UNIT-IV
Classification and Prediction

6.1 What is classification? What is prediction?

Classification:

*used for prediction(future analysis) to know the unknown attributes with their values.by using classifier algorithms and decision tree.(in data mining)
*which constructs some models(like decision trees) then which classifies the attributes..
*already we know the types of attributes are 1.categorical attribute and 2.numerical attribute
*these classification can work on both the above mentioned attributes.

Prediction: prediction also used for to know the unknown or missing values..

1. which also uses some models in order to predict the attributes
2. models like neural networks, if else rules and other mechanisms

Classification and prediction are used in the Applications like

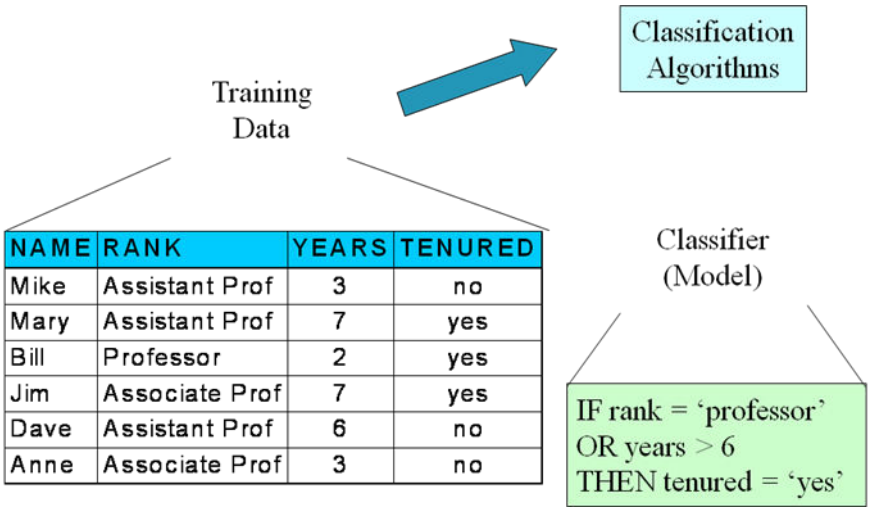
*credit approval
*target marketing
*medical diagnosis

Classification—A Two-Step Process

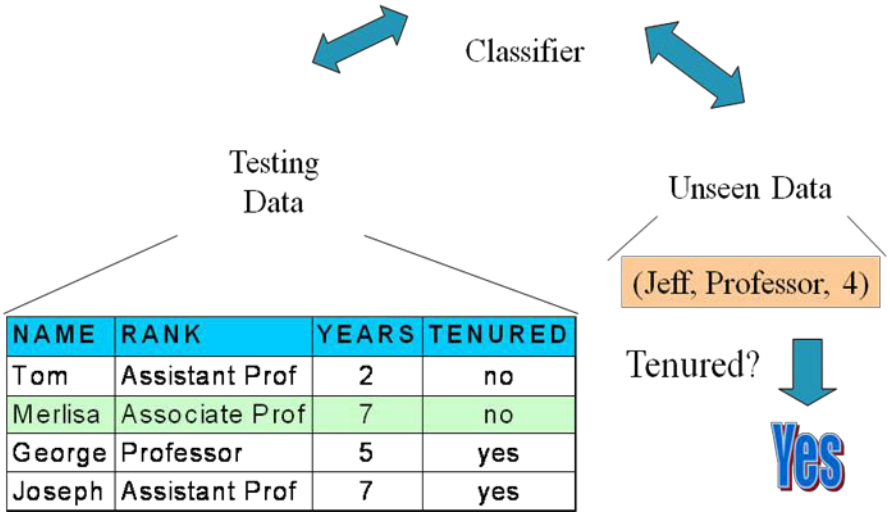
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, 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

Process (1): Model Construction



Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

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

6.2 Issues regarding classification and prediction:

There are two issues regarding classification and prediction they are
Issues (1): Data Preparation
Issues (2): Evaluating Classification Methods

Issues (1): Data Preparation: Issues of data preparation includes the following

1) Data cleaning

   *Preprocess data in order to reduce noise and handle missing values (refer preprocessing techniques i.e. data cleaning notes)

2) Relevance analysis (feature selection)

Remove the irrelevant or redundant attributes (refer unit-iv AOI Relevance analysis)

3) Data transformation (refer preprocessing techniques i.e data cleaning notes) Generalize and/or normalize data

Issues (2): Evaluating Classification Methods: considering classification methods should satisfy the following properties

