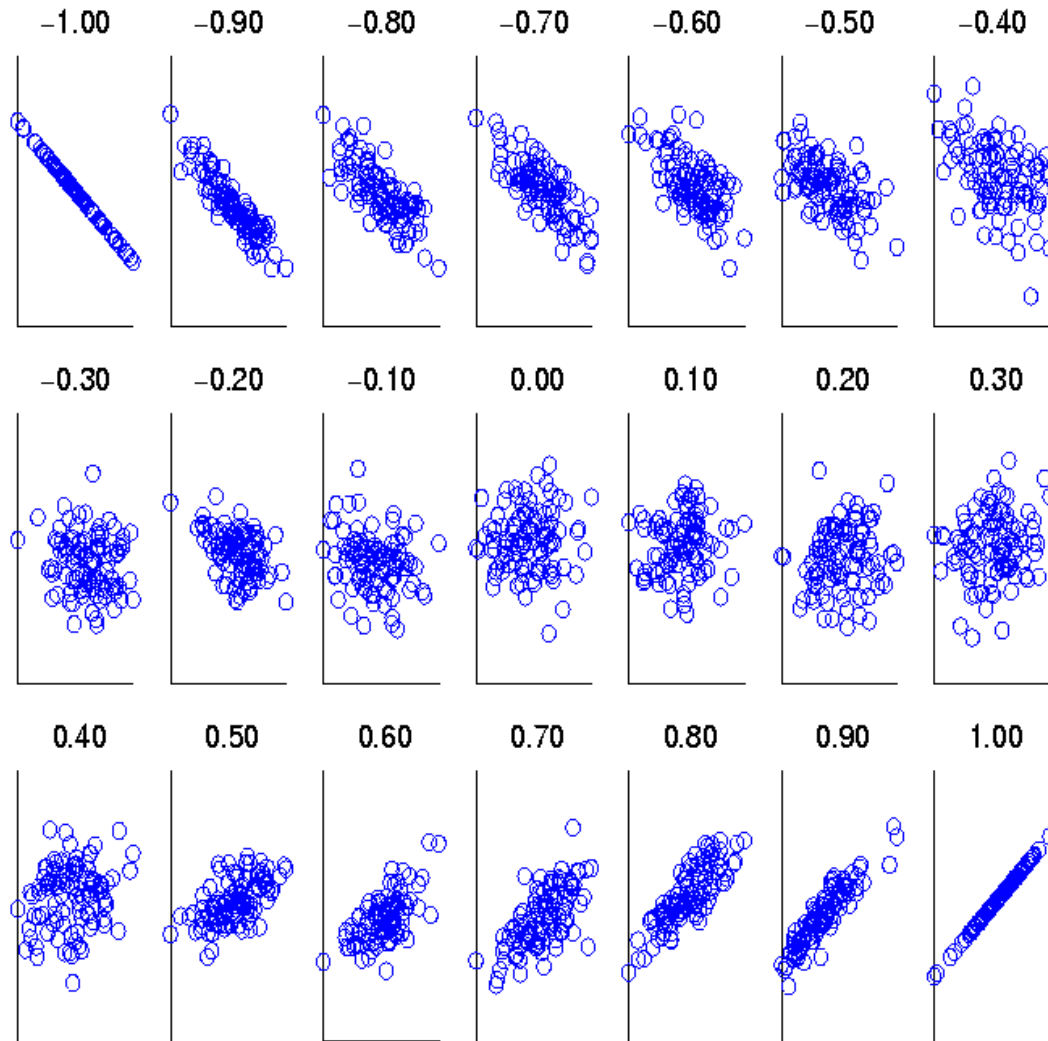
- Correlation coefficient (also called **Pearson's product moment coefficient**)

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

where n is the number of tuples, \bar{A} and \bar{B} are the respective means of A and B , σ_A and σ_B are the respective standard deviation of A and B , and $\sum(AB)$ is the sum of the AB cross-product.

- If $r_{A,B} > 0$, A and B are positively correlated (A 's values increase as B 's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{A,B} < 0$: negatively correlated

Visually Evaluating Correlation



**Scatter plots
showing the
similarity from
-1 to 1.**

Correlation Analysis (Categorical Data)

- χ^2 (chi-square) test

$$\chi^2 = \sum_{i=1}^c \sum_{j=1}^r \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \quad (3.1)$$

where o_{ij} is the *observed frequency* (i.e., actual count) of the joint event (A_i, B_j) and e_{ij} is the *expected frequency* of (A_i, B_j) , which can be computed as

$$e_{ij} = \frac{\text{count}(A = a_i) \times \text{count}(B = b_j)}{n}, \quad (3.2)$$

- The larger the χ^2 value, the more likely the variables are related
- The cells that contribute the most to the χ^2 value are those whose actual count is very different from the expected count

Chi-Square Calculation: An Example

	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

- χ^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$e_{11} = \frac{\text{count}(\text{male}) \times \text{count}(\text{fiction})}{n} = \frac{300 \times 450}{1500} = 90$$

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

- It shows that like_science_fiction and play_chess are correlated in the group

Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
 - min-max normalization
 - z-score normalization
 - normalization by decimal scaling
- Attribute/feature construction
 - New attributes constructed from the given ones

Data Transformation: Normalization

- Min-max normalization: to $[new_min_A, new_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$

- Ex. 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 (μ : mean, σ : standard deviation):

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

- Ex. Let $\mu = 54,000$, $\sigma = 16,000$. Then $\frac{73,600 - 54,000}{16,000} = 1.225$

- Normalization by decimal scaling

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that } \text{Max}(|v'|) < 1$$

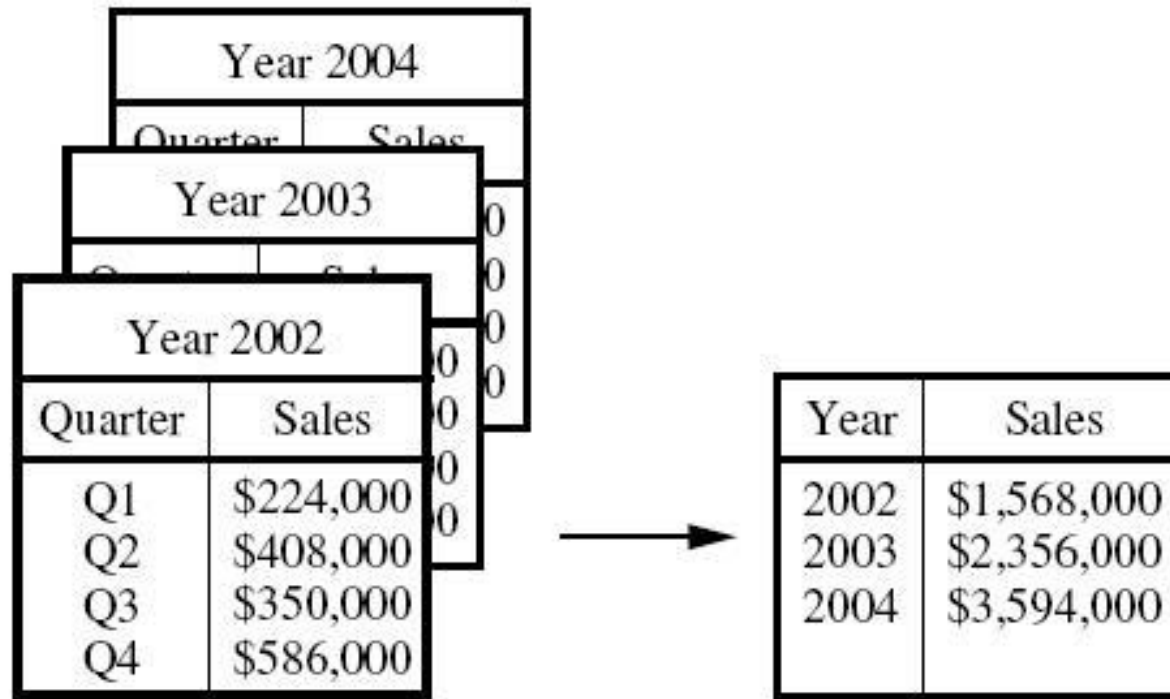
Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Reduction Strategies

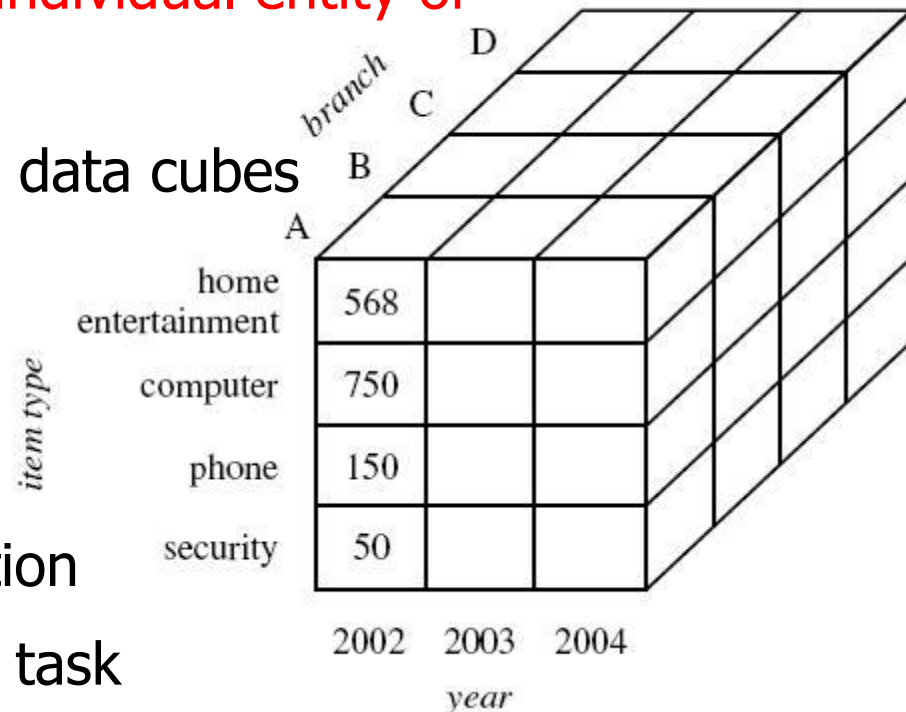
- Why data reduction?
 - A database/data warehouse may store terabytes of data
 - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
 - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
 - Data cube aggregation
 - Attribute subset selection — e.g., remove unimportant attributes
 - Dimensionality reduction — use encoding mechanisms
 - Numerosity reduction — e.g., fit data into models
 - Discretization and concept hierarchy generation

Aggregation Example



Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
 - The aggregated data for an **individual entity of interest**
- Multiple levels of aggregation in data cubes
 - Further reduce the size of data to deal with
- Reference appropriate levels
 - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible



Attribute Subset Selection

- Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
- reduce # of attributes in the patterns, easier to understand
- Redundant attributes
 - duplicate much or all of the information contained in one or more other attributes
 - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
 - contain no information that is useful for the data mining task at hand
 - E.g., students' ID is often irrelevant to the task of predicting students' GPA

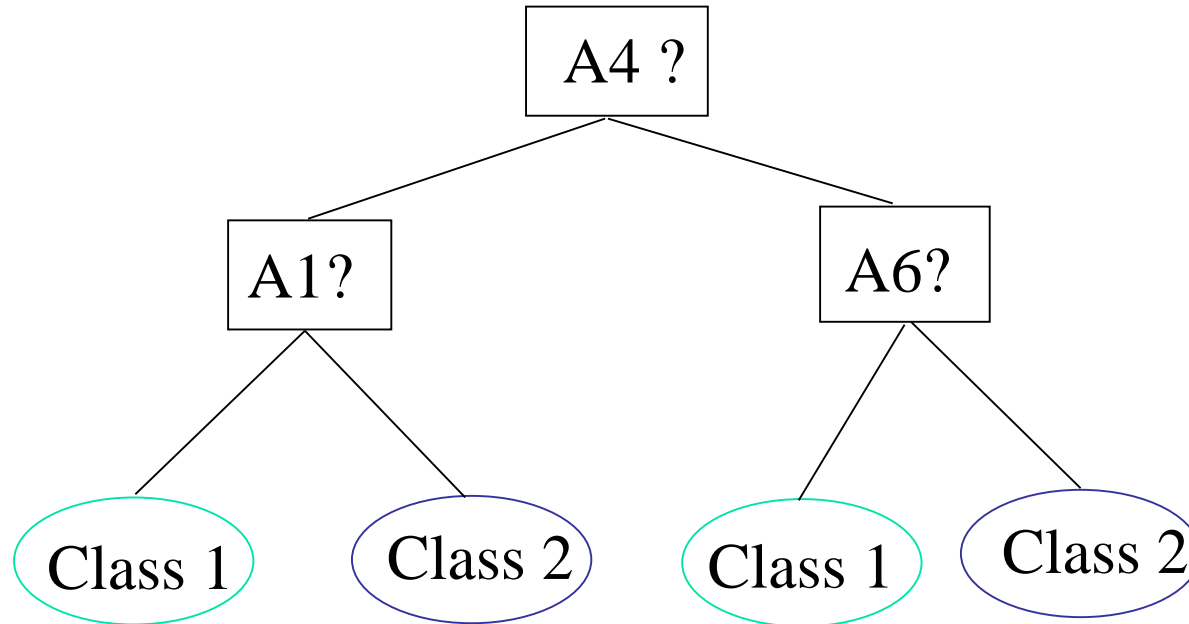
Heuristic Search in Attribute Selection

- There are 2^d possible attribute combinations of d attributes
- Typical heuristic attribute selection methods:
 - Best single attribute under the attribute independence assumption: choose by significance tests
 - Best step-wise feature selection:
 - The best single-attribute is picked first
 - Then next best attribute condition to the first, ...
 - Step-wise attribute elimination:
 - Repeatedly eliminate the worst attribute
 - Best combined attribute selection and elimination
 - Decision-tree induction

Example of Decision Tree Induction

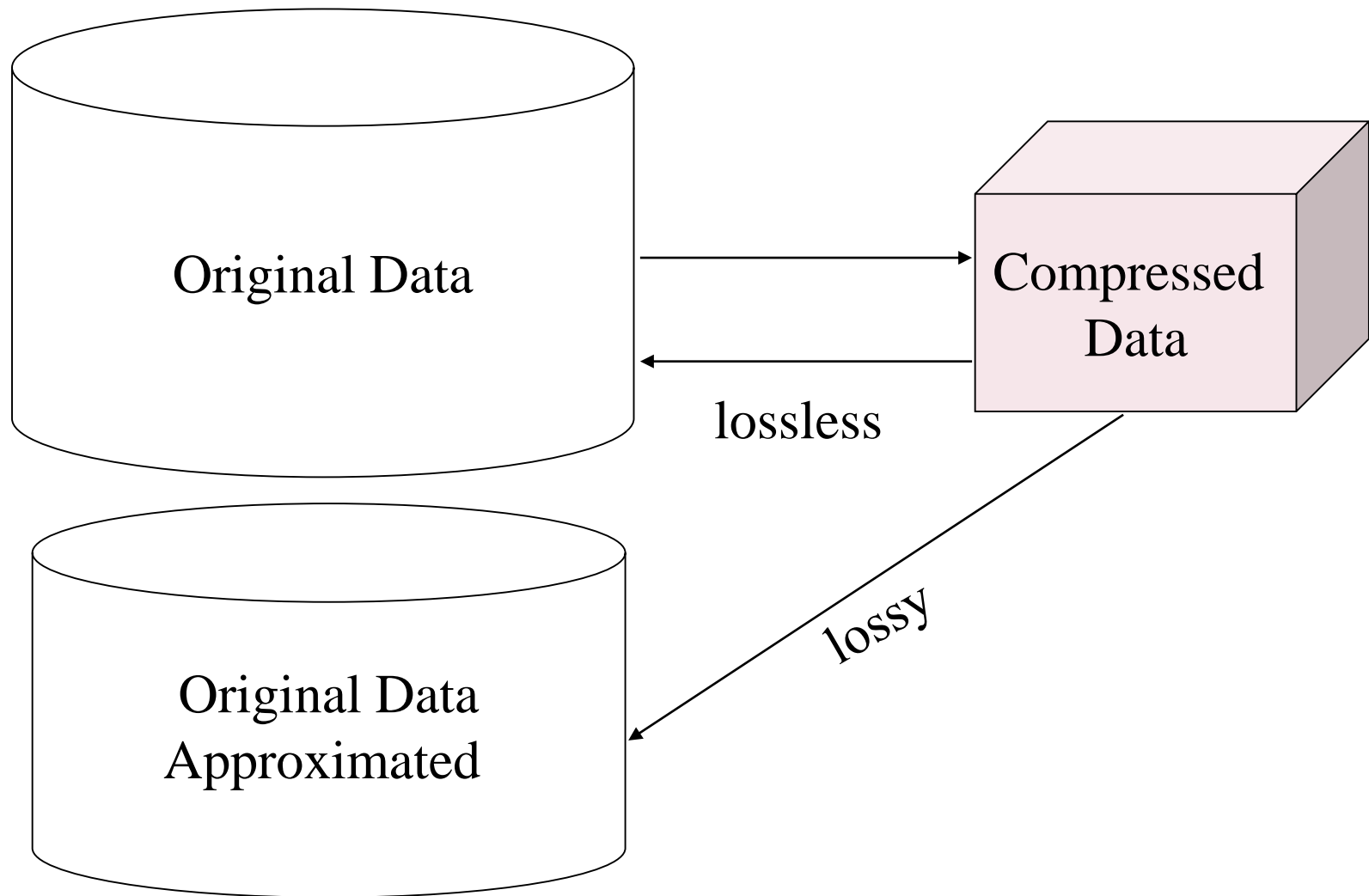
Initial attribute set:

{ A1, A2, A3, A4, A5, A6 }



-----> Reduced attribute set: { A1, A4, A6 }

Data Compression



Data Compression

- String compression
 - There are extensive theories and well-tuned algorithms
 - Typically lossless
- Audio/video compression
 - Typically lossy compression
 - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Two popular methods of lossy dimensionality reduction:
 - Wavelet transforms
 - Principle component analysis

Numerosity Reduction

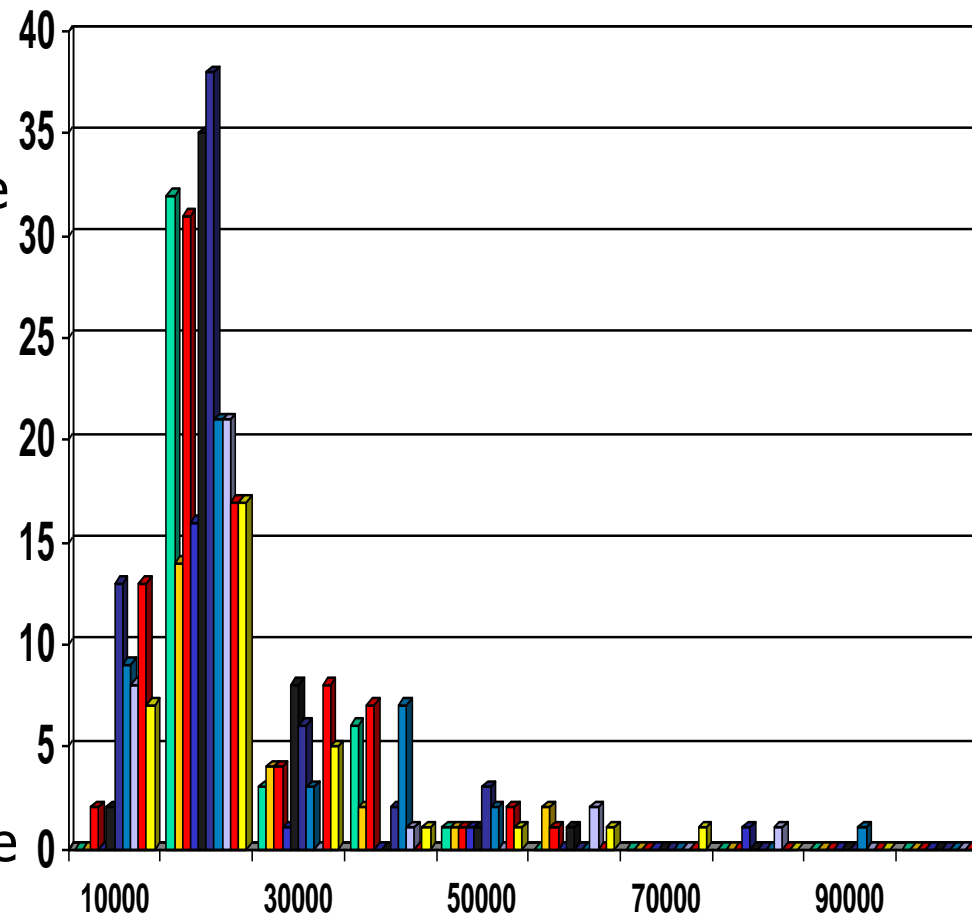
- Reduce data volume by choosing alternative, smaller forms of data representation
- Parametric methods
 - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
- Non-parametric methods
 - Do not assume models
 - Major families: histograms, clustering, sampling

Data Reduction Method (1): Regression

- Linear regression: Data are modeled to fit a straight line
 - $Y = wX + b$
 - Two regression coefficients, w and b , specify the line and are to be estimated by using the data at hand
 - Often uses the least-square method to fit the line
- Multiple regression: allows a response variable Y to be modeled as a linear function of multidimensional feature vector
 - $Y = b_0 + b_1 X_1 + b_2 X_2$

Data Reduction Method (2): Histograms

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
 - Equal-width: equal bucket range
 - Equal-frequency (or equal-depth)
 - V-optimal: with the least *histogram variance* (weighted sum of the original values that each bucket represents)
 - MaxDiff: set bucket boundary between each pair for pairs have the $\beta-1$ largest differences



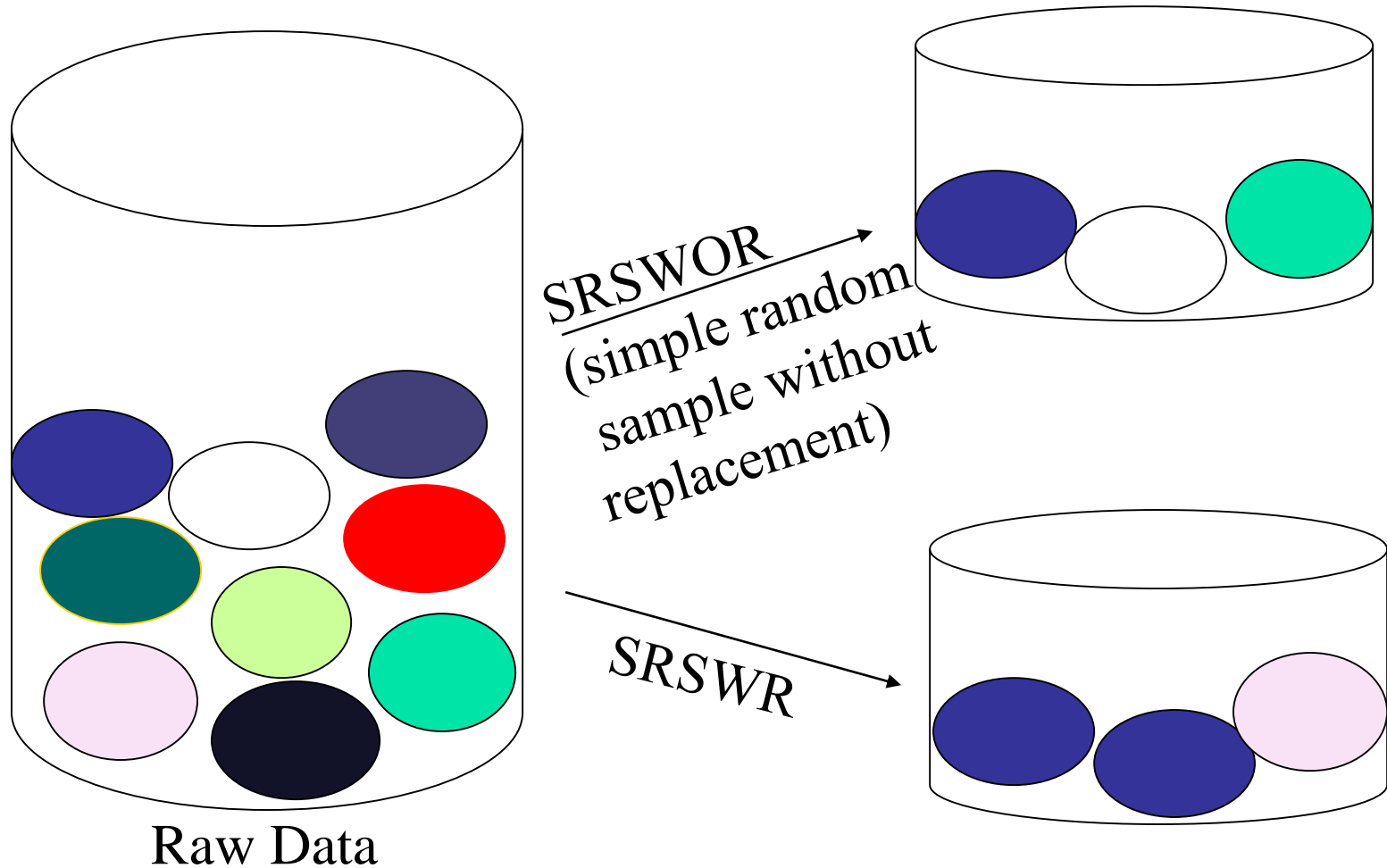
Data Reduction Method (3): Clustering

- Partition data set into clusters based on similarity, and store cluster representation only
- Can be very effective if data is clustered
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 7

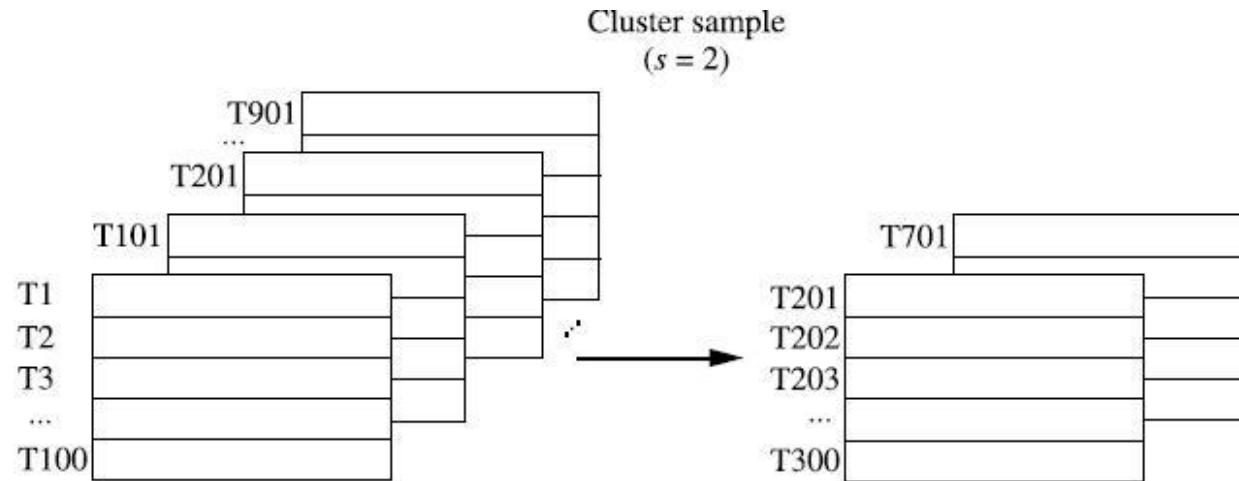
Data Reduction Method (4): Sampling

- Sampling: obtaining a small sample s to represent the whole data set N
- The cost of obtaining a sample is proportional to the size of the sample opposed to the size of data set.
- Choose a **representative** subset of the data
 - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
 - Stratified sampling:
 - Approximate the percentage of each class (or subpopulation of interest) in the overall database
 - Used in conjunction with skewed data

Sampling: with or without Replacement



Sampling: Cluster or Stratified Sampling



Stratified sample
(according to *age*)

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

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

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Discretization and Concept Hierarchy

- Three types of attributes
 - Nominal—values from an unordered set, e.g., color, profession
 - Ordinal—values from an ordered set, e.g., military or academic rank
 - Numeric—a measurable quantity, e.g., integer or real numbers
- Discretization: Divide the range of a continuous attribute into intervals
 - Interval labels can then be used to replace actual data values
 - Reduce data size by discretization
 - Supervised vs. unsupervised
 - Split (top-down) vs. merge (bottom-up)
 - Discretization can be performed recursively on an attribute
 - Prepare for further analysis, e.g., classification
- Concept hierarchy formation
 - Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as young, middle-aged, or senior)

Discretization and Concept Hierarchy Generation for Numeric Data

- Typical methods: All the methods can be applied recursively
 - Binning and Histogram Analysis
 - Top-down split, unsupervised
 - Clustering analysis
 - Either top-down split or bottom-up merge, unsupervised
 - Entropy-based discretization: supervised, top-down split
 - Interval merging by χ^2 Analysis: supervised, bottom-up merge

Entropy-Based Discretization

- Given a set of samples S , if S is partitioned into two intervals S_1 and S_2 using boundary T , the information gain after partitioning is

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

- Entropy is calculated based on class distribution of the samples in the set. Given m classes, the entropy of S_1 is

$$\text{Entropy}(S_1) = - \sum_{i=1}^m p_i \log_2(p_i)$$

where p_i is the probability of class i in S_1

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization
- The process is recursively applied to partitions obtained until some stopping criterion is met
- Such a boundary may reduce data size and improve classification accuracy

Interval Merge by χ^2 Analysis

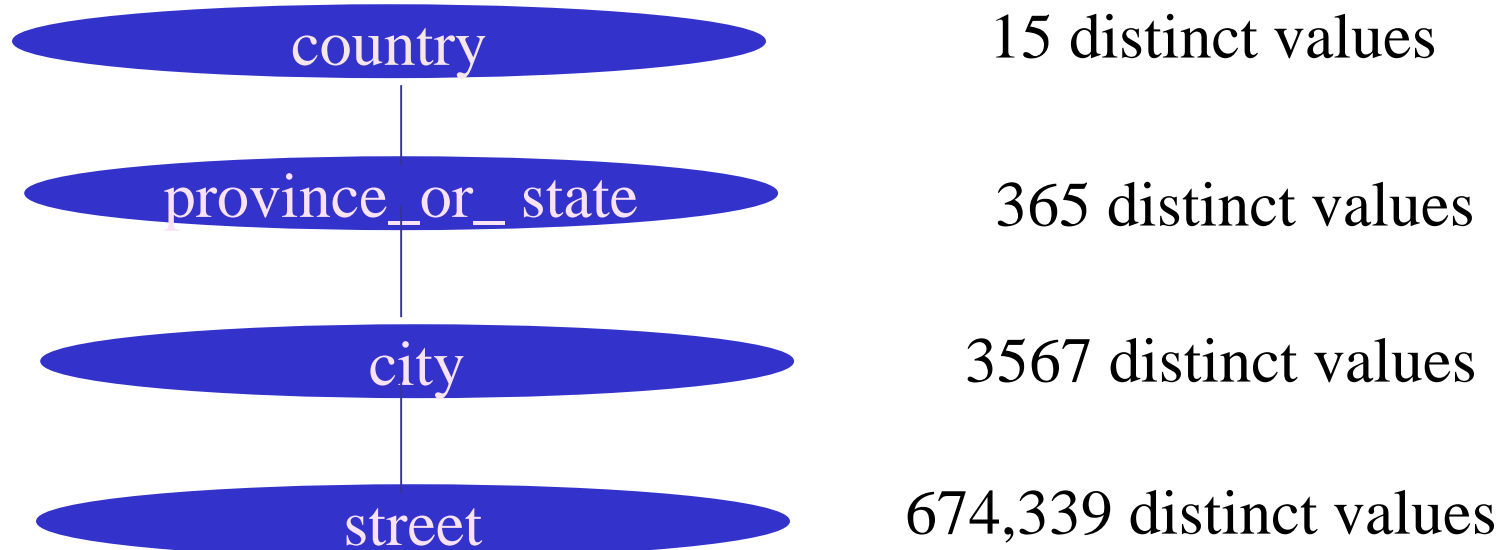
- Merging-based (bottom-up) vs. splitting-based methods
- Merge: Find the best neighboring intervals and merge them to form larger intervals recursively
- ChiMerge [Kerber AAAI 1992, See also Liu et al. DMKD 2002]
 - Initially, each distinct value of a numerical attr. A is considered to be one interval
 - χ^2 tests are performed for every pair of adjacent intervals
 - Adjacent intervals with the least χ^2 values are merged together
 - Hypothesis: the class is independent of which of the two adjacent intervals an instance belongs to
 - If the hypothesis is confirmed, there is no advantage in treating the intervals separately and they are merged.
 - This merge process proceeds recursively until a predefined stopping criterion is met

Concept Hierarchy Generation for Categorical Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
 - $\text{street} < \text{city} < \text{state} < \text{country}$
- Specification of a hierarchy for a set of values by explicit data grouping
 - $\{\text{Urbana, Champaign, Chicago}\} < \text{Illinois}$
- Automatic generation of hierarchies (or attribute levels)
 - E.g., for a set of attributes: $\{\text{street, city, state, country}\}$

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
 - The attribute with the most distinct values is placed at the lowest level of the hierarchy
 - Exceptions, e.g., weekday, month, quarter, year



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Summary

- Data preparation or preprocessing is a big issue for both data warehousing and data mining
- Descriptive data summarization is need for quality data preprocessing
- Data preparation includes
 - Data cleaning and data integration
 - Data reduction and feature selection
 - Discretization
- A lot of methods have been developed but data preprocessing still an active area of research

Data Mining:

Concepts and Techniques


(3rd ed.)

— Chapter 8 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

©2011 Han, Kamber & Pei. All rights reserved.

Chapter 6. Classification and Prediction

- What is classification? What is prediction? 
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

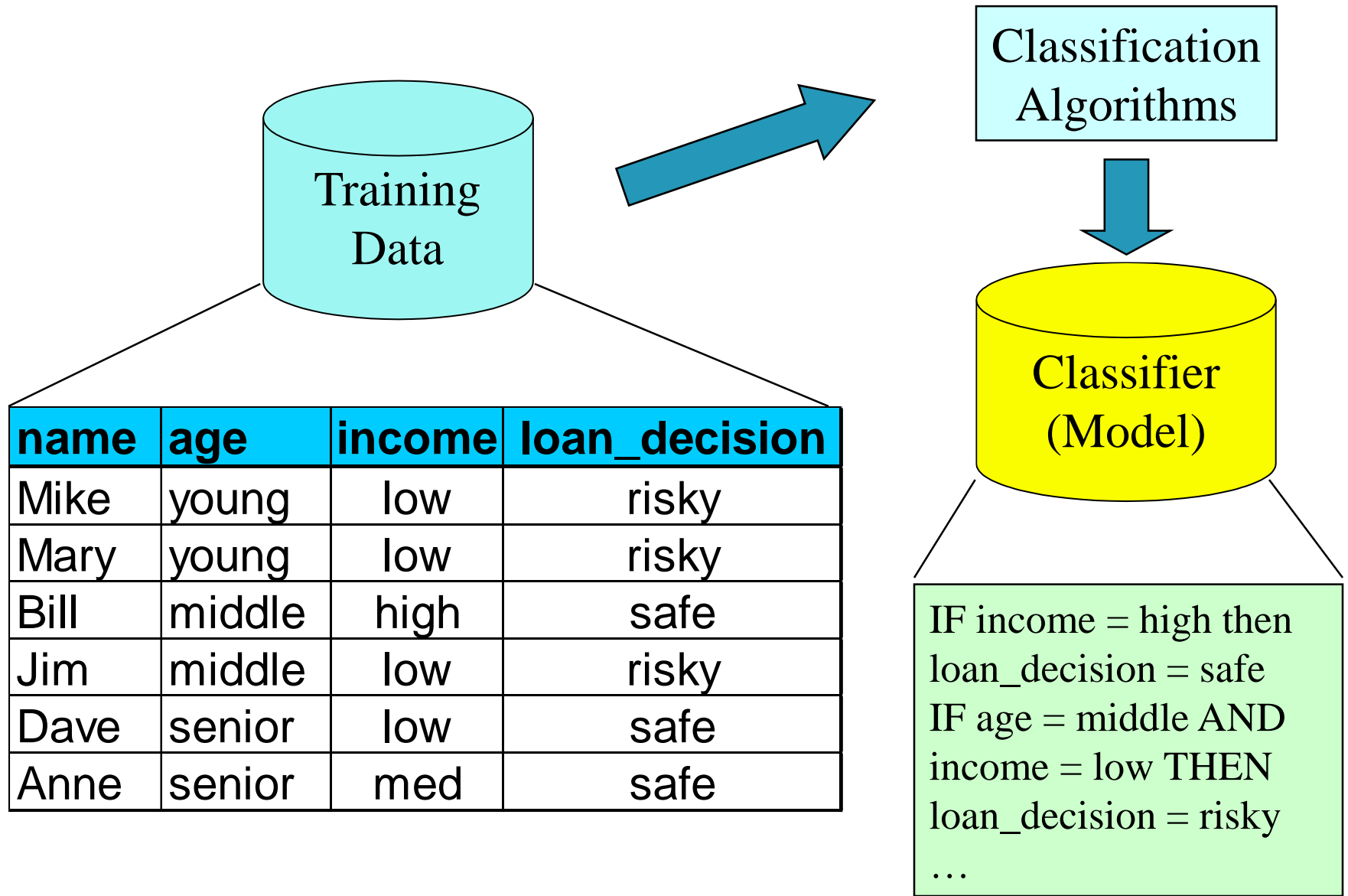
Classification vs. Prediction

- **Classification**
 - predicts categorical class labels (discrete or nominal)
 - classifies data (constructs a model) based on the training set and the values (**class labels**) in a classifying attribute and uses it in classifying new data
- **Prediction**
 - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
 - Target marketing
 - Medical diagnosis
 - Fraud detection