1. **Predictive accuracy**
2. **Speed and scalability**
   *time to construct the model
   *time to use the model
3. **Robustness**
   *handling noise and missing values
4. **Scalability**
   *efficiency in disk-resident databases
5. **Interpretability**:
   *understanding and insight provided by the model
6. **Goodness of rules**
   *decision tree size
   *compactness of classification rules

6.3 Classification by Decision Tree Induction

6.3.1 Decision tree

- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample

- Test the attribute values of the sample against the decision tree

Training Dataset

This follows an example from Quinlan's ID3

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

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
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - 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

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- 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 node holds the class prediction
- Rules are easier for humans to understand

- Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

IF *age* = "31…40" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid over fitting
  - 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
  - Post pruning: 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"

Tree Mining in Weka and Tree Mining in Clementine.

Tree Mining in Weka

- Example:
  - Weather problem: build a decision tree to guide the decision about whether or not to play tennis.
  - Dataset (weather.nominal.arff)

- Validation:

  - Using training set as a test set will provide optimal classification accuracy.
  - Expected accuracy on a different test set will always be less.
  - 10-fold cross validation is more robust than using the training set as a test set.
    - Divide data into 10 sets with about same proportion of class label values as in original set.
    - Run classification 10 times independently with the remaining 9/10 of the set as the training set.
    - Average accuracy.
  - Ratio validation: 67% training set / 33% test set.
  - Best: having a separate training set and test set.

- Results:
  - Classification accuracy (correctly classified instances).
  - Errors (absolute mean, root squared mean, …)
  - Kappa statistic (measures agreement between predicted and observed classification; -100%-100% is the proportion of agreements after chance agreement has been excluded; 0% means complete agreement by chance)
- Results:
  - TP (True Positive) rate per class label
  - FP (False Positive) rate
  - Precision = TP rate = TP / (TP + FN)) * 100%
  - Recall = TP / (TP + FP)) * 100%
  - F-measure = 2* recall * precision / recall + precision
- ID3 characteristics:
  - Requires nominal values
  - Improved into C4.5
    - Dealing with numeric attributes
    - Dealing with missing values
    - Dealing with noisy data
    - Generating rules from trees

Tree Mining in Clementine

- Methods:

- C5.0: target field must be categorical, predictor fields may be numeric or categorical, provides multiple splits on the field that provides the maximum information gain at each level
- QUEST: target field must be categorical, predictor fields may be numeric ranges or categorical, statistical binary split
- C&RT: target and predictor fields may be numeric ranges or categorical, statistical binary split based on regression
- CHAID: target and predictor fields may be numeric ranges or categorical, statistical binary split based on chi-square

6.3.2 Attribute Selection Measures

- Information Gain
- Gain ratio
- Gini Index

6.3.3 Pruning of decision trees

Discarding one or more sub trees and replacing them with leaves simplify a decision tree, and that is the main task in decision-tree pruning. In replacing the sub tree with a leaf, the algorithm expects to lower the *predicted error rate and* increase the quality of a classification model. But computation of error rate is not simple. An error rate based only on a training data set does not provide a suitable estimate. One possibility to estimate the predicted error rate is to use a new, additional set of test samples if they are available, or to use the cross-validation techniques. This technique divides initially available samples into equal sized blocks and, for each block, the tree is constructed from all samples except this block and tested with a given block of samples. With the available training and testing samples, the basic idea of decision tree-pruning is to remove parts of the tree (sub trees) that do not contribute to the classification accuracy of unseen testing samples, producing a less complex and thus more comprehensible tree. There are two ways in which the recursive-partitioning method can be modified:

1. Deciding not to divide a set of samples any further under some conditions. The stopping criterion is usually based on some statistical tests, such as the χ_2 test: *If there are no significant differences in classification accuracy before and after division, then represent a current node as a leaf*. The decision is

made in advance, before splitting, and therefore this approach is called *pre pruning*.

2. Removing restrospectively some of the tree structure using selected accuracy criteria. The decision in this process of *post pruning* is made after the tree has been built.

C4.5 follows the *post pruning* approach, but it uses a specific technique to estimate the predicted error rate. This method is called *pessimistic pruning*. For every node in a tree, the estimation of the upper confidence limit u_{cf} is computed using the statistical tables for binomial distribution (given in most textbooks on statistics). Parameter $U_{25\%}(|T_i|/E)$ for a given node T_i with a confidence level of 25%, and compares $U_{25\%}(|T_i|/E)$ for a given node T_i with a leaf. If the predicted error for a root node in a sub tree is less than weighted sum of $U_{25\%}$ for the leaves (predicted error for the sub tree), then a sub tree will be replaced with its root node, which becomes a new leaf in a pruned tree.