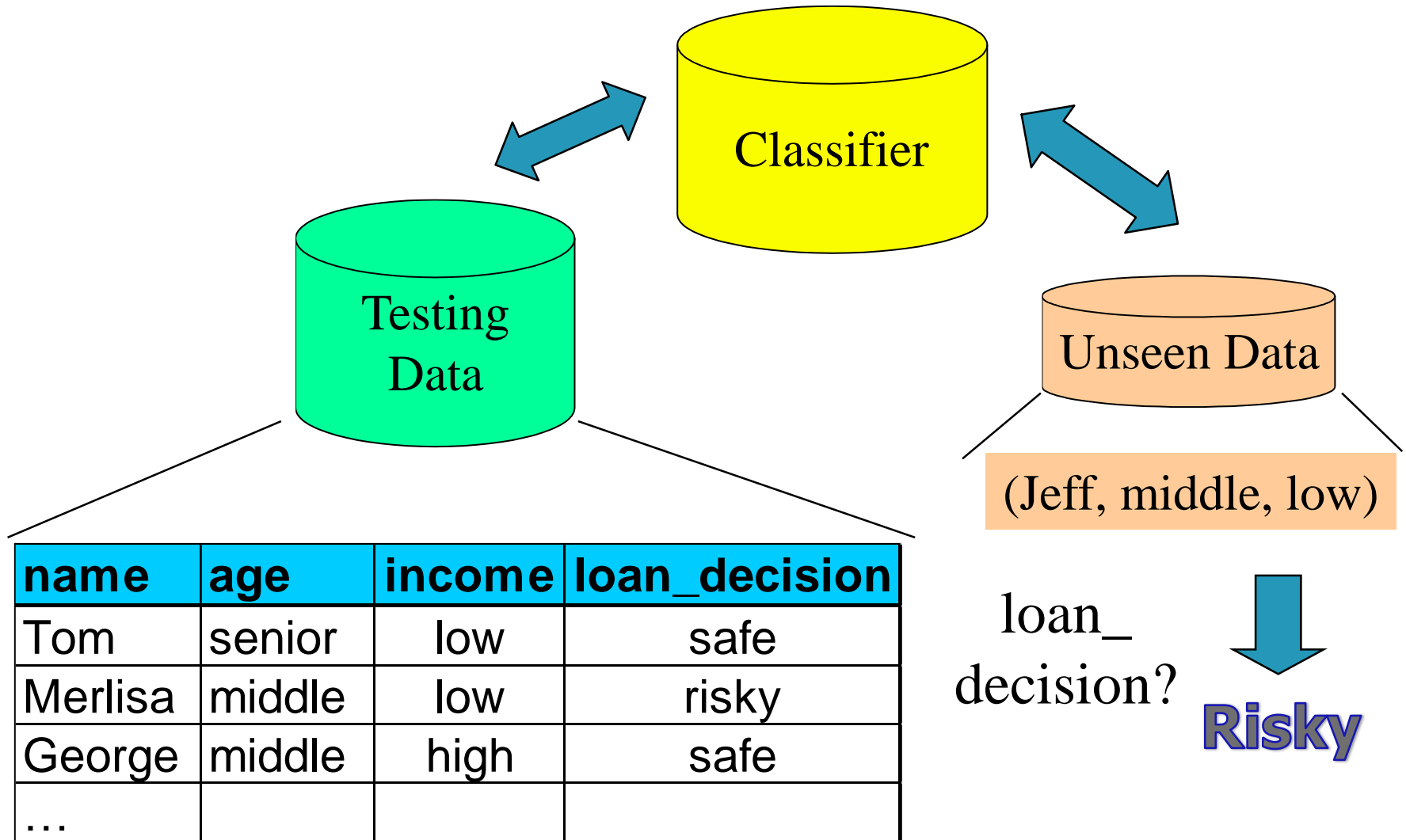
Classification—A Two-Step Process

- **Model construction**: describing a set of predetermined classes
 - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
 - The set of tuples used for model construction is **training set**
 - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage**: for classifying future or unknown objects
 - **Estimate accuracy** of the model
 - The known label of test sample is compared with the classified result from the model
 - Accuracy rate is the percentage of test set samples that are correctly classified by the model
 - Test set is independent of training set, otherwise over-fitting will occur
 - If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known

Process (1): Model Construction




Process (2): Using the Model in Prediction



Supervised vs. Unsupervised Learning

- Supervised learning (classification)
 - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
 - New data is classified based on the training set
- Unsupervised learning (clustering)
 - The class labels of training data is unknown
 - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction 
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary


Issues: Data Preparation

- Data cleaning
 - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
 - Remove the irrelevant or redundant attributes
- Data transformation
 - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Accuracy
 - classifier accuracy: predicting class label
 - predictor accuracy: guessing value of predicted attributes
- Speed
 - time to construct the model (training time)
 - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
 - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Chapter 6. Classification and Prediction

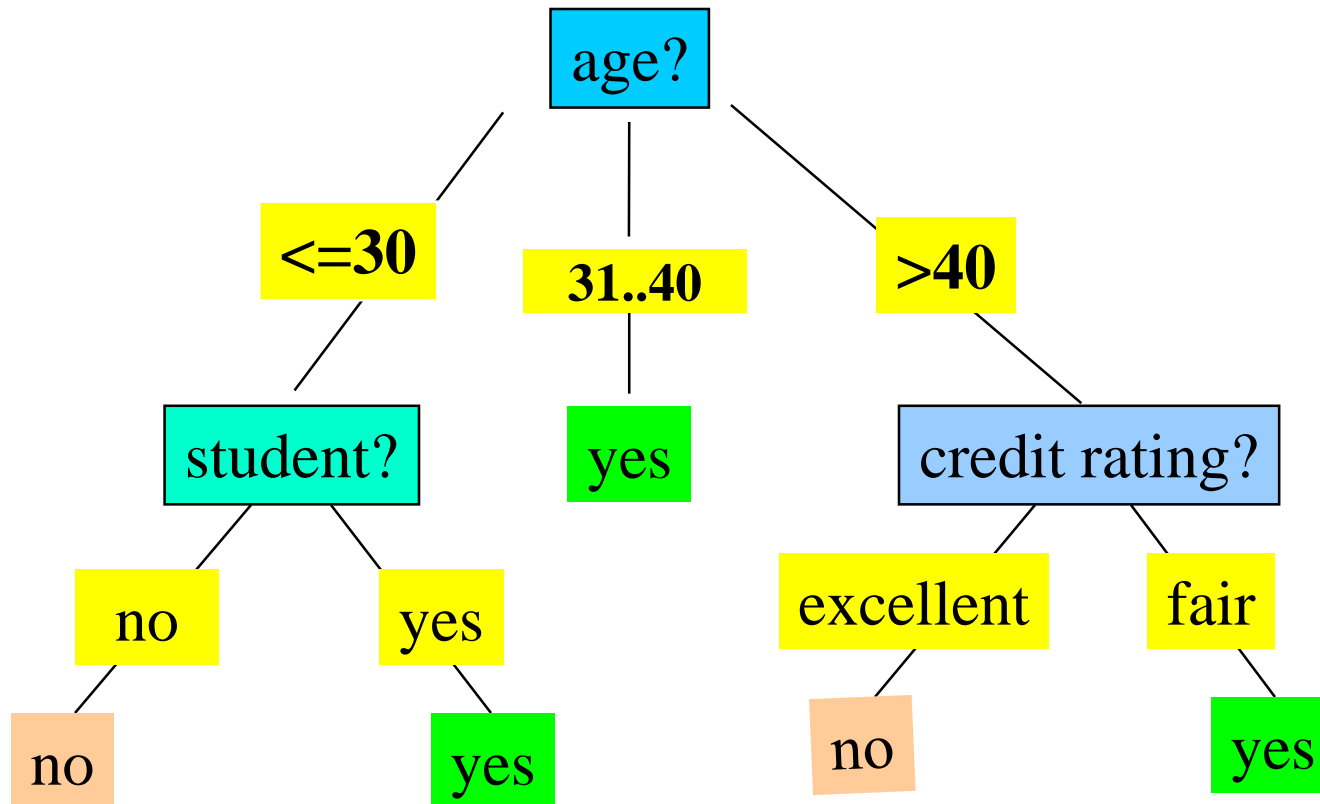
- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction 
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Decision Tree Induction: Training Dataset

This follows an example of Quinlan's ID3 (Playing Tennis)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
 - Tree is constructed in a **top-down recursive divide-and-conquer manner**
 - At start, all the training examples are at the root
 - Examples are partitioned recursively based on selected attributes
 - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)
- Conditions for stopping partitioning
 - All samples for a given node belong to the same class
 - There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
 - There are no samples left

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D :

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

- **Information** needed (after using A to split D into v partitions) to classify D :

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times I(D_j)$$

- **Information gained** by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

age	p _i	n _i	I(p _i , n _i)
<=30	2	3	0.971
31...40	4	0	0
>40	3	2	0.971

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

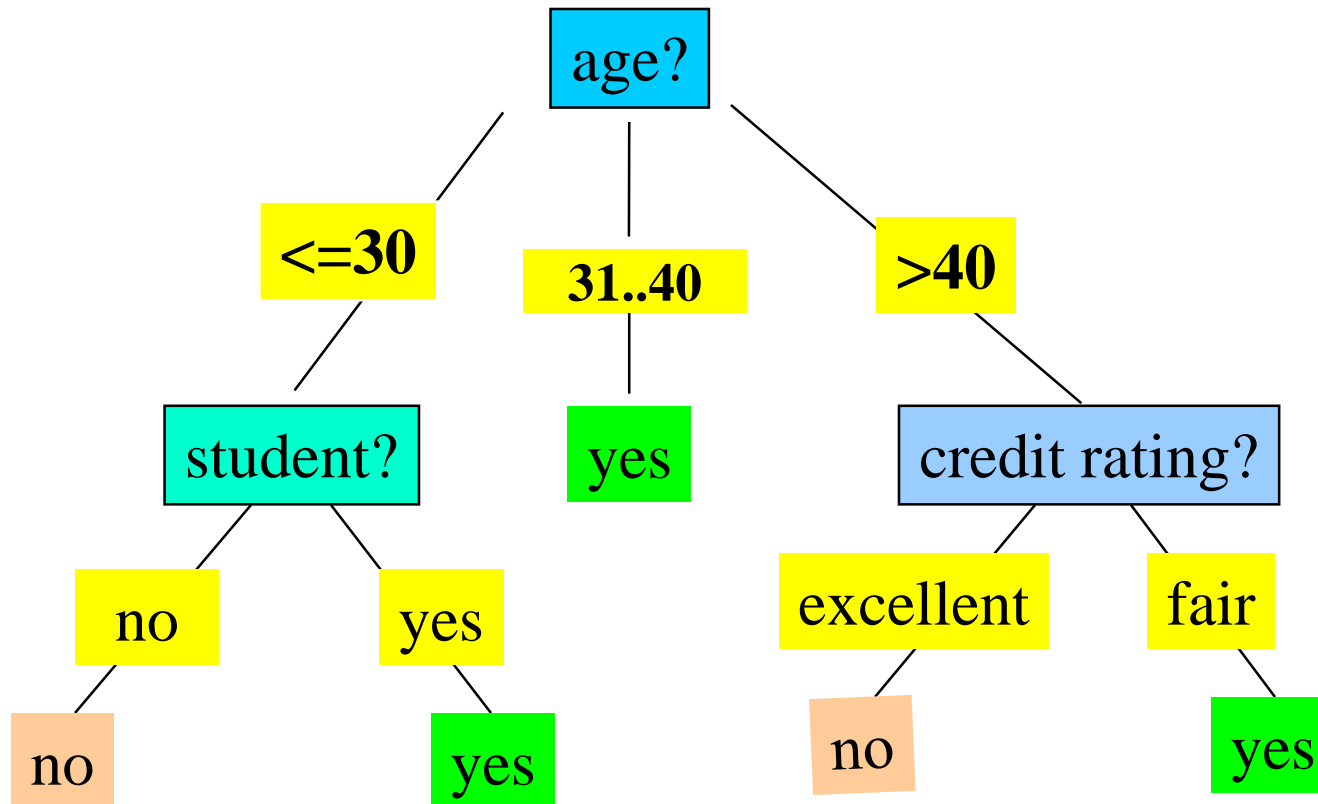
$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
 - Sort the value A in increasing order
 - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
 - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
 - D_1 is the set of tuples in D satisfying $A \leq \text{split-point}$, and D_2 is the set of tuples in D satisfying $A > \text{split-point}$

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $GainRatio(A) = Gain(A)/SplitInfo(A)$
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$
 - $gain_ratio(income) = 0.029/0.926 = 0.031$
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2 , the $gini$ index $gini_A(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Gini index (CART, IBM IntelligentMiner)

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1 : {low, medium} and 4 in D_2

$$\begin{aligned} gini_{income \in \{low, medium\}}(D) &= \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2) \\ &= \frac{10}{14} \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14} \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

$Gini_{\{low, high\}}$ is 0.458; $Gini_{\{medium, high\}}$ is 0.450. Therefore, split on the {low, medium} (and {high}) since it has the lowest Gini index

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
 - Information gain:
 - biased towards multivalued attributes
 - Gain ratio:
 - tends to prefer unbalanced splits in which one partition is much smaller than the others
 - Gini index:
 - biased to multivalued attributes
 - tends to favor tests that result in equal-sized partitions and purity in both partitions


Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
 - Too many branches, some may reflect anomalies due to noise or outliers
 - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
 - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
 - Difficult to choose an appropriate threshold
 - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
 - Use a set of data different from the training data to decide which is the “best pruned tree”

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
 - relatively faster learning speed (than other classification methods)
 - convertible to simple and easy to understand classification rules
 - comparable classification accuracy with other methods

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification 
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction*, *i.e.*, predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayesian Theorem: Basics

- Let \mathbf{X} be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine $P(H|\mathbf{X})$, the probability that the hypothesis holds given the observed data sample \mathbf{X}
- $P(H)$ (*prior probability*), the initial probability
 - E.g., \mathbf{X} will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$: probability that sample data is observed
- $P(\mathbf{X}|H)$ the probability of observing the sample \mathbf{X} , given that the hypothesis holds
 - E.g., Given that \mathbf{X} will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given training data \mathbf{X} , *posteriori probability of a hypothesis* H , $P(H|\mathbf{X})$, follows the Bayes theorem

$$P(H | \mathbf{X}) = \frac{P(\mathbf{X} | H)P(H)}{P(\mathbf{X})}$$

- Predicts \mathbf{X} belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the k classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n -D attribute vector $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are m classes C_1, C_2, \dots, C_m .
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since $P(\mathbf{X})$ is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X} | C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- Here x_k refers to the value of attribute A_k for tuple X .
- If A_k is categorical, $P(x_k | C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)

Derivation of Naïve Bayes Classifier

- If A_k is continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

And

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(x_k | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

We need to compute μ_{C_i} and σ_{C_i} , which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute A_k for training tuples of class C_i . We then plug these two above quantities, together with x_k , to estimate $P(x_k | C_i)$

Naïve Bayesian Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data sample

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

- $P(C_i)$:
 $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute $P(X|C_i)$ for each class
 $P(\text{age} = \text{"<=30"} \mid \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 $P(\text{age} = \text{"<= 30"} \mid \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 $P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 $P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 $P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 $P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 $P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 $P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
- **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$**

$$P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) * P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) = 0.007$$

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
 - Adding 1 to each case
 - Prob(income = low) = 1/1003
 - Prob(income = medium) = 991/1003
 - Prob(income = high) = 11/1003
 - The “corrected” prob. estimates are close to their “uncorrected” counterparts

Naïve Bayesian Classifier: Comments

- Advantages

- Easy to implement
- Good results obtained in most of the cases


- Disadvantages

- Assumption: class conditional independence, therefore loss of accuracy
- Practically, dependencies exist among variables
 - E.g., hospitals: patients: Profile: age, family history, etc.
Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
 - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

- How to deal with these dependencies?

- Bayesian Belief Networks
- Ref. D. Heckerman: Bayesian networks for data mining

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification 
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules

R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

- Rule antecedent/precondition vs. rule consequent

- Assessment of a rule: *coverage* and *accuracy*

- n_{covers} = # of tuples covered by R

- n_{correct} = # of tuples correctly classified by R

$\text{coverage}(R) = n_{\text{covers}} / |D|$ /* D: training data set */

$\text{accuracy}(R) = n_{\text{correct}} / n_{\text{covers}}$

- If more than one rule is triggered, need **conflict resolution**

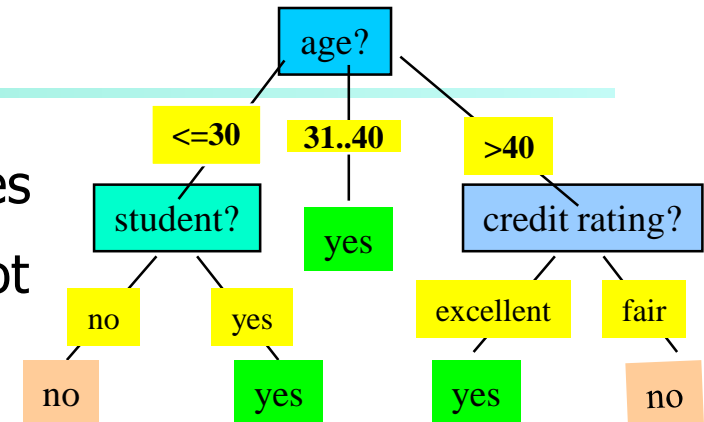
- Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the *most attribute test*)

- Class-based ordering: decreasing order of *prevalence*

- Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive
- Example: Rule extraction from our *buys_computer* decision-tree



IF *age* = young AND *student* = *no*

THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes*

THEN *buys_computer* = *yes*

IF *age* = mid-age

THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *no*

Rule Extraction from the Training Data

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
 - Rules are learned one at a time
 - Each time a rule is learned, the tuples covered by the rules are removed
 - The process repeats on the remaining tuples unless *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

How to Learn-One-Rule?