Let us illustrate this procedure with one simple example. A sub tree of a decision tree is given in Figure, where the root node is the test x_1 on three possible values {1, 2, 3} of the attribute A. The children of the root node are leaves denoted with corresponding classes and ($|T_i|/E$) parameters. The question is to estimate the possibility of pruning the sub tree and replacing it with its root node as a new, generalized leaf node.

To analyze the possibility of replacing the sub tree with a leaf node it is necessary to compute a predicted error PE for the initial tree and for a replaced node. Using default confidence of 25%, the upper confidence limits for all nodes are collected from statistical tables: $U_{25\%}(6, 0) = 0.206$, $U_{25\%}(9, 0) = 0.143$, $U_{25\%}(1, 0) = 0.750$, and $U_{25\%}(16, 1) = 0.157$. Using these values, the predicted errors for the initial tree and the replaced node are

$$PE_{tree} = 6.0.206 + 9.0.143 + 1.0.750 = 3.257$$

$$PE_{node} = 16.0.157 = 2.512$$

Since the existing subtree has a higher value of predicted error than the replaced node, it is recommended that the decision tree be pruned and the subtree replaced with the new leaf node.

6.4 Bayesian Classification:

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by th**eir probabilities**
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

6.4.1 Bayesian Theorem

- Given training data *D, posteriori probability of a hypothesis h, P(h|D)* follows the Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayes Classifier (I)

- A simplified assumption: attributes are conditionally independent:

$$P(C_j|V) \propto P(C_j)\prod_{i=1}^{n} P(v_i|C_j)$$

- Greatly reduces the computation cost, only count the class distribution.

Naive Bayesian Classifier (II)

Given a training set, we can compute the probabilities

Outlook	P	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Temperature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	P	N
high	3/9	4/5
normal	6/9	1/5
Windy		
true	3/9	3/5
false	6/9	2/5

Bayesian classification

- The classification problem may be formalized using a-posteriori probabilities:
- P(C|X) = prob. that the sample tuple
- X=<x1,…,xk> is of class C.
- E.g. P(class=N | outlook=sunny, windy=true,…)
- Idea: assign to sample X the class label C such that P(C|X) is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$P(C|X) = P(X|C)\cdot P(C) / P(X)$

- P(X) is constant for all classes
- P(C) = relative freq of class C samples
- C such that P(C|X) is maximum =
  C such that P(X|C)·P(C) is maximum
- Problem: computing P(X|C) is unfeasible!

6.4.2 Naïve Bayesian Classification

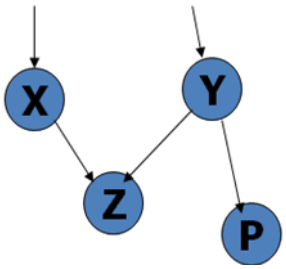
- Naïve assumption: attribute independence

$P(x1,\ldots,xk|C) = P(x1|C)\cdot\ldots\cdot P(xk|C)$

- If i-th attribute is categorical:
  P(xi|C) is estimated as the relative freq of samples having value xi as i-th attribute in class C
- If i-th attribute is continuous:
  P(xi|C) is estimated thru a Gaussian density function
- Computationally easy in both cases

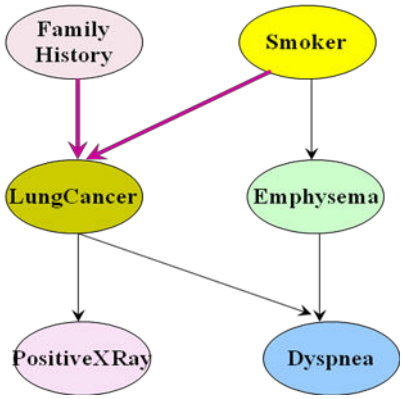
6.4.3 Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops or cycles

Bayesian Belief Network: An Example



The conditional probability table (CPT) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of X, from CPT:

$$P(x_1,...,x_n) = \prod_{i=1}^{n} P(xi \mid Parents(Yi))$$

Association-Based Classification

- Several methods for association-based classification
  - ARCS: Quantitative association mining and clustering of association rules (Lent et al'97)
    - It beats C4.5 in (mainly) scalability and also accuracy
  - Associative classification: (Liu et al'98)
    - It mines high support and high confidence rules in the form of "cond_set => y", where y is a class label
  - CAEP (Classification by aggregating emerging patterns) (Dong et al'99)
    - Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
    - Mine Eps based on minimum support and growth rate