- Start with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
 - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
 - Foil-gain (in FOIL & RIPPER): assesses info gained by extending condition

$$FOIL_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

It favors rules that have high accuracy and cover many positive tuples


- Rule pruning based on an independent set of test tuples

$$FOIL_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors) 
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Lazy vs. Eager Learning

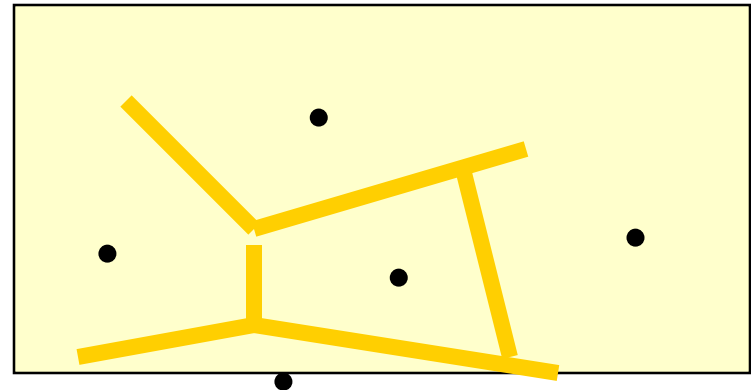
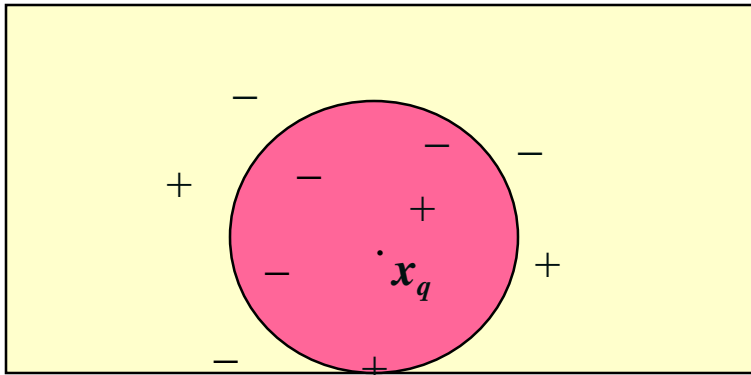
- Lazy vs. eager learning
 - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
 - Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting

Lazy Learner: Instance-Based Methods

- Instance-based learning:
 - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
 - k -nearest neighbor approach
 - Instances represented as points in a Euclidean space.
 - Case-based reasoning
 - Uses symbolic representations and knowledge-based inference

The k -Nearest Neighbor Algorithm


- All instances correspond to points in the n -D space
- The nearest neighbors are defined in terms of Euclidean distance, $\text{dist}(\mathbf{X}_1, \mathbf{X}_2)$
- Target function could be discrete- or real- valued
- For discrete-valued, k -NN returns the most common value among the k training examples nearest to x_q
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples



Discussion on the *k*-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
 - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
 - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
 - Give greater weight to closer neighbors $w \equiv \frac{1}{d(x_q, x_i)^2}$
- Robust to noisy data by averaging *k*-nearest neighbors
- Distance between neighbors could be dominated by irrelevant attributes
 - To overcome it, attribute weighting or elimination of the least relevant attributes


Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods 
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Other Classification Methods

- Support Vector Machines (SVM)
- Genetic Algorithms (GA)
- Rough Set Approach
- Fuzzy Set Approaches

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures 
- Ensemble methods
- Model selection
- Summary

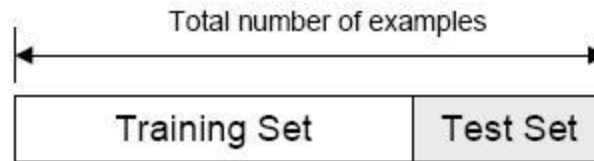
Model Evaluation

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
 - Holdout method, random subsampling
 - Cross-validation
 - Bootstrap

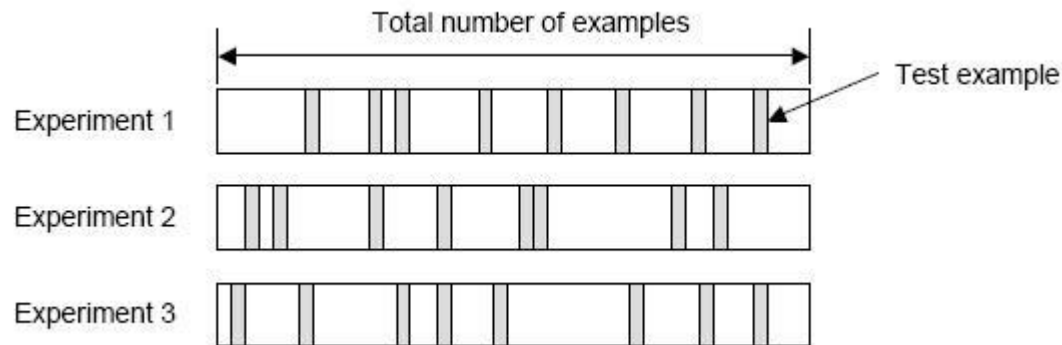
Evaluating the Accuracy of a Classifier or Predictor (I)

- Holdout method

- Given data is randomly partitioned into two independent sets
 - Training set (e.g., 2/3) for model construction
 - Test set (e.g., 1/3) for accuracy estimation

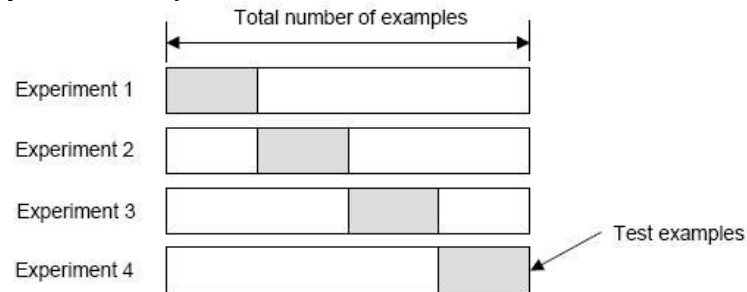


- Random sampling: a variation of holdout
 - Repeat holdout k times, accuracy = avg. of the accuracies obtained

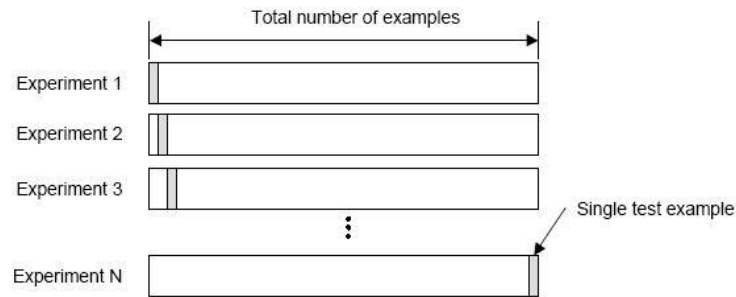


Evaluating the Accuracy of a Classifier or Predictor (II)

- Cross-validation (k -fold, where $k = 10$ is most popular)
 - Randomly partition the data into k *mutually exclusive* subsets, each approximately equal size
 - At i -th iteration, use D_i as test set and others as training set



- Leave-one-out: k folds where $k = \#$ of tuples, for small sized data



Classifier Evaluation Metrics: Accuracy & Error Rate

Confusion Matrix:

Actual class\Predicted class	C_1	$\sim C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\sim C_1$	False Positives (FP)	True Negatives (TN)

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified,

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Error rate: $1 - accuracy$, or

$$error\ rate = \frac{FP + FN}{TP + TN + FP + FN}$$

Classifier Evaluation Metrics:

Example - Confusion Matrix

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total	Recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
Total	7366	2634	10000	95.42

- Given m classes, an entry, **$CM_{i,j}$** in a **confusion matrix** indicates # of tuples in class i that were labeled by the classifier as class j .
- May be extra rows/columns to provide totals or recognition rate per class.

Classifier Evaluation Metrics: Sensitivity and Specificity

- **Class Imbalance Problem:**

- one class may be *rare*, e.g. fraud detection data, medical data
- significant *majority of the negative class* and minority of the positive class

- **Sensitivity:** True Positive recognition rate,

$$\text{sensitivity} = \frac{TP}{P}$$

Specificity: True Negative recognition rate,

$$\text{specificity} = \frac{TN}{N}$$

Classifier Evaluation Metrics:

Precision and Recall

- **Precision:** exactness – what % of tuples that the classifier labeled as positive are actually positive?

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$


- Perfect score is 1.0
- Inverse relationship between precision & recall

Classifier Evaluation Metrics: Example

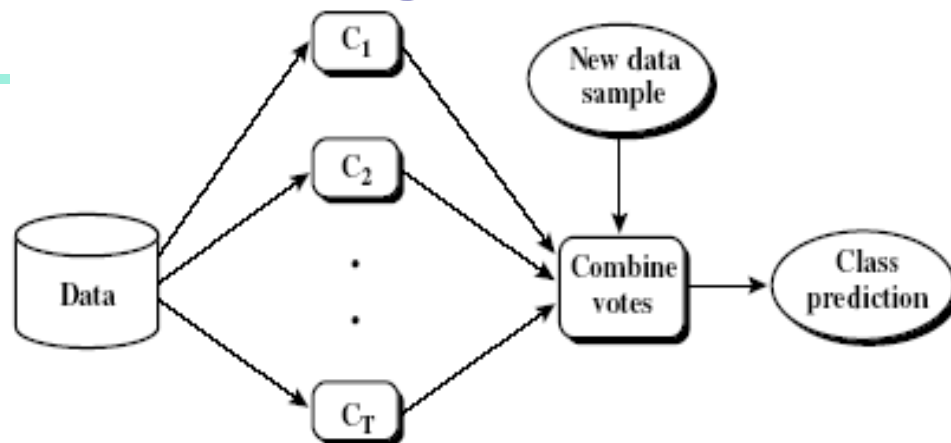
Actual class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	90	210	300	30.00 <i>sensitivity</i>
cancer = no	140	9560	9700	98.56 <i>specificity</i>
Total	230	9770	10000	96.40 <i>accuracy</i>

Precision = $90/230 = 39.13\%$; *Recall* = $90/300 = 30.00\%$

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods 
- Model selection
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
 - Use a combination of models to increase accuracy
 - Combine a series of k learned models, M_1, M_2, \dots, M_k , with the aim of creating an improved model M^*
- Popular ensemble methods
 - Bagging: averaging the prediction over a collection of classifiers
 - Boosting: weighted vote with a collection of classifiers

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
 - Given a set D of d tuples, at each iteration i , a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)
 - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample \mathbf{X}
 - Each classifier M_i returns its class prediction
 - The bagged classifier M^* counts the votes and assigns the class with the most votes to \mathbf{X}
- Accuracy
 - Often significantly better than a single classifier derived from D
 - For noise data: not considerably worse, more robust
 - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
 - Weights are assigned to each training tuple
 - A series of k classifiers is iteratively learned
 - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1} , to pay more attention to the training tuples that were misclassified by M_i
 - The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data


Adaboost (Freund and Schapire, 1997)

- Given a set of d class-labeled tuples, $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the weights of tuples are set the same ($1/d$)
- Generate k classifiers in k rounds. At round i ,
 - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
 - Each tuple's chance of being selected is based on its weight
 - A classification model M_i is derived from D_i
 - Its error rate is calculated using D_i as a test set
 - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: $\text{err}(\mathbf{X}_j)$ is the misclassification error of tuple \mathbf{X}_j . Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$\text{error}(M_i) = \sum_j^d w_j \times \text{err}(\mathbf{X}_j)$$

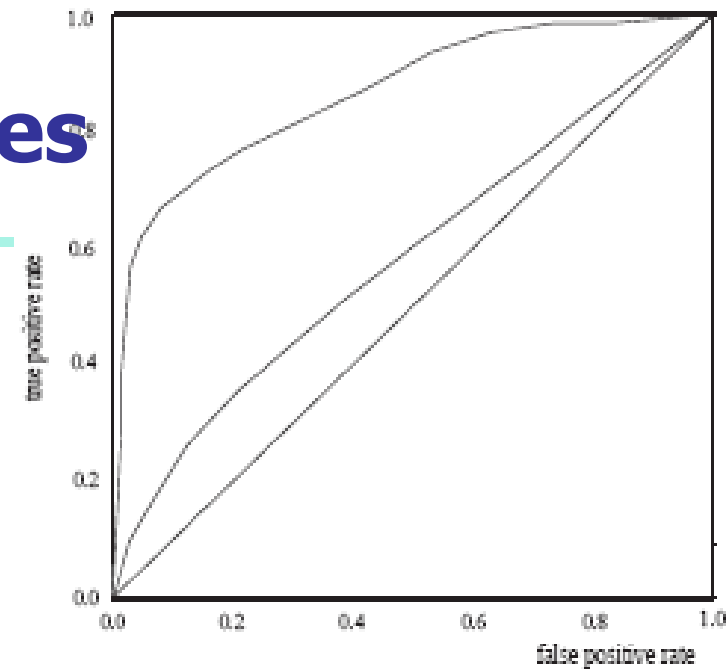
- The weight of classifier M_i 's vote is $\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection 
- Summary


Model Selection: ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary 

Summary (I)

- **Classification** and **prediction** are two forms of data analysis that can be used to extract **models** describing important data classes or to predict future data trends.
- Effective and scalable methods have been developed for **decision trees induction**, **Naive Bayesian classification**, **Bayesian belief network**, **rule-based classifier**, **Backpropagation**, **Support Vector Machine (SVM)**, **associative classification**, **nearest neighbor classifiers**, and **case-based reasoning**, and other classification methods such as **genetic algorithms**, **rough set** and **fuzzy set** approaches.
- **Linear**, **nonlinear**, and **generalized linear models of regression** can be used for **prediction**. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables. **Regression trees** and **model trees** are also used for prediction.

Summary (II)

- **Stratified k-fold cross-validation** is a recommended method for accuracy estimation. **Bagging** and **boosting** can be used to increase overall accuracy by learning and combining a series of individual models.
- **Significance tests** and **ROC curves** are useful for model selection
- There have been numerous **comparisons of the different classification and prediction methods**, and the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, interpretability, and scalability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

Data Mining:

Concepts and Techniques

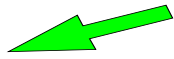
(3rd ed.)

— Chapter 6 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

©2011 Han, Kamber & Pei. All rights reserved.

Chapter 5: Mining Frequent Patterns, Association and Correlations

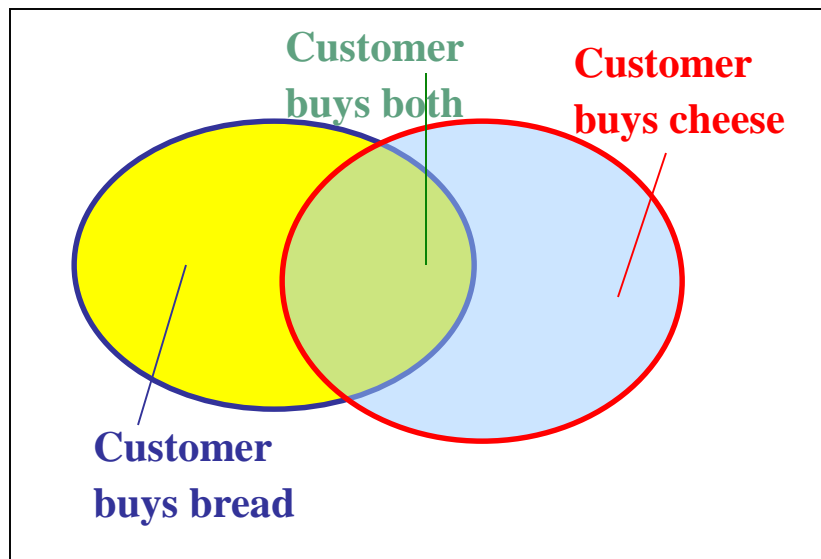
- Basic concepts and a road map 
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?— Bread and milk?!
 - What are the subsequent purchases after buying a PC?
- Applications
 - Basket data analysis, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns

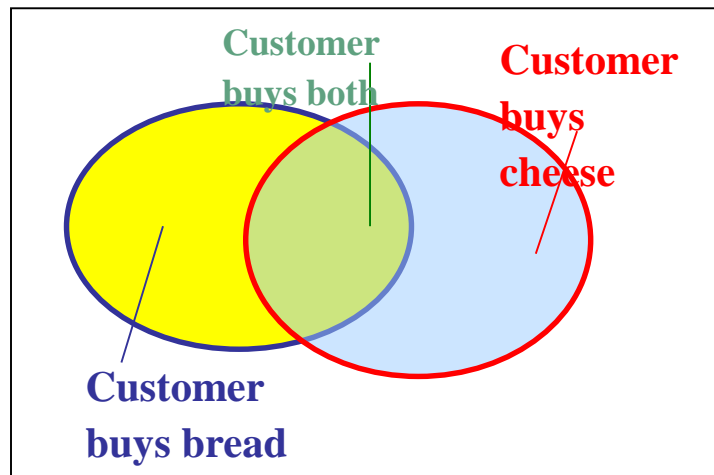
Tid	Items bought
10	Bread, Nuts, Cheese
20	Bread, Coffee, Cheese
30	Bread, Cheese, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Cheese, Eggs, Milk



- **itemset**: A set of one or more items
- **k-itemset** $X = \{x_1, \dots, x_k\}$
- **(absolute) support**, or, **support count** of X : Frequency or occurrence of an itemset X
- **(relative) support**, s , is the fraction of transactions that contains X (i.e., the **probability** that a transaction contains X)
- An itemset X is **frequent** if X 's support is no less than a *minsup* threshold

Basic Concepts: Association Rules

Tid	Items bought
10	Bread, Nuts, Cheese
20	Bread, Coffee, Cheese
30	Bread, Cheese, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Cheese, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - support**, s , **probability** that a transaction contains $X \cup Y$
 - confidence**, c , **conditional probability** that a transaction having X also contains Y

Let $minsup = 50\%$, $minconf = 50\%$

Freq. Pat.: Bread:3, Nuts:3, Cheese:4, Eggs:3, {Bread, Cheese}:3

- Association rules:
 - $Bread \rightarrow Cheese$ (60%, 100%)
 - $Cheese \rightarrow Bread$ (60%, 75%)


Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \dots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!
- Solution: Mine *closed patterns* and *max-patterns* instead
- An itemset X is **closed** in S if X is *frequent* and there exists *no super-pattern* $Y \supset X$, with the same support as X in S (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a **max-pattern** in S if X is frequent and there exists no frequent super-pattern $Y \supset X$ in S (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
 - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Example. $DB = \{ \langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle \}$
 - $Min_sup = 1$.
- What is the set of **closed itemset**?
 - $\langle a_1, \dots, a_{100} \rangle: 1$
 - $\langle a_1, \dots, a_{50} \rangle: 2$
- What is the set of **max-pattern**?
 - $\langle a_1, \dots, a_{100} \rangle: 1$
- What is the set of **all patterns**?
 - !!