6.5 Rule Based Classification

6.5.1 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

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

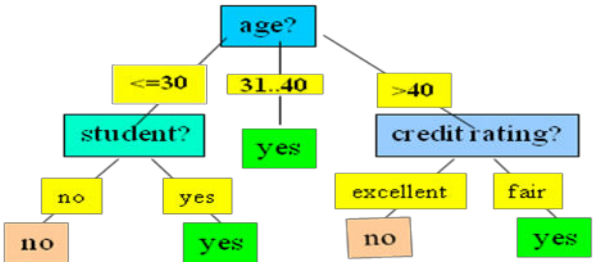
accuracy(R) = $n_{correct}$ / n_{covers}

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

6.5.2 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* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

6.5.3 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*

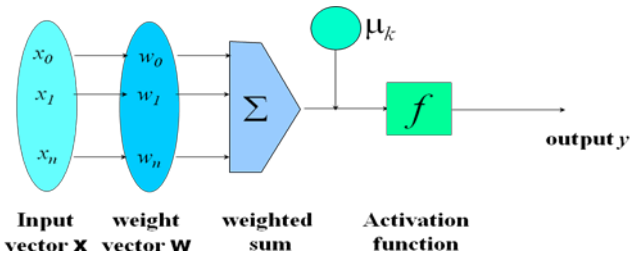
6.6 Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

Neural Network as a Classifier

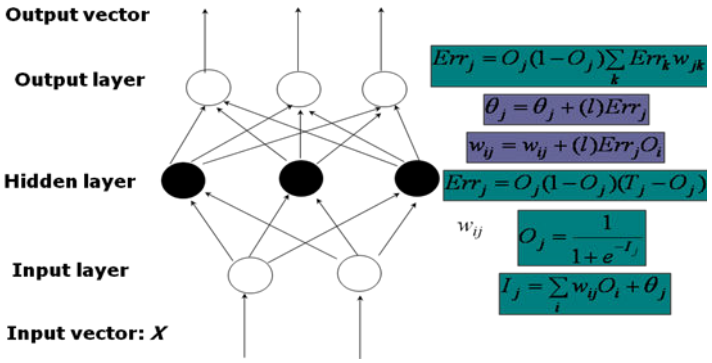
- Weakness
    - Long training time
    - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
    - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network
- Strength
    - High tolerance to noisy data
    - Ability to classify untrained patterns
    - Well-suited for continuous-valued inputs and outputs
    - Successful on a wide array of real-world data
    - Algorithms are inherently parallel
    - Techniques have recently been developed for the extraction of rules from trained neural networks

A Neuron (= a perceptron)



- The *n*-dimensional input vector x is mapped into variable y by means of the scalar product and a nonlinear function mapping

6.6.1 A Multi-Layer Feed-Forward Neural Network



- The inputs to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the input layer
- They are then weighted and fed simultaneously to a hidden layer
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network's prediction
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform nonlinear regression: Given enough hidden units and enough training samples, they can closely approximate any function

6.6.2 Backpropagation

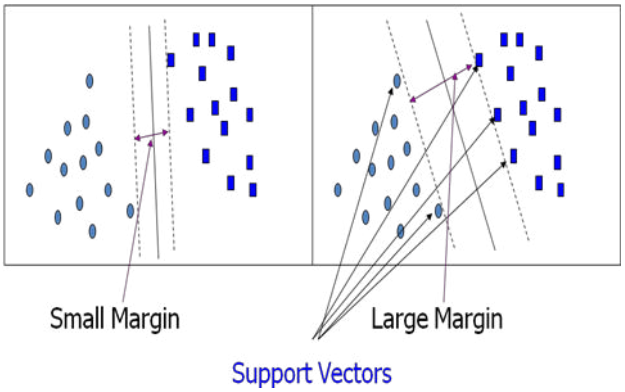
- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value
- Modifications are made in the "backwards" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "backpropagation"
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Back propagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)
- Efficiency of backpropagation: Each epoch (one interaction through the training set) takes O(|D| * *w*), with |D| tuples and *w* weights, but # of epochs can be exponential to n, the number of inputs, in the worst case
- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

6.7 SVM—Support Vector Machines

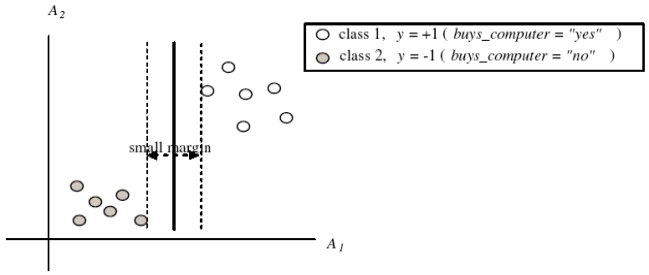
- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyper plane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyper plane
- SVM finds this hyper plane using support vectors ("essential" training tuples) and margins (defined by the support vectors)
- **Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)**
- **Used both for classification and prediction**
- **Applications:**