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods 
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

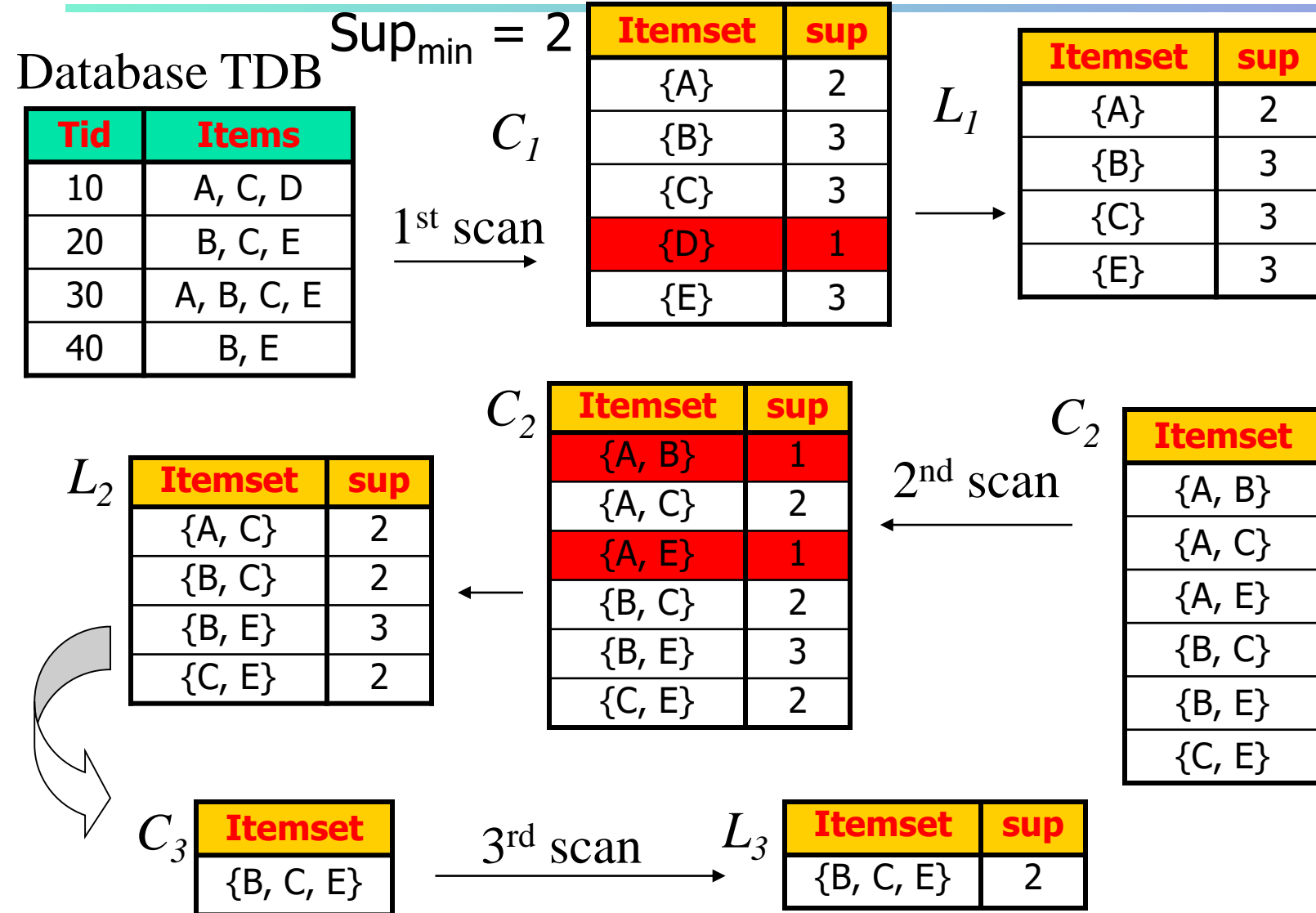
Scalable Methods for Mining Frequent Patterns

- The **downward closure** property of frequent patterns
 - Any subset of a frequent itemset must be frequent
 - If **{bread, milk, cheese}** is frequent, so is **{bread, milk}**
 - i.e., every transaction having {bread, milk, cheese} also contains {bread, milk}
- Scalable mining methods: Three major approaches
 - Apriori (Agrawal & Srikant@VLDB'94)
 - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
 - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is **any** itemset which is infrequent, its superset should not be generated/tested!
(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
 - Initially, scan DB once to get frequent 1-itemset
 - **Generate** length $(k+1)$ **candidate** itemsets from length k **frequent** itemsets
 - **Test** the candidates against DB
 - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



The Apriori Algorithm

- Pseudo-code:

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \emptyset; k++$) **do begin**

C_{k+1} = candidates generated from L_k ;

for each transaction t in database do

increment the count of all candidates in C_{k+1}
that are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}
 - insert into C_k
 - select **$p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$**
 - from **$L_{k-1} p, L_{k-1} q$**
 - where **$p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$**
 $p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning
 - forall ***itemsets* c in C_k** do
 - forall ***(k-1)-subsets* s of c** do
 - if (s is not in L_{k-1}) then delete c from C_k**

Example of Candidate-generation

- $L_3 = \{abc, abd, acd, ace, bcd\}$
- Self-joining: $L_3 * L_3$
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
- Pruning:
 - $acde$ is removed because ade is not in L_3
- $C_4 = \{abcd\}$

Generating Association Rules from Frequent Itemsets

- Generate **strong** association rules from frequent itemsets
 - **Strong** association rules satisfy both minimum support and minimum confidence
- 1. For each frequent itemset I , generate all nonempty subsets of I .
- 2. For every nonempty subset s of I , output the rule " $s \rightarrow (I - s)$ " if $(\text{support_count}(I)/\text{support_count}(s)) \geq \text{min_conf}$

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}.$$

Example

$I = \{I1, I2, I5\}$ Confidence Threshold : 70%

Non empty subsets: $\{I1, I2\}, \{I1, I5\}, \{I2, I5\}$
 $\{I1\}, \{I2\}, \{I5\}$

$I1 \wedge I2 \Rightarrow I5$, confidence = $2 / 4 = 50\%$

$I1 \wedge I5 \Rightarrow I2$, confidence = $2 / 2 = 100\%$

$I2 \wedge I5 \Rightarrow I1$, confidence = $2 / 2 = 100\%$

$I1 \Rightarrow I2 \wedge I5$, confidence = $2 / 6 = 33\%$

$I2 \Rightarrow I1 \wedge I5$, confidence = $2 / 7 = 29\%$

$I5 \Rightarrow I1 \wedge I2$, confidence = $2 / 2 = 100\%$

Challenges of Frequent Pattern Mining

- Challenges
 - Multiple scans of transaction database
 - Huge number of candidates
 - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
 - Reduce passes of transaction database scans
 - Shrink number of candidates
 - Facilitate support counting of candidates

Improving the Efficiency of Apriori

- Hash-based technique
- Transaction reduction
- Partitioning
- Dynamic itemset counting (DIC)

DHP(direct hashing and pruning): Reduce the Number of Candidates

- A k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
 - Candidates: a, b, c, d, e
 - Hash entries
 - {ab, ad, ae}
 - {bd, be, de}
 - ...
 - Frequent 1-itemset: a, b, d, e
 - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold
- J. Park, M. Chen, and P. Yu. *An effective hash-based algorithm for mining association rules. SIGMOD'95*

count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
.	.
.	.
.	.
102	{yz, qs, wt}

Hash Table

Hash-based technique, Example

Create hash table H_2
using hash function
 $h(x, y) = ((\text{order of } x) \times 10 + (\text{order of } y)) \bmod 7$

→

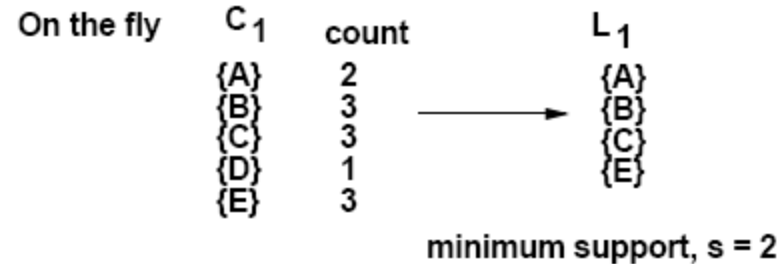
bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3}

Hash table, H_2 , for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining L_1 . If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2 .

DHP: Example of Generating Candidate Itemsets

Database D

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E



Making a hash table

100 {A C}, {A D}, {C D}
 200 {B C}, {B E}, {C E}
 300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
 400 {B E}

$$h\{\{x y\}\} = ((\text{order of } x) * 10 + (\text{order of } y)) \bmod 7;$$

{C E}				{B E}		{A C}
{C E}				{B E}		{C D}
{A D}	{A E}	{B C}		{B E}	{A B}	{A C}
3	1	2	0	3	1	3
0	1	2	3	4	5	6

Hash table H_2
Hash address

The number of items hashed to bucket 2

Generating C_2

$L_1 \times L_1$	# in a bucket with the itemset	C_2
{A B}	1	{A C}
{A C}	3	
{A E}	1	
{B C}	2	
{B E}	3	
{C E}	3	

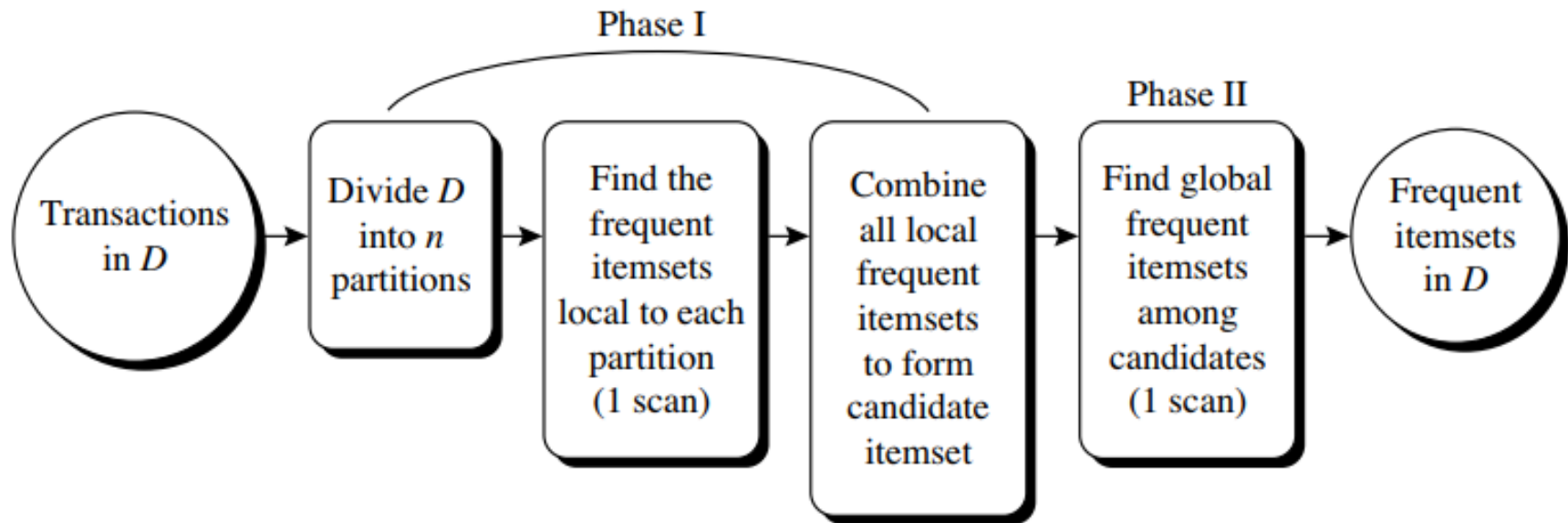
Transaction reduction

- A transaction that does not contain any frequent k -itemsets cannot contain any frequent $(k+1)$ -itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent database scans for j -itemsets, where $j > k$, will not need to consider such a transaction.

Partition: Scan Database Only Twice

- **Definition:** A partition $p \subseteq D$ of the database refers to any subset of the transactions contained in the database D . Any two different partitions are non-overlapping, i.e., $p_i \cap p_j = \emptyset, i \neq j$.
- **Idea:** Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB. *Partition* scans DB only twice.
 - Scan 1: partition database and find local frequent patterns
 - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. *An efficient algorithm for mining association in large databases*. In *VLDB'95*

Partition: Scan Database Only Twice



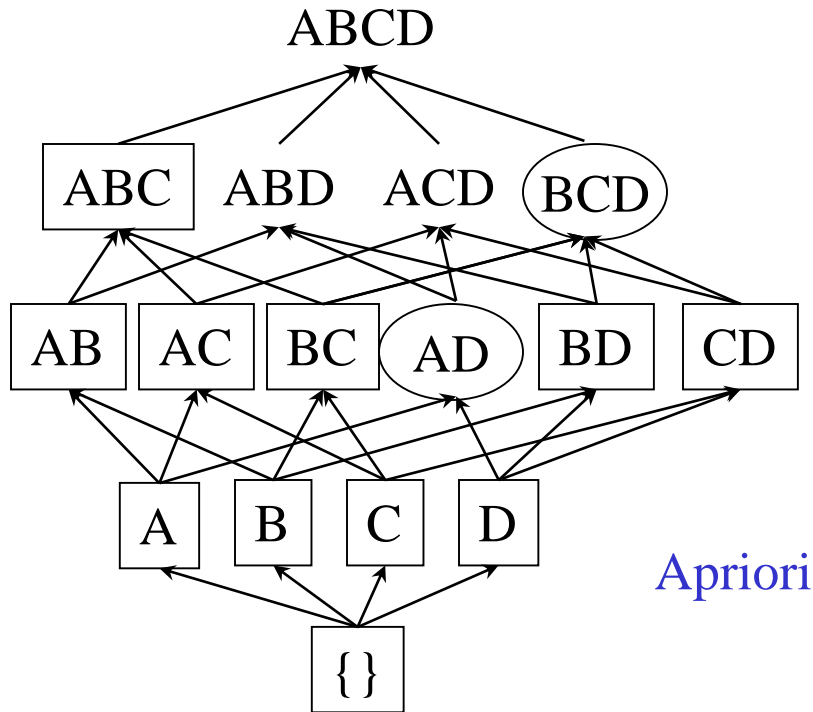
Partition Algorithm: Pseudo code

```
1)  $P = \text{partition\_database}(\mathcal{D})$ 
2)  $n = \text{Number of partitions}$ 
   // Phase I
3) for  $i = 1$  to  $n$  do begin
4)      $\text{read\_in\_partition}(p_i \in P)$ 
5)      $L^i = \text{gen\_large\_itemsets}(p_i)$ 
6) end
   // Merge Phase
7) for ( $i = 2$ ;  $L_i^j \neq \emptyset$ ,  $j = 1, 2, \dots, n$ ;  $i++$ ) do begin
8)      $C_i^G = \cup_{j=1,2,\dots,n} L_i^j$ 
9) end
   // Phase II
10) for  $i = 1$  to  $n$  do begin
11)      $\text{read\_in\_partition}(p_i \in P)$ 
12)     for all candidates  $c \in C^G$   $\text{gen\_count}(c, p_i)$ 
13) end
14)  $L^G = \{c \in C^G | c.\text{count} \geq \text{minSup}\}$ 
15)  $\text{Answer} = L^G$ 
```

DIC

- A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan.
- The technique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.

DIC: Reduce Number of Scans



Itemset lattice

S. Brin R. Motwani, J. Ullman,
and S. Tsur. Dynamic itemset
counting and implication rules for
market basket data. In

SIGMOD'97

May 12, 2021

Apriori

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



1-itemsets

2-itemsets

...

1-itemsets

2-items

3-items

DIC

Bottleneck of Frequent-pattern Mining

- Multiple database scans are **costly**
- Mining long patterns needs many passes of scanning and generates lots of candidates
 - To find frequent itemset $i_1 i_2 \dots i_{100}$
 - # of scans: **100**
 - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$
= $1.27 * 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

- Adopts a *divide-and-conquer* strategy
 - Compress the database representing frequent items into a frequent pattern tree (FP-tree)
 - Divide the compressed database into a set of conditional databases
 - Mine each such database separately
- Grow long patterns from short ones using local frequent items
 - "abc" is a frequent pattern
 - Get all transactions having "abc": DB|abc
 - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

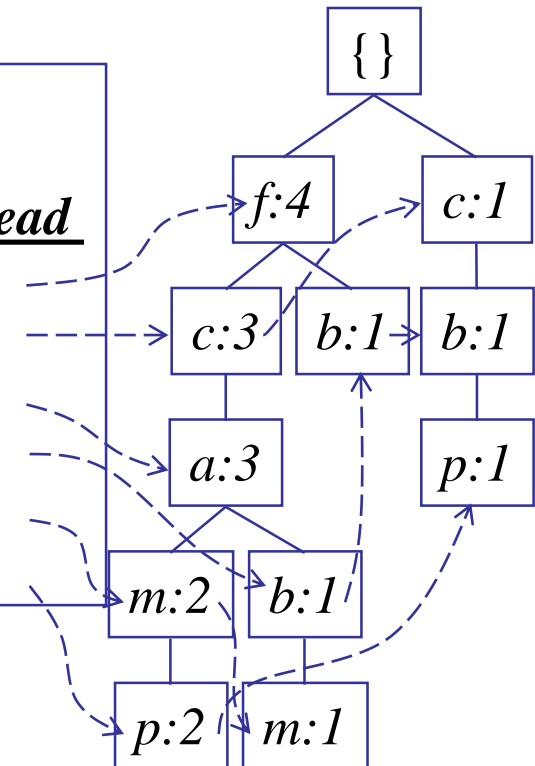
min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

<i>Item</i>	<i>frequency</i>	<i>head</i>
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

F-list=f-c-a-b-m-p



Benefits of the FP-tree Structure

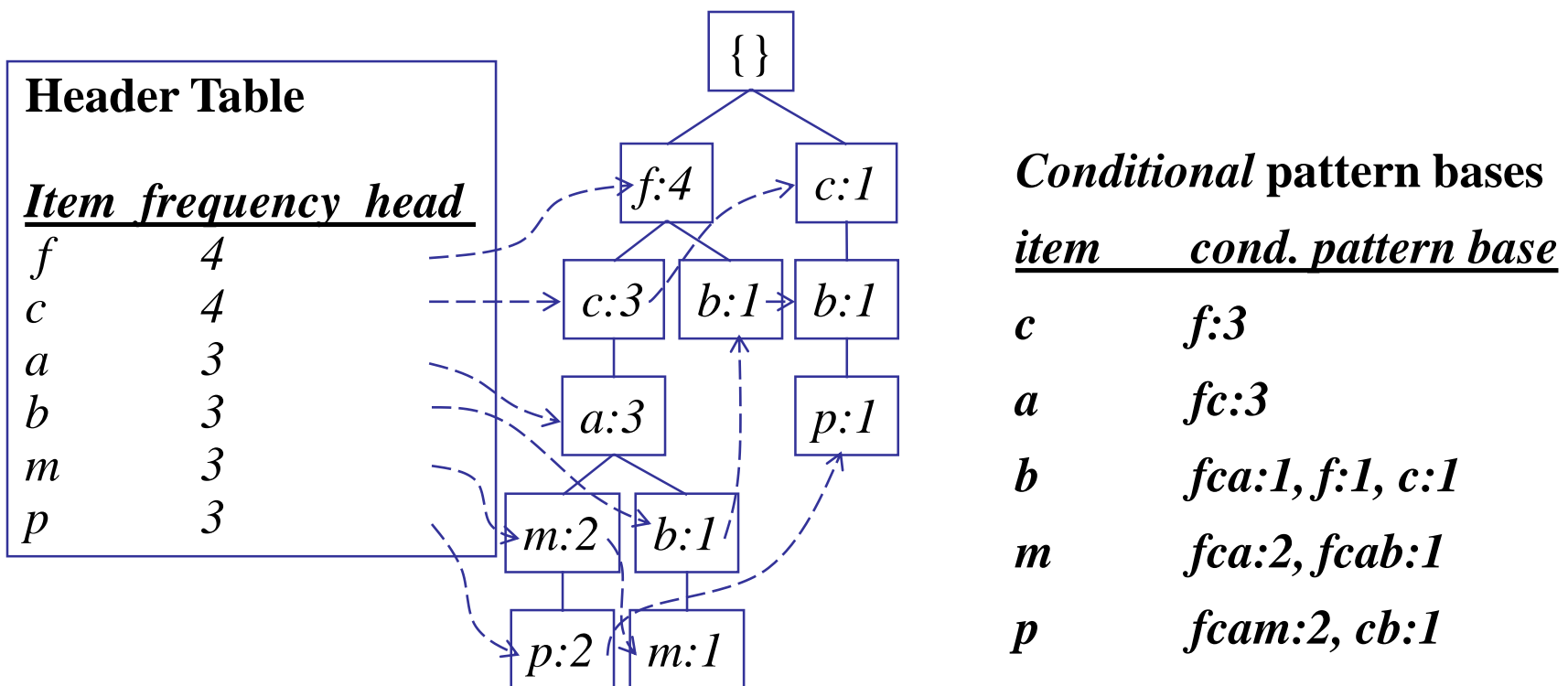
- Completeness
 - Preserve complete information for frequent pattern mining
- Compactness
 - Reduce irrelevant info—infrequent items are gone
 - Items in frequency descending order: the more frequently occurring, the more likely to be shared
 - Never be larger than the original database (not count node-links and the *count* field)

Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
 - F-list=f-c-a-b-m-p
 - Patterns containing p
 - Patterns having m but no p
 - ...
 - Patterns having c but no a nor b, m, p
 - Pattern f
- Completeness and non-redundancy

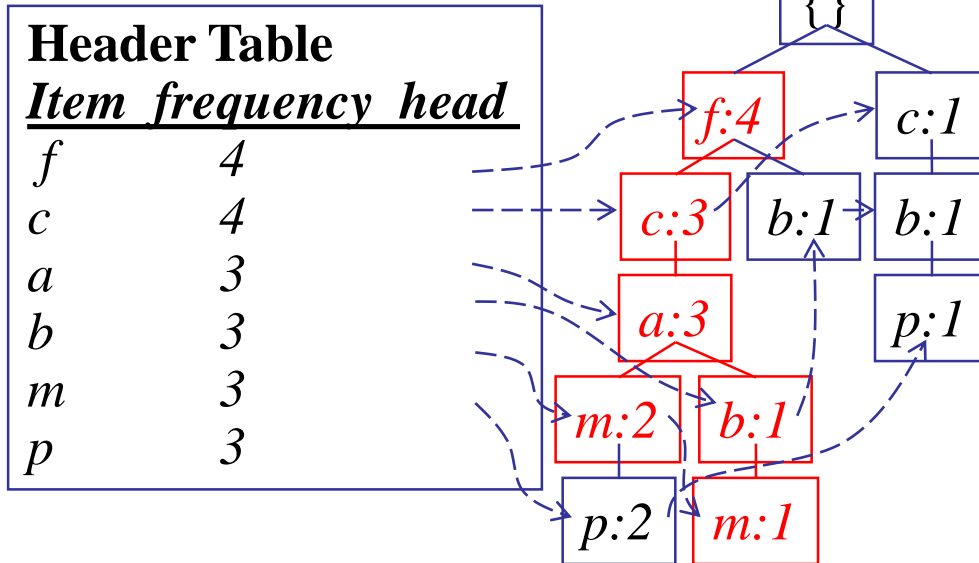
Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item p
- Accumulate all of *transformed prefix paths* of item p to form p 's conditional pattern base



From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
 - Accumulate the count for each item in the base
 - Construct the FP-tree for the frequent items of the pattern base



m-conditional pattern base:
fca:2, fcab:1



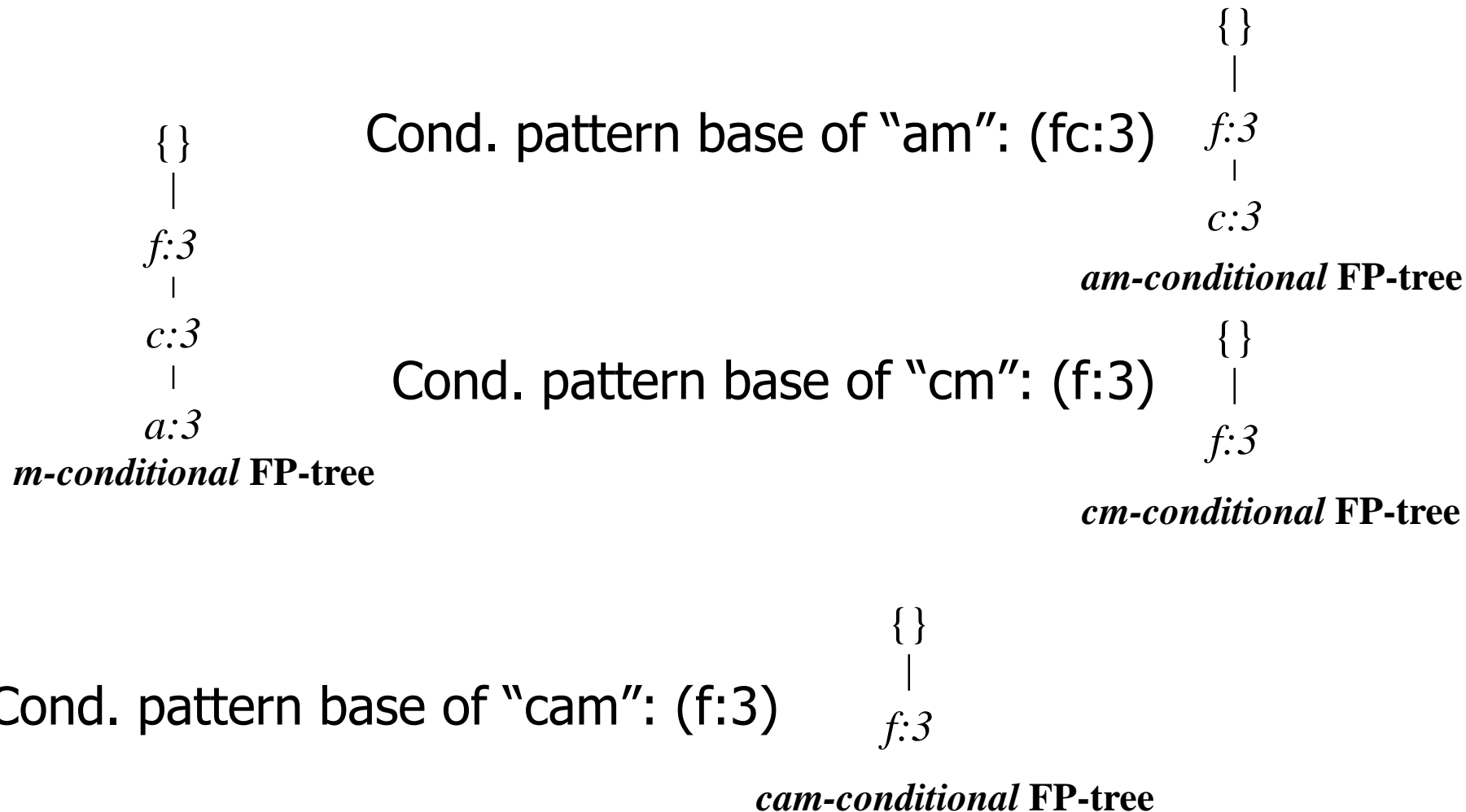
{ }
 |
f:3
 |
c:3
 |
a:3



All frequent patterns relate to *m*
m,
fm, cm, am,
fcm, fam, cam,
fcam

m-conditional FP-tree

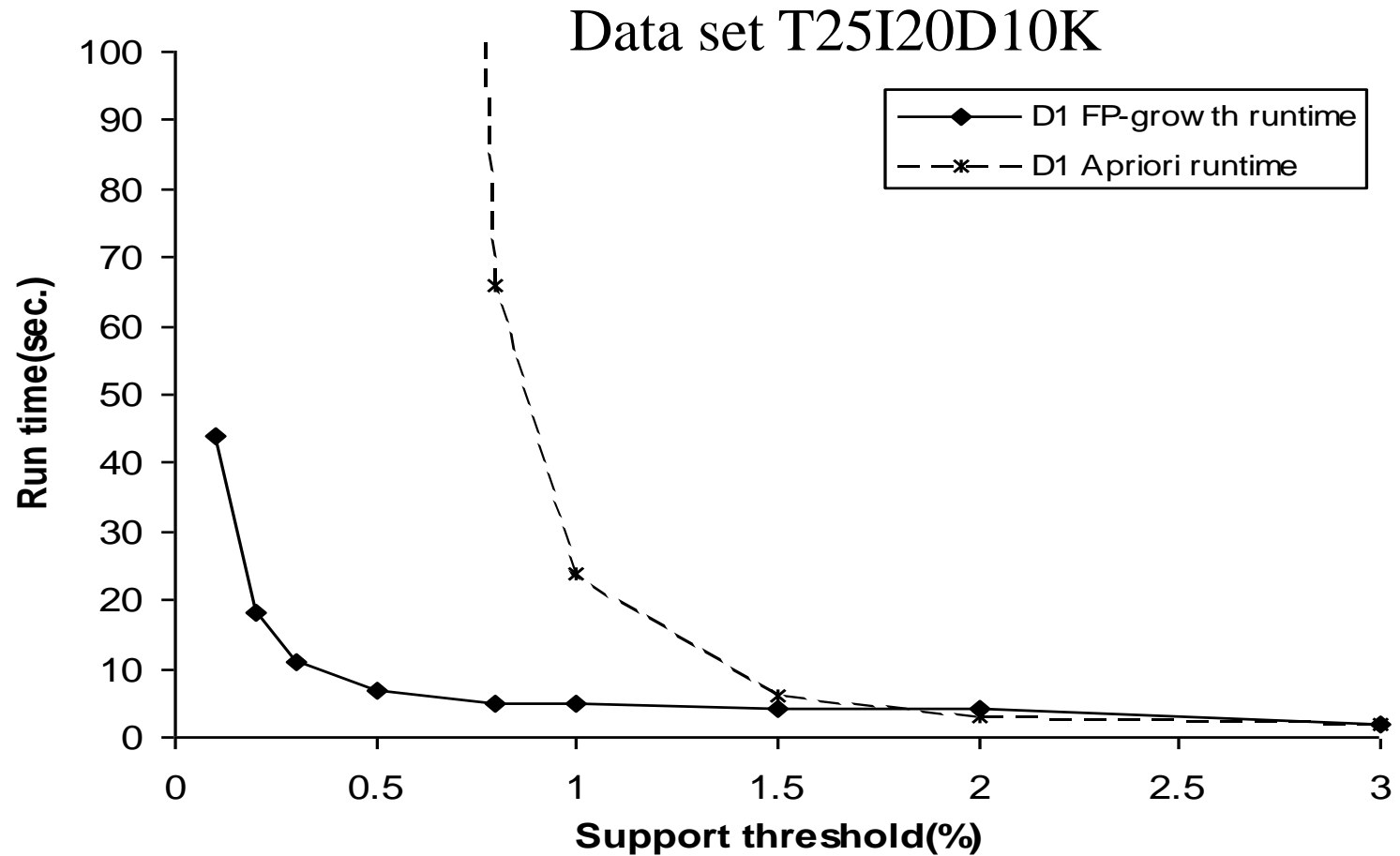
Recursion: Mining Each Conditional FP-tree



Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
 - Recursively grow frequent patterns by pattern and database partition
- Method
 - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
 - Repeat the process on each newly created conditional FP-tree
 - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

FP-Growth vs. Apriori: Scalability With the Support Threshold



Why Is FP-Growth the Winner?

- Divide-and-conquer:
 - decompose both the mining task and DB according to the frequent patterns obtained so far
 - leads to focused search of smaller databases
- Other factors
 - no candidate generation, no candidate test
 - compressed database: FP-tree structure
 - no repeated scan of entire database

Mining Frequent Itemsets Using Vertical Data Format

- Horizontal data format: {TID: itemset}
- Vertical data format: {item: TID_set}
- Frequent itemset mining process:
 - Transform the horizontally formatted data to the vertical format
 - Starting with $k = 1$, construct the candidate $(k+1)$ -itemsets using k -itemsets based on Apriori property
 - Stop when no frequent itemsets or no candidate itemsets can be found

Mining Frequent Itemsets Using Vertical Data Format (Cont'd)

min-sup = 2

Transaction data set

<i>TID</i>	<i>List of item IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Vertical data format of data set

<i>itemset</i>	<i>TID_set</i>
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

<i>itemset</i>	<i>TID_set</i>
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Candidate 3-itemsets

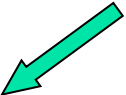
<i>itemset</i>	<i>TID_set</i>
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Candidate 2-itemsets

Mining Frequent Itemsets Using Vertical Data Format (Cont'd)

- The TID-sets can be quite long
 - Take substantial memory space and computation time for intersection
- Use *diffset* to reduce these costs
 - Keep track of only the differences of the TID-sets of a $(k + 1)$ -itemset and a corresponding k -itemset
 - Example
 - $\{I1\} = \{T100, T400, T500, T700, T800, T900\}$
 - $\{I1, I2\} = \{T100, T400, T800, T900\}$
 - $diffset(\{I1, I2\}, \{I1\}) = \{T500, T700\}$.
- Experiments show that in certain situations, such as when the data set contains many dense and long patterns, this technique can substantially reduce the total cost of vertical format mining of frequent itemsets.

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis 
- Constraint-based association mining
- Summary

Interestingness Measure: Correlations (Lift)

- $Min_Supp = 30\%$, $Min_conf = 60\%$
- $buys("computer\ game") \Rightarrow buys("video")$ [40%, 66.7%]
 - This rule is misleading
 - The overall % of customers buying video is 75% > 66.7%.
- Measure of dependent/correlated events: **lift**

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Game	Not game	Sum (row)
Video	4000	3500	7500
Not video	2000	500	2500
Sum(col.)	6000	4000	10000

$$lift(G, V) = \frac{4000 / 10000}{6000 / 10000 * 7500 / 10000} = 0.89 \quad lift(G, \neg V) = \frac{2000 / 10000}{6000 / 10000 * 2500 / 10000} = 1.33$$

Is *lift* A Good Measures of Correlation?

"Buy walnuts \Rightarrow buy milk [1%, 80%]" is misleading if 85% of customers buy milk

Support and confidence are not good to indicate correlations

Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD'02)

Which are good ones?

symbol	measure	range	formula
ϕ	ϕ -coefficient	-1 ... 1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},\bar{B}) + P(A,\bar{B})P(\bar{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A,B) - P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)})$
AV	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
K	Klosgen's Q	-0.33 ... 0.38	$\sqrt{P(A,B)} \max(P(B A) - P(B), P(A B) - P(A))$
g	Goodman-kruskal's	0 ... 1	$\frac{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}{2 - \max_j P(A_j) - \max_k P(B_k)}$
M	Mutual Information	0 ... 1	$\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}$
J	J-Measure	0 ... 1	$\min(-\sum_i P(A_i) \log P(A_i) \log P(A_i), -\sum_i P(B_i) \log P(B_i) \log P(B_i))$
G	Gini index	0 ... 1	$\max(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2, P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2)$
s	support	0 ... 1	$P(A, B)$
c	confidence	0 ... 1	$\max(P(B A), P(A B))$
L	Laplace	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
IS	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all.confidence	0 ... 1	$\frac{P(A,B)}{\max(P(A), P(B))}$
o	odds ratio	0 ... ∞	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(A,\bar{B})}$
V	Conviction	0.5 ... ∞	$\max(\frac{P(A)P(\bar{B})}{P(A\bar{B})}, \frac{P(B)P(\bar{A})}{P(\bar{B}A)})$
λ	lift	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 ... ∞	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$
χ^2	χ^2	0 ... ∞	$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$

Comparison of Interestingness Measures

- Null-(transaction) invariance is crucial for correlation analysis
- Lift is not null-invariant

$$lift = \frac{P(A \cup B)}{P(A) \times P(B)}$$

$$all_conf(A, B) = \frac{sup(A \cup B)}{\max\{sup(A), sup(B)\}}$$

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

Null-transactions
w.r.t. m and c

Null-invariant

DataSet	m, c	~m, c	m, ~c	~m, ~c	lift	all-conf	cosine
D1	10,000	1,000	1,000	100,000	9.26	0.91	0.91
D2	10,000	1,000	1,000	100	1	0.91	0.91
D3	100	1,000	1,000	100,000	8.44	0.09	0.09
D4	1,000	1,000	1,000	100,000	25.75	0.5	0.5

Data Mining:

Concepts and Techniques

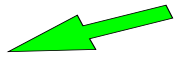
(3rd ed.)

— Chapter 6 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

©2011 Han, Kamber & Pei. All rights reserved.

Chapter 5: Mining Frequent Patterns, Association and Correlations

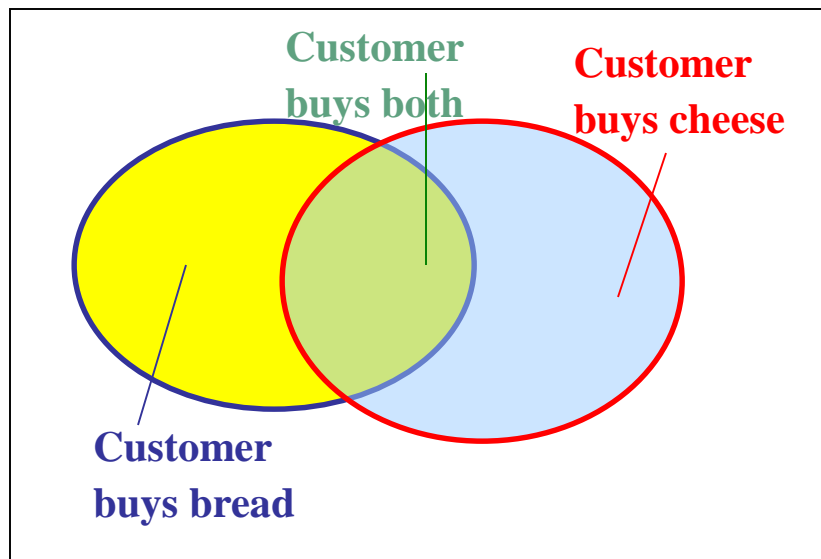
- Basic concepts and a road map 
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of **frequent itemsets** and **association rule mining**
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?— Bread and milk?!
 - What are the subsequent purchases after buying a PC?
- Applications
 - Basket data analysis, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns

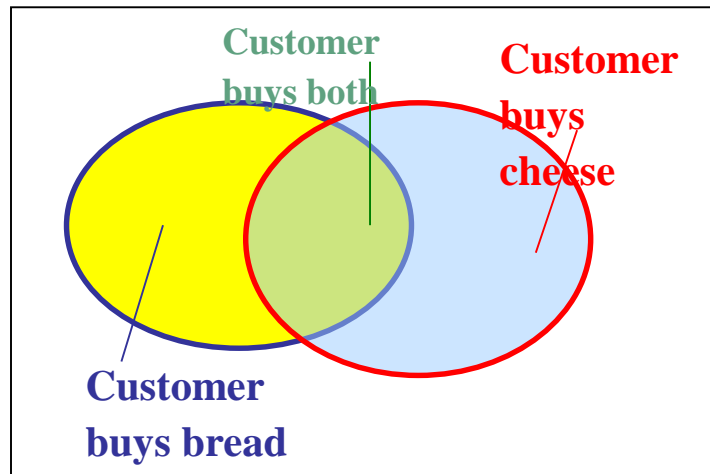
Tid	Items bought
10	Bread, Nuts, Cheese
20	Bread, Coffee, Cheese
30	Bread, Cheese, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Cheese, Eggs, Milk



- **itemset**: A set of one or more items
- **k-itemset** $X = \{x_1, \dots, x_k\}$
- **(absolute) support**, or, **support count** of X : Frequency or occurrence of an itemset X
- **(relative) support**, s , is the fraction of transactions that contains X (i.e., the **probability** that a transaction contains X)
- An itemset X is **frequent** if X 's support is no less than a *minsup* threshold

Basic Concepts: Association Rules

Tid	Items bought
10	Bread, Nuts, Cheese
20	Bread, Coffee, Cheese
30	Bread, Cheese, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Cheese, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - support**, s , **probability** that a transaction contains $X \cup Y$
 - confidence**, c , **conditional probability** that a transaction having X also contains Y

Let $minsup = 50\%$, $minconf = 50\%$

Freq. Pat.: Bread:3, Nuts:3, Cheese:4, Eggs:3, {Bread, Cheese}:3

- Association rules:
 - $Bread \rightarrow Cheese$ (60%, 100%)
 - $Cheese \rightarrow Bread$ (60%, 75%)


Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \dots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}$ sub-patterns!
- Solution: Mine *closed patterns* and *max-patterns* instead
- An itemset X is **closed** in S if X is *frequent* and there exists *no super-pattern* $Y \supset X$, with the same support as X in S (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a **max-pattern** in S if X is frequent and there exists no frequent super-pattern $Y \supset X$ in S (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
 - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Example. $DB = \{ \langle a_1, \dots, a_{100} \rangle, \langle a_1, \dots, a_{50} \rangle \}$
 - $Min_sup = 1$.
- What is the set of **closed itemset**?
 - $\langle a_1, \dots, a_{100} \rangle: 1$
 - $\langle a_1, \dots, a_{50} \rangle: 2$
- What is the set of **max-pattern**?
 - $\langle a_1, \dots, a_{100} \rangle: 1$
- What is the set of **all patterns**?
 - !!