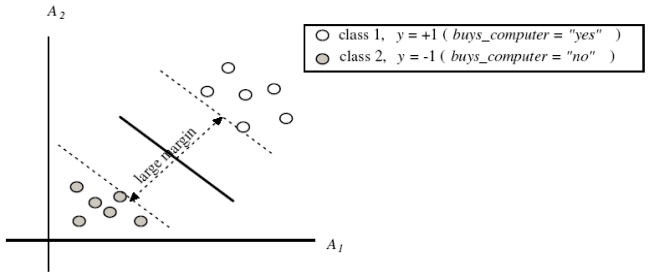
- handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

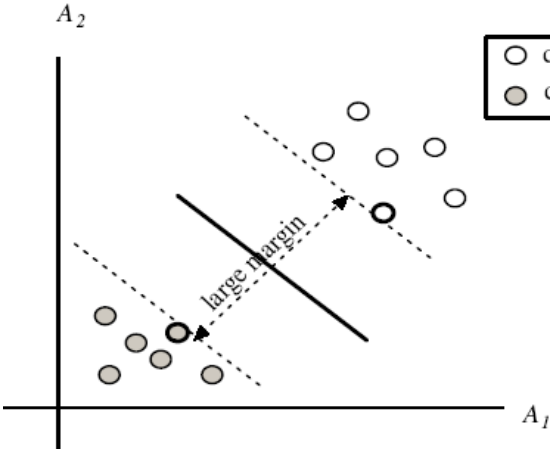
SVM—General Philosophy



SVM—Margins and Support Vectors







6.7.1 SVM—Linearly Separable

- A separating hyper plane can be written as

$W \bullet X + b = 0$

where $W=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)

- For 2-D it can be written as

$w_0 + w_1 x_1 + w_2 x_2 = 0$

- The hyper plane defining the sides of the margin:

H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- Any training tuples that fall on hyper planes H_1 or H_2 (i.e., the sides defining the margin) are support vectors
- This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyper plane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

6.8 Associative Classification

- Associative classification
  - Association rules are generated and analyzed for use in classification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Classification: Based on evaluating a set of rules in the form of

$P_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{class} = C$" (conf, sup)

- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time

In many studies, associative classification has been found to be more accurate than some traditional classification methods, such as C4.

Typical Associative Classification Methods

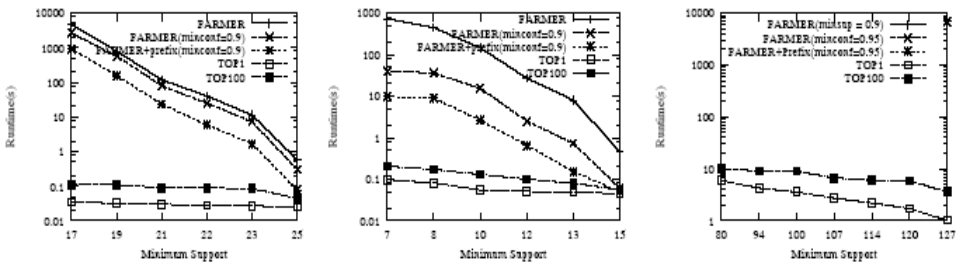
- CBA (Classification By Association: Liu, Hsu & Ma, KDD'98)
  - Mine association possible rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- CPAR (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis)
  - High efficiency, accuracy similar to CMAR
- RCBT (Mining top-*k* covering rule groups for gene expression data, Cong et al. SIGMOD'05)
  - Explore high-dimensional classification, using top-k rule groups
  - Achieve high classification accuracy and high run-time efficiency

Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

Table 2: Classification Results



(a) ALL-AML leukemia (b) Lung Cancer (c) Ovarian Cancer

6.9 Lazy Learners

6.9.1 The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist(X_1, 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

- 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
- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

6.9.2.**Case-Based Reasoning:**

1. Also uses: lazy evaluation + analyze similar instances
2. Difference: Instances are not "points in a Euclidean space"

6.10 Other Classification Methods

6.10.1 Genetic Algorithms

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
  - If an attribute has k > 2 values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offspring are generated by *crossover* and *mutation*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

6.10.2 Rough Set Approach:

- Rough sets are used to **approximately or "roughly" define equivalent classes**
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)

Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity

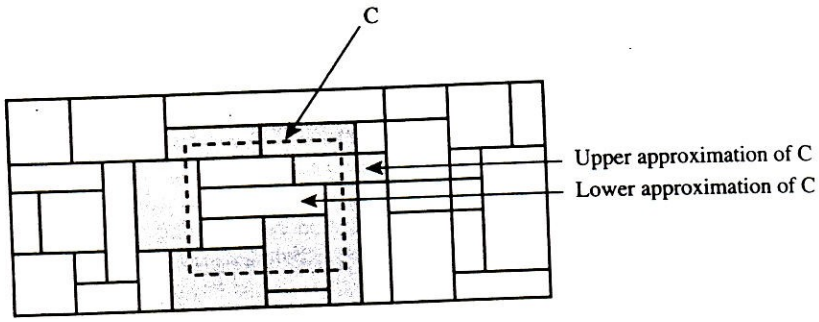
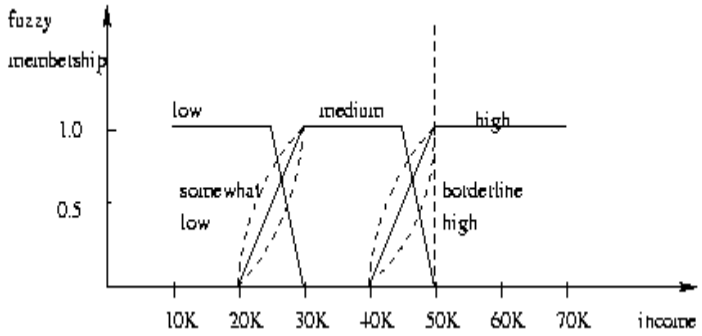


Figure: A rough set approximation of the set of tuples of the class C suing lower and upper approximation sets of C. The rectangular regions represent equivalence classes

6.10.3 Fuzzy Set approaches

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined



6.11 Prediction

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or predictor variables and a *dependent* or response variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

6.11.1 Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x

$y = w_0 + w_1 x$

where w_0 (y-intercept) and w_1 (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line
  - Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1), (\mathbf{X_2}, y_2), \ldots, (\mathbf{X_{|D|}}, y_{|D|})$
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above

6.11.2 Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$

convertible to linear with new variables: $x_2 = x_2$, $x_3 = x_3$

$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

UNIT V

Cluster Analysis

7.1 What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

General Applications of Clustering

- Pattern Recognition
- Spatial Data Analysis
  - create thematic maps in GIS by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

Examples of Clustering Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location

- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Requirements of Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

7.2 Type of data in clustering analysis

- Interval-scaled variables:
- Binary variables:
- Nominal, ordinal, and ratio variables:
- Variables of mixed types:

7.2.1 Interval-valued variables

- Standardize data
  - Calculate the mean absolute deviation:

Where $$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf}).$$ —

  - Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \ldots + |x_{i_p} - x_{j_p}|^q)}$$

where i = (xi1, xi2, …, xip) and j = (xj1, xj2, …, xjp) are two p-dimensional data objects, and q is a positive integer

- *If q = 1, d is Manhattan distance*

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

$i_2 \quad j_2 \qquad\qquad\qquad\qquad i_p$

j_p

- *If q = 2, d is Euclidean distance:*

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

$$s_f = \frac{1}{n_f}(|x_{1f} - m_f|^2 + |x_{2f} - m_f|^2 + \ldots + |x_{nf} - m_f|^2)$$

– *Properties*

- $d(i,j) \geq 0$
- $d(i,i) = 0$
- $d(i,j) = d(j,i)$
- $d(i,j) \leq d(i,k) + d(k,j)$

- *Also one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures.*

7.2.2 Binary Variables

A contingency table for binary data

		Object j	
	1	0	*sum*
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

Object i

- *Simple matching coefficient (invariant, if the binary variable is symmetric):*
- *Jaccard coefficient (noninvariant if the binary variable is asymmetric):*

Dissimilarity between Binary Variables

- **Example**

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

– gender is a symmetric attribute
– the remaining attributes are asymmetric binary
– let the values Y and P be set to 1, and the value N be set to 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) =$$

$$d(jim, mary) =$$

$$\frac{1+1}{1+1+1} = 0.67$$

$$\frac{1+2}{1+1+2} = 0.75$$

7.2.3 Nominal Variables

A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green

- *Method 1: Simple matching*
  - *m: # of matches, p: total # of variables*

$$d(i,j) = p - m$$

- *Method 2: use a large number of binary variables*
  - *creating a new binary variable for each of the M nominal states*