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods 
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

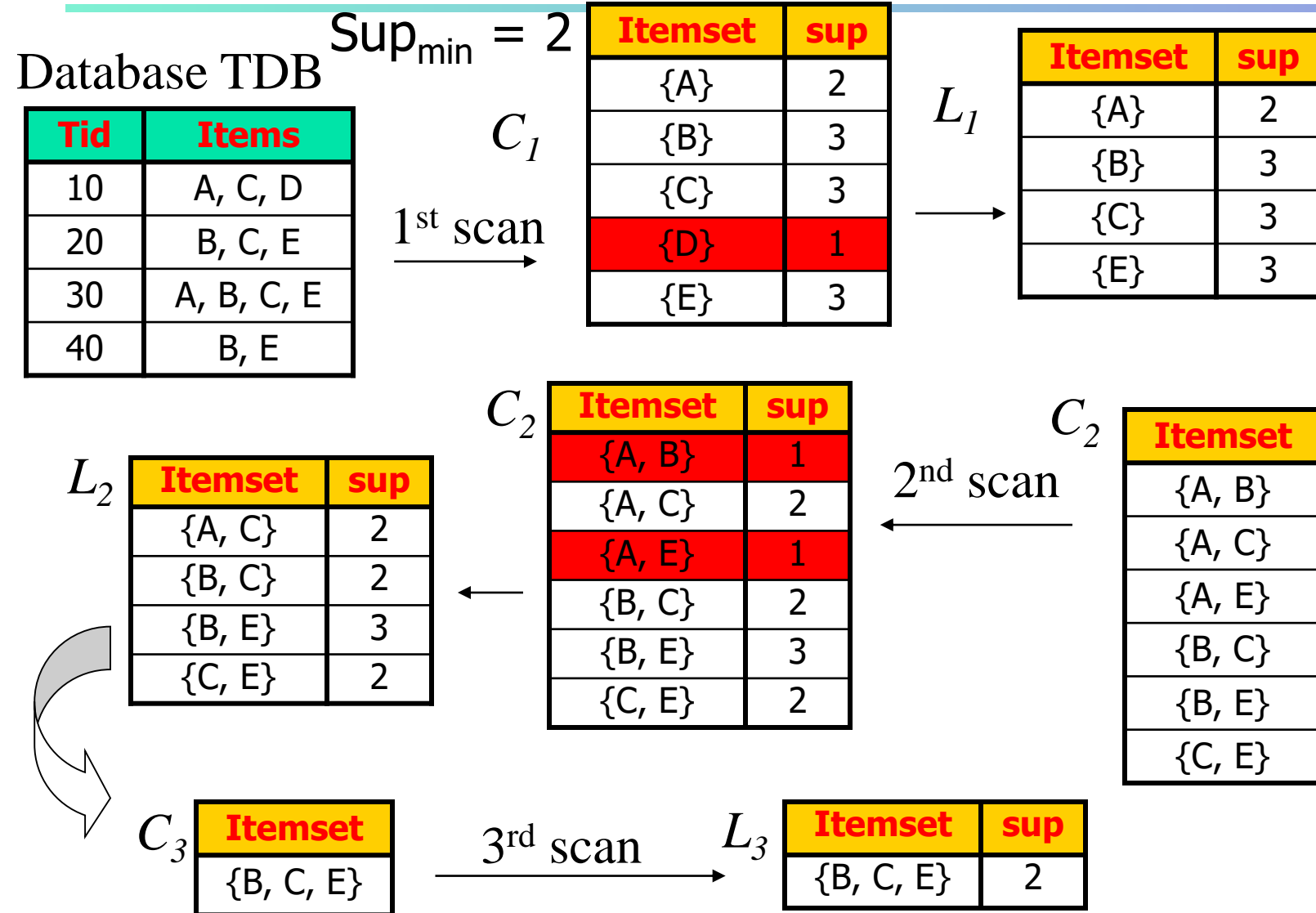
Scalable Methods for Mining Frequent Patterns

- The **downward closure** property of frequent patterns
 - Any subset of a frequent itemset must be frequent
 - If **{bread, milk, cheese}** is frequent, so is **{bread, milk}**
 - i.e., every transaction having {bread, milk, cheese} also contains {bread, milk}
- Scalable mining methods: Three major approaches
 - Apriori (Agrawal & Srikant@VLDB'94)
 - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
 - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is **any** itemset which is infrequent, its superset should not be generated/tested!
(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
 - Initially, scan DB once to get frequent 1-itemset
 - **Generate** length $(k+1)$ **candidate** itemsets from length k **frequent** itemsets
 - **Test** the candidates against DB
 - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



The Apriori Algorithm

- Pseudo-code:

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \emptyset; k++$) **do begin**

C_{k+1} = candidates generated from L_k ;

for each transaction t in database **do**

increment the count of all candidates in C_{k+1}
that are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}
 - insert into C_k
 - select **$p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$**
 - from **$L_{k-1} p, L_{k-1} q$**
 - where **$p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$**
 $p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning
 - forall ***itemsets* c in C_k** do
 - forall ***(k-1)-subsets* s of c** do
 - if (s is not in L_{k-1}) then delete c from C_k**

Example of Candidate-generation

- $L_3 = \{abc, abd, acd, ace, bcd\}$
- Self-joining: $L_3 * L_3$
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
- Pruning:
 - $acde$ is removed because ade is not in L_3
- $C_4 = \{abcd\}$

Generating Association Rules from Frequent Itemsets

- Generate **strong** association rules from frequent itemsets
 - **Strong** association rules satisfy both minimum support and minimum confidence
- 1. For each frequent itemset I , generate all nonempty subsets of I .
- 2. For every nonempty subset s of I , output the rule " $s \rightarrow (I - s)$ " if $(\text{support_count}(I)/\text{support_count}(s)) \geq \text{min_conf}$

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{\text{support_count}(A \cup B)}{\text{support_count}(A)}.$$

Example

$I = \{I1, I2, I5\}$ Confidence Threshold : 70%

Non empty subsets: $\{I1, I2\}, \{I1, I5\}, \{I2, I5\}$
 $\{I1\}, \{I2\}, \{I5\}$

$I1 \wedge I2 \Rightarrow I5$, confidence = $2 / 4 = 50\%$

$I1 \wedge I5 \Rightarrow I2$, confidence = $2 / 2 = 100\%$

$I2 \wedge I5 \Rightarrow I1$, confidence = $2 / 2 = 100\%$

$I1 \Rightarrow I2 \wedge I5$, confidence = $2 / 6 = 33\%$

$I2 \Rightarrow I1 \wedge I5$, confidence = $2 / 7 = 29\%$

$I5 \Rightarrow I1 \wedge I2$, confidence = $2 / 2 = 100\%$

Challenges of Frequent Pattern Mining

- Challenges
 - Multiple scans of transaction database
 - Huge number of candidates
 - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
 - Reduce passes of transaction database scans
 - Shrink number of candidates
 - Facilitate support counting of candidates

Improving the Efficiency of Apriori

- Hash-based technique
- Transaction reduction
- Partitioning
- Dynamic itemset counting (DIC)

DHP(direct hashing and pruning): Reduce the Number of Candidates

- A k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
 - Candidates: a, b, c, d, e
 - Hash entries
 - {ab, ad, ae}
 - {bd, be, de}
 - ...
 - Frequent 1-itemset: a, b, d, e
 - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold
- J. Park, M. Chen, and P. Yu. *An effective hash-based algorithm for mining association rules. SIGMOD'95*

count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
.	.
.	.
.	.
102	{yz, qs, wt}

Hash Table

Hash-based technique, Example

Create hash table H_2
using hash function
 $h(x, y) = ((\text{order of } x) \times 10 + (\text{order of } y)) \bmod 7$

→

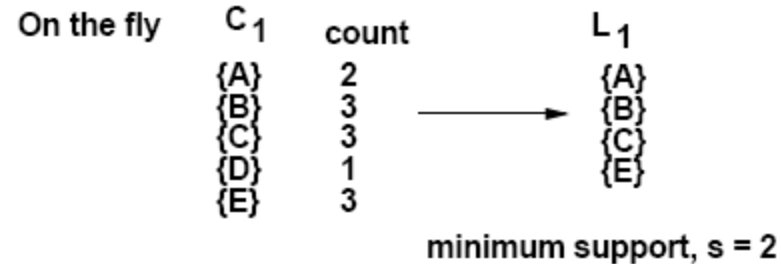
bucket address	0	1	2	3	4	5	6
bucket count	2	2	4	2	2	4	4
bucket contents	{I1, I4} {I3, I5}	{I1, I5} {I1, I5}	{I2, I3} {I2, I3} {I2, I3} {I2, I3}	{I2, I4} {I2, I4}	{I2, I5} {I2, I5}	{I1, I2} {I1, I2} {I1, I2}	{I1, I3} {I1, I3} {I1, I3}

Hash table, H_2 , for candidate 2-itemsets. This hash table was generated by scanning Table 6.1's transactions while determining L_1 . If the minimum support count is, say, 3, then the itemsets in buckets 0, 1, 3, and 4 cannot be frequent and so they should not be included in C_2 .

DHP: Example of Generating Candidate Itemsets

Database D

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E



Making a hash table

100 {A C}, {A D}, {C D}
 200 {B C}, {B E}, {C E}
 300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
 400 {B E}

$$h\{\{x y\}\} = ((\text{order of } x) * 10 + (\text{order of } y)) \bmod 7;$$

{C E}				{B E}		{A C}
{C E}				{B E}		{C D}
{A D}	{A E}	{B C}		{B E}	{A B}	{A C}
3	1	2	0	3	1	3
0	1	2	3	4	5	6

Hash table H_2
Hash address

The number of items hashed to bucket 2

Generating C_2

$L_1 \times L_1$	# in a bucket with the itemset	C_2
{A B}	1	→ {A C} {B C} {B E} {C E}
{A C}	3	
{A E}	1	
{B C}	2	
{B E}	3	
{C E}	3	

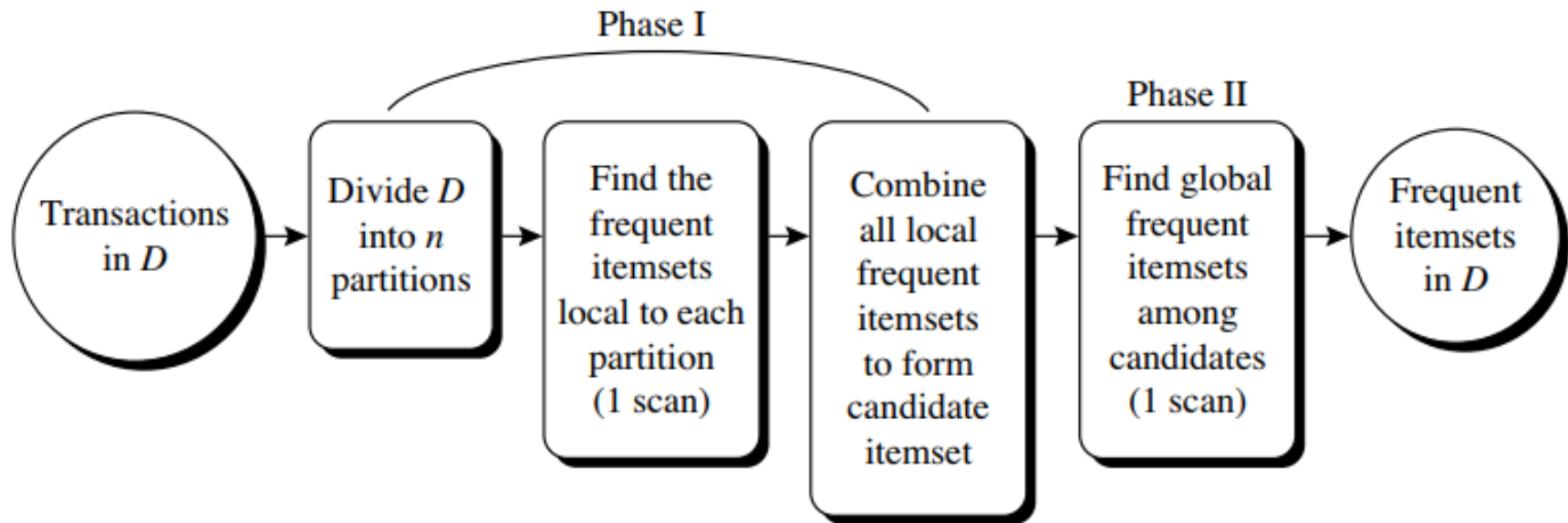
Transaction reduction

- A transaction that does not contain any frequent k -itemsets cannot contain any frequent $(k+1)$ -itemsets. Therefore, such a transaction can be marked or removed from further consideration because subsequent database scans for j -itemsets, where $j > k$, will not need to consider such a transaction.

Partition: Scan Database Only Twice

- **Definition:** A partition $p \subseteq D$ of the database refers to any subset of the transactions contained in the database D . Any two different partitions are non-overlapping, i.e., $p_i \cap p_j = \emptyset, i \neq j$.
- **Idea:** Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB. *Partition* scans DB only twice.
 - Scan 1: partition database and find local frequent patterns
 - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. *An efficient algorithm for mining association in large databases*. In *VLDB'95*

Partition: Scan Database Only Twice



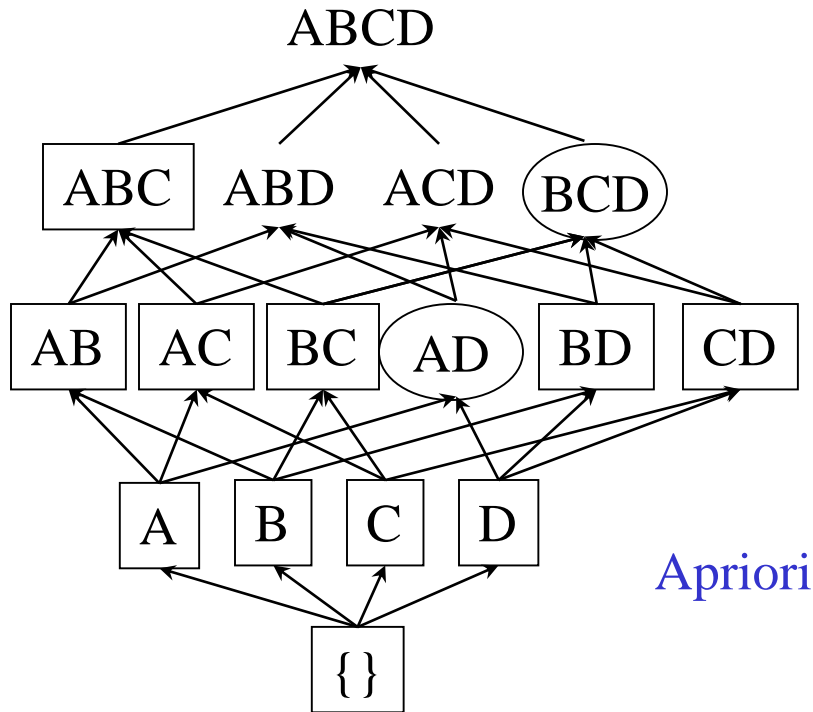
Partition Algorithm: Pseudo code

```
1)  $P = \text{partition\_database}(\mathcal{D})$ 
2)  $n = \text{Number of partitions}$ 
   // Phase I
3) for  $i = 1$  to  $n$  do begin
4)      $\text{read\_in\_partition}(p_i \in P)$ 
5)      $L^i = \text{gen\_large\_itemsets}(p_i)$ 
6) end
   // Merge Phase
7) for ( $i = 2$ ;  $L_i^j \neq \emptyset$ ,  $j = 1, 2, \dots, n$ ;  $i++$ ) do begin
8)      $C_i^G = \cup_{j=1,2,\dots,n} L_i^j$ 
9) end
   // Phase II
10) for  $i = 1$  to  $n$  do begin
11)      $\text{read\_in\_partition}(p_i \in P)$ 
12)     for all candidates  $c \in C^G$   $\text{gen\_count}(c, p_i)$ 
13) end
14)  $L^G = \{c \in C^G | c.\text{count} \geq \text{minSup}\}$ 
15)  $\text{Answer} = L^G$ 
```

DIC

- A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately before each complete database scan.
- The technique uses the count-so-far as the lower bound of the actual count. If the count-so-far passes the minimum support, the itemset is added into the frequent itemset collection and can be used to generate longer candidates. This leads to fewer database scans than with Apriori for finding all the frequent itemsets.

DIC: Reduce Number of Scans



Itemset lattice

S. Brin R. Motwani, J. Ullman,
and S. Tsur. Dynamic itemset
counting and implication rules for
market basket data. In

SIGMOD'97

May 12, 2021

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins



1-itemsets

2-itemsets

...

1-itemsets

2-items

3-items

DIC

Bottleneck of Frequent-pattern Mining

- Multiple database scans are **costly**
- Mining long patterns needs many passes of scanning and generates lots of candidates
 - To find frequent itemset $i_1 i_2 \dots i_{100}$
 - # of scans: **100**
 - # of Candidates: $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1$
 $= 1.27 * 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?