Ordinal Variables

An ordinal variable can be

- *discrete or continuous*
- *order is important, e.g., rank*
- *Can be treated like interval-scaled*
  - *replacing xif by their rank*
  - *map the range of each variable onto [0, 1] by replacing i-th object in the f-th variable by*

$$r_{if} \in \{1,...,M_f\}$$

$$z = \frac{r_{if} - 1}{M_f - 1}$$

- *compute the dissimilarity using methods for interval-scaled variables*

Ratio-Scaled Variables

- *Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale,*
  such as AeBt or Ae-Bt
- *Methods:*
  - *treat them like interval-scaled variables — not a good choice! (why?)*
  - *apply logarithmic transformation*
    yif = log(xif)

- *treat them as continuous ordinal data treat their rank as interval-scaled.*

7.2.4 Variables of Mixed Types

- *A database may contain all the six types of variables*
  - *symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio.*
- *One may use a weighted formula to combine their effects.*
  - $$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$
  - –
  - *f is binary or nominal: dij(f) = 0 if xif = xjf , dij(f) = 1 o.w.*

$f = 1$ *ij*

- *f is interval-based: use the normalized distance*
- *f is ordinal or ratio-scaled*
  - *compute ranks rif and and treat zif as interval-scaled*

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

7.3 A Categorization of Major Clustering Methods:

1. **Partitioning algorithms**: Construct various partitions and then evaluate them by some criterion
2. **Hierarchy algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion
3. **Density-based**: based on connectivity and density functions
4. **Grid-based**: based on a multiple-level granularity structure
5. **Model-based**: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

7.4 Partitioning Algorithms: Basic Concept

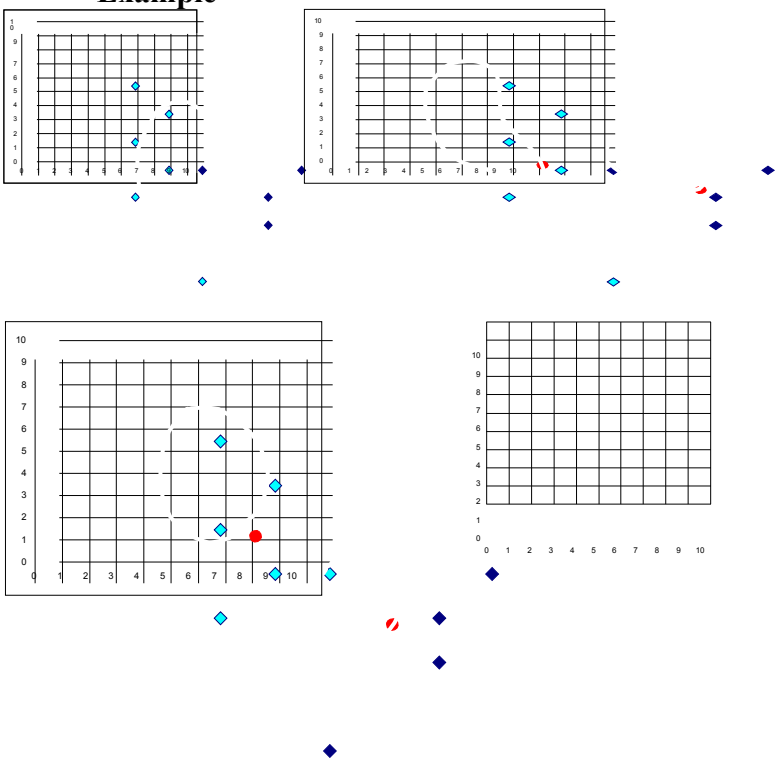
- Partitioning method: Construct a partition of a database *D* of *n* objects into a set of *k* clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

7.4.1 The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in 4 steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  - Assign each object to the cluster with the nearest seed point.
  - Go back to Step 2, stop when no more new assignment.