Mining Frequent Patterns Without Candidate Generation

- Adopts a *divide-and-conquer* strategy
 - Compress the database representing frequent items into a frequent pattern tree (FP-tree)
 - Divide the compressed database into a set of conditional databases
 - Mine each such database separately
- Grow long patterns from short ones using local frequent items
 - “abc” is a frequent pattern
 - Get all transactions having “abc”: DB|abc
 - “d” is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

<i>TID</i>	<i>Items bought</i>	<i>(ordered) frequent items</i>
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

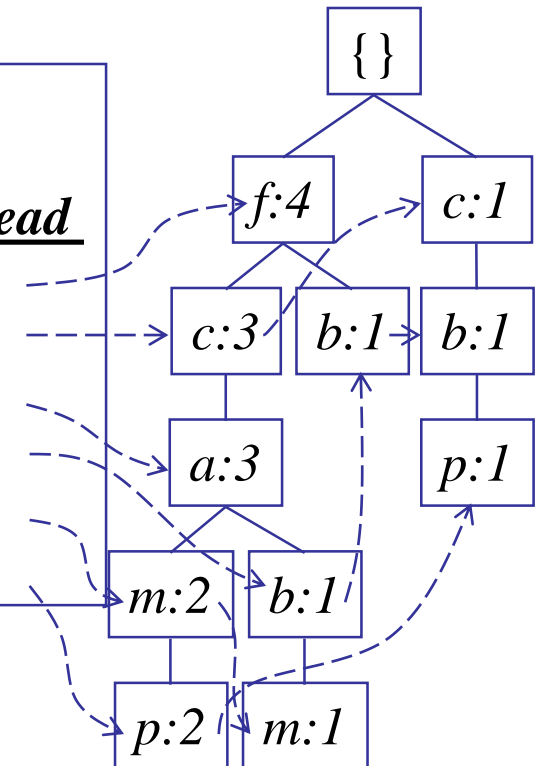
min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

Header Table

<i>Item</i>	<i>frequency</i>	<i>head</i>
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

F-list=f-c-a-b-m-p



Benefits of the FP-tree Structure

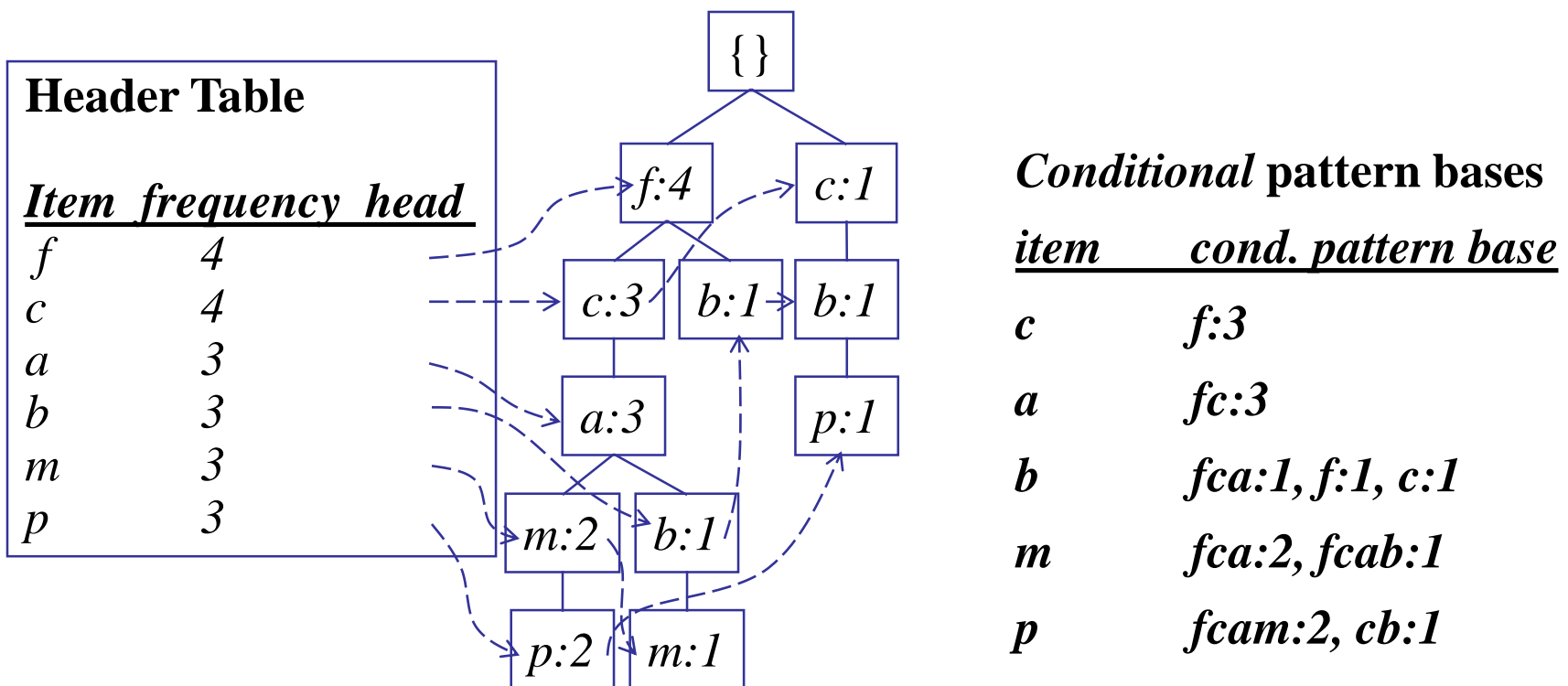
- Completeness
 - Preserve complete information for frequent pattern mining
- Compactness
 - Reduce irrelevant info—infrequent items are gone
 - Items in frequency descending order: the more frequently occurring, the more likely to be shared
 - Never be larger than the original database (not count node-links and the *count* field)

Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
 - F-list=f-c-a-b-m-p
 - Patterns containing p
 - Patterns having m but no p
 - ...
 - Patterns having c but no a nor b, m, p
 - Pattern f
- Completeness and non-redundancy

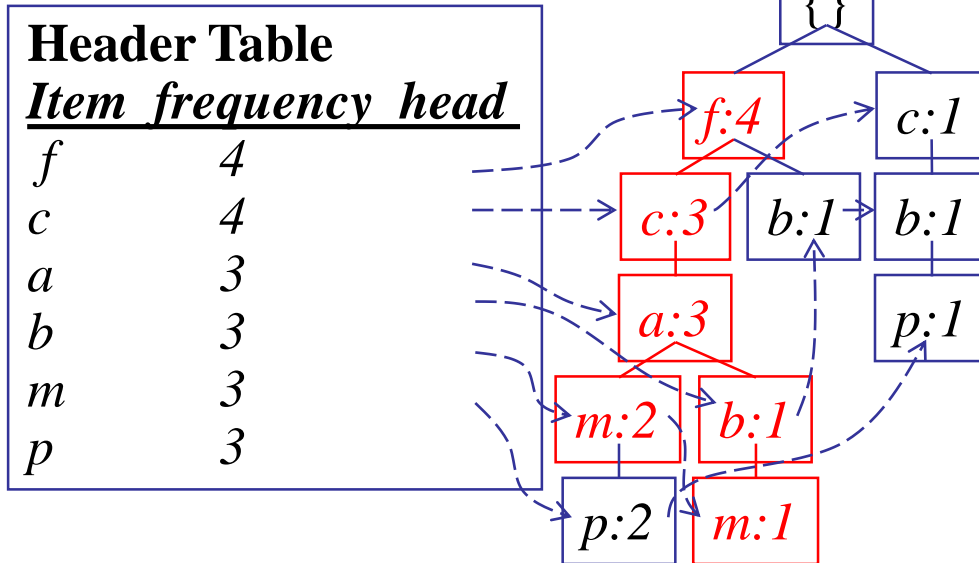
Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item p
- Accumulate all of *transformed prefix paths* of item p to form p 's conditional pattern base



From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
 - Accumulate the count for each item in the base
 - Construct the FP-tree for the frequent items of the pattern base



m-conditional pattern base:
fca:2, fcab:1



{}

|

f:3

|

c:3

|

a:3



All frequent patterns relate to *m*

m,

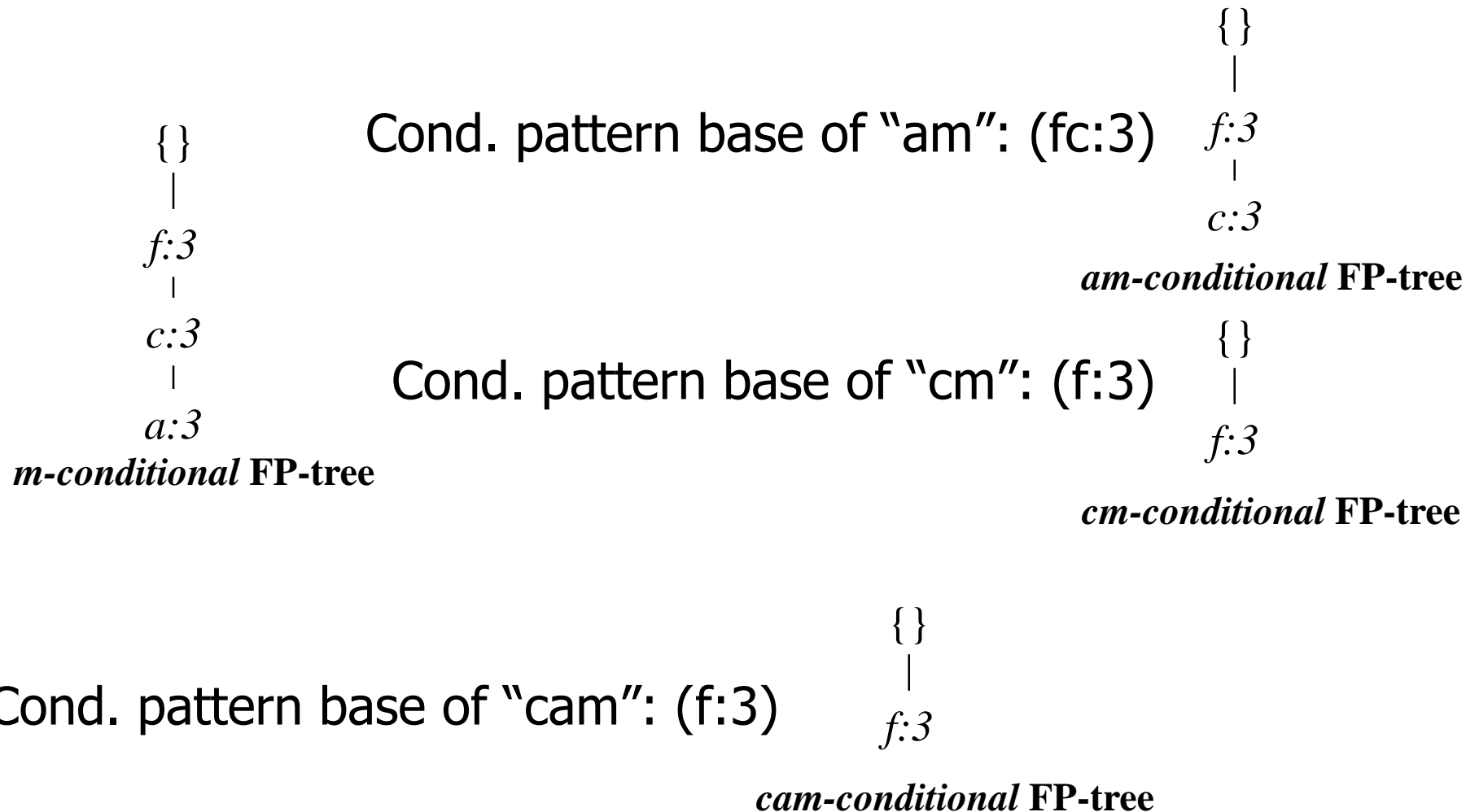
fm, cm, am,

fcm, fam, cam,

fcam

m-conditional FP-tree

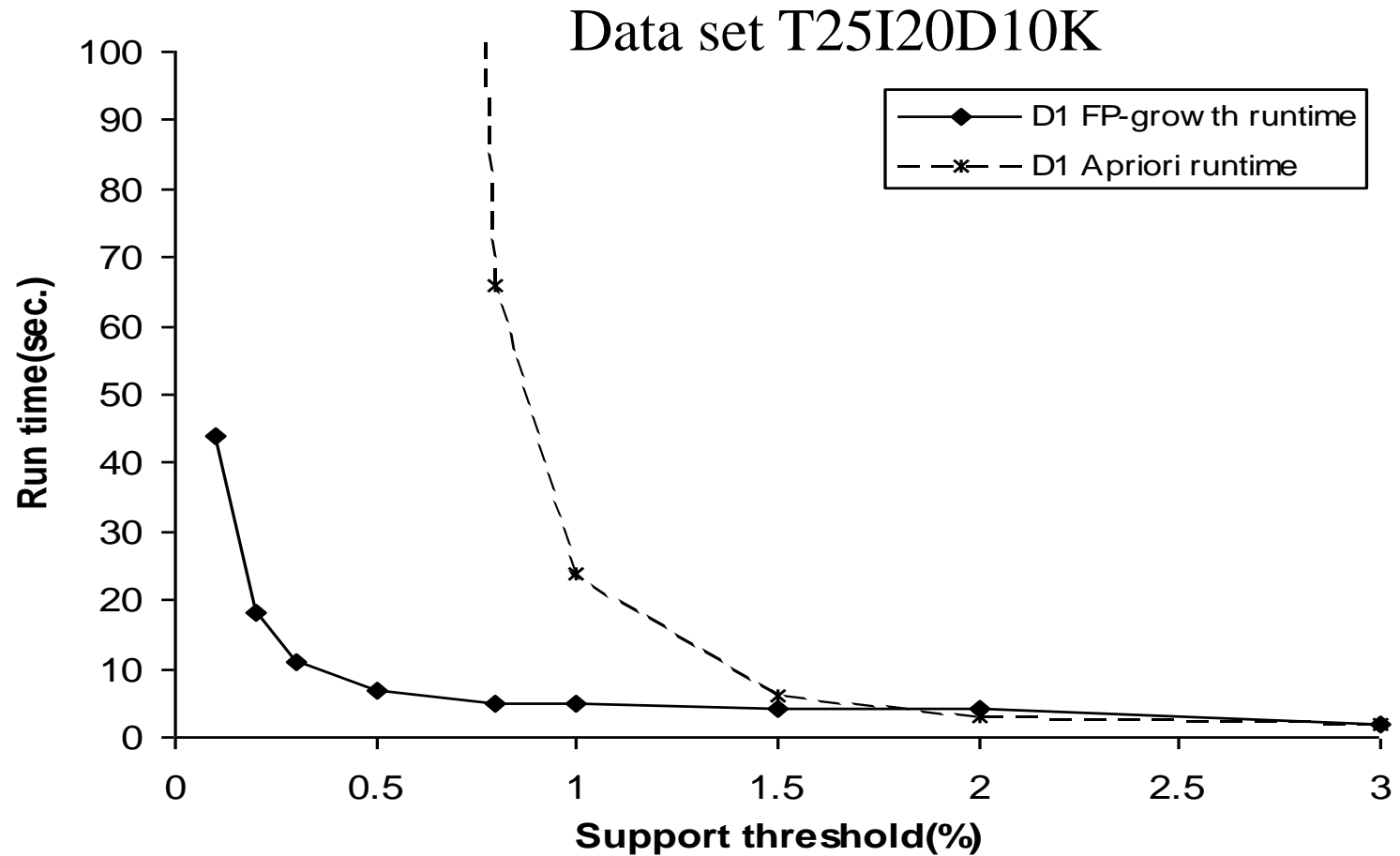
Recursion: Mining Each Conditional FP-tree



Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
 - Recursively grow frequent patterns by pattern and database partition
- Method
 - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
 - Repeat the process on each newly created conditional FP-tree
 - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

FP-Growth vs. Apriori: Scalability With the Support Threshold



Why Is FP-Growth the Winner?

- Divide-and-conquer:
 - decompose both the mining task and DB according to the frequent patterns obtained so far
 - leads to focused search of smaller databases
- Other factors
 - no candidate generation, no candidate test
 - compressed database: FP-tree structure
 - no repeated scan of entire database

Mining Frequent Itemsets Using Vertical Data Format

- Horizontal data format: {TID: itemset}
- Vertical data format: {item: TID_set}
- Frequent itemset mining process:
 - Transform the horizontally formatted data to the vertical format
 - Starting with $k = 1$, construct the candidate $(k+1)$ -itemsets using k -itemsets based on Apriori property
 - Stop when no frequent itemsets or no candidate itemsets can be found

Mining Frequent Itemsets Using Vertical Data Format (Cont'd)

min-sup = 2

Transaction data set

<i>TID</i>	<i>List of item IDs</i>
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Vertical data format of data set

<i>itemset</i>	<i>TID_set</i>
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

<i>itemset</i>	<i>TID_set</i>
{I1, I2, I3}	{T800, T900}
{I1, I2, I5}	{T100, T800}

Candidate 3-itemsets

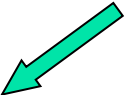
<i>itemset</i>	<i>TID_set</i>
{I1, I2}	{T100, T400, T800, T900}
{I1, I3}	{T500, T700, T800, T900}
{I1, I4}	{T400}
{I1, I5}	{T100, T800}
{I2, I3}	{T300, T600, T800, T900}
{I2, I4}	{T200, T400}
{I2, I5}	{T100, T800}
{I3, I5}	{T800}

Candidate 2-itemsets

Mining Frequent Itemsets Using Vertical Data Format (Cont'd)

- The TID-sets can be quite long
 - Take substantial memory space and computation time for intersection
- Use *diffset* to reduce these costs
 - Keep track of only the differences of the TID-sets of a $(k + 1)$ -itemset and a corresponding k -itemset
 - Example
 - $\{I1\} = \{T100, T400, T500, T700, T800, T900\}$
 - $\{I1, I2\} = \{T100, T400, T800, T900\}$
 - $diffset(\{I1, I2\}, \{I1\}) = \{T500, T700\}$.
- Experiments show that in certain situations, such as when the data set contains many dense and long patterns, this technique can substantially reduce the total cost of vertical format mining of frequent itemsets.

Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis 
- Constraint-based association mining
- Summary

Interestingness Measure: Correlations (Lift)

- $Min_Supp = 30\%$, $Min_conf = 60\%$
- $buys("computer\ game") \Rightarrow buys("video")$ [40%, 66.7%]
 - This rule is misleading
 - The overall % of customers buying video is 75% > 66.7%.
- Measure of dependent/correlated events: **lift**

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

	Game	Not game	Sum (row)
Video	4000	3500	7500
Not video	2000	500	2500
Sum(col.)	6000	4000	10000

$$lift(G, V) = \frac{4000 / 10000}{6000 / 10000 * 7500 / 10000} = 0.89 \quad lift(G, \neg V) = \frac{2000 / 10000}{6000 / 10000 * 2500 / 10000} = 1.33$$

Is *lift* A Good Measures of Correlation?

- "Buy walnuts \Rightarrow buy milk [1%, 80%]" is misleading if 85% of customers buy milk
- Support and confidence are not good to indicate correlations
- Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD'02)
- Which are good ones?

symbol	measure	range	formula
ϕ	ϕ -coefficient	-1 ... 1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},\bar{B}) + P(A,\bar{B})P(\bar{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A,B) - P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)})$
AV	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
K	Klogsen's Q	-0.33 ... 0.38	$\sqrt{P(A,B)} \max(P(B A) - P(B), P(A B) - P(A))$
g	Goodman-kruskal's	0 ... 1	$\frac{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}{2 - \max_j P(A_j) - \max_k P(B_k)}$
M	Mutual Information	0 ... 1	$\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{\sum_i P(A_i) \log \frac{P(A_i)}{P(A)} + \sum_j P(B_j) \log \frac{P(B_j)}{P(B)}}$
J	J-Measure	0 ... 1	$\max(P(A,B) \log(\frac{P(B A)}{P(B)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{B} \bar{A})}{P(\bar{B})}))$
G	Gini index	0 ... 1	$P(A,B) \log(\frac{P(A B)}{P(A)}) + P(\bar{A}\bar{B}) \log(\frac{P(\bar{A} \bar{B})}{P(\bar{A})})$
s	support	0 ... 1	$\max(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2,$
c	confidence	0 ... 1	$P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2)$
L	Laplace	0 ... 1	$\max(P(B A), P(A B))$
IS	Cosine	0 ... 1	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
α	all.confidence	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
o	odds ratio	0 ... ∞	$\frac{\max(P(A), P(B))}{P(A,B)}$
V	Conviction	0.5 ... ∞	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(A,\bar{B})}$
λ	lift	0 ... ∞	$\max(\frac{P(A)P(\bar{B})}{P(A\bar{B})}, \frac{P(B)P(\bar{A})}{P(\bar{B}\bar{A})})$
S	Collective strength	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
χ^2	χ^2	0 ... ∞	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$
			$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$

Comparison of Interestingness Measures

- Null-(transaction) invariance is crucial for correlation analysis
- Lift is not null-invariant

$$lift = \frac{P(A \cup B)}{P(A) \times P(B)}$$

$$all_conf(A, B) = \frac{sup(A \cup B)}{\max\{sup(A), sup(B)\}}$$

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}}$$

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

Null-transactions
w.r.t. m and c

Null-invariant

DataSet	m, c	~m, c	m, ~c	~m, ~c	lift	all-conf	cosine
D1	10,000	1,000	1,000	100,000	9.26	0.91	0.91
D2	10,000	1,000	1,000	100	1	0.91	0.91
D3	100	1,000	1,000	100,000	8.44	0.09	0.09
D4	1,000	1,000	1,000	100,000	25.75	0.5	0.5