The *K-Means* Clustering Method

- **Example**



Comments on the *K-Means* Method

- Strength
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)

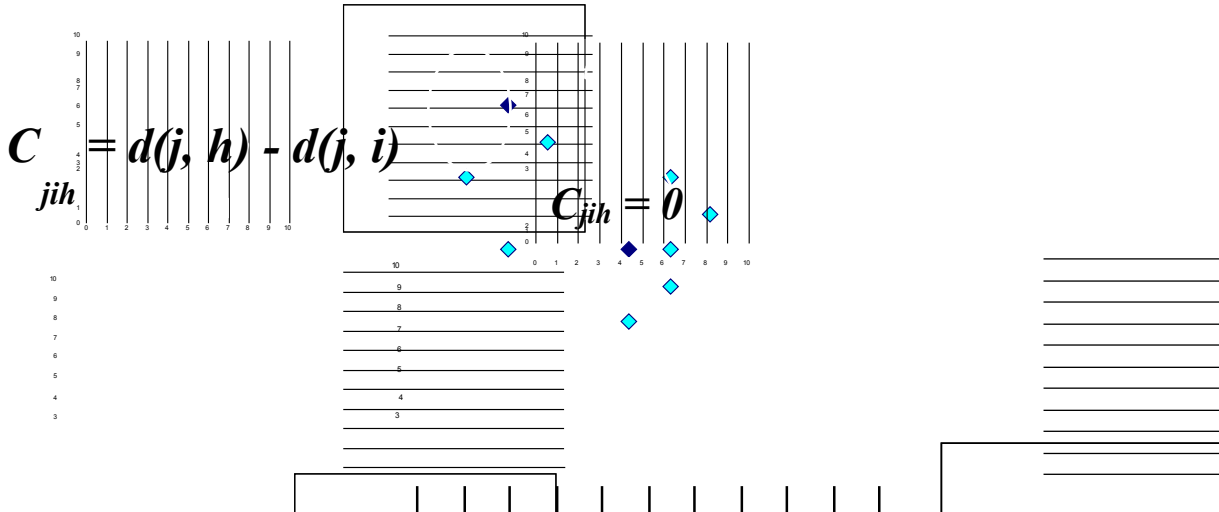
K-MMedoids Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)

7.4.2 PAM (Partitioning Around Medoids) (1987)

- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  - Select ***k*** representative objects arbitrarily
  - For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost ***TCih***
  - For each pair of ***i*** and ***h***,
    - If *TCih* < 0, ***i*** is replaced by ***h***
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

PAM Clustering: Total swapping cost $TCih=\sum_j Cjih$



2
1
2
1

$$C_{jih} = d(j, h) - d(j, t)$$

$C_{jih} = d(j, t) - d(j, i)$

CLARA (Clustering Large Applications) (1990)

- CLARA (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as S+
- It draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than PAM
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS ("Randomized" CLARA) (1994)

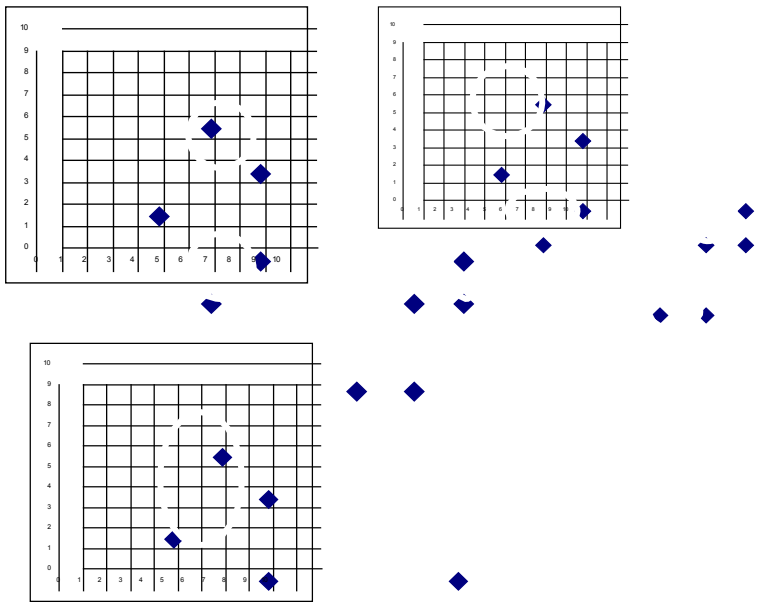
- CLARANS (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
- CLARANS draws sample of neighbors dynamically
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids
- If the local optimum is found, CLARANS starts with new randomly selected node in search for a new local optimum
- It is more efficient and scalable than both PAM and CLARA
- Focusing techniques and spatial access structures may further improve its performance (Ester et al.'95)

7.5 Hierarchical Clustering

Use distance matrix as clustering criteria. This method does not require the number of clusters *k* as an input, but needs a termination condition

7.5.1 AGNES (Agglomerative Nesting)

- *Introduced in Kaufmann and Rousseeuw (1990)*
- *Implemented in statistical analysis packages, e.g., Splus*
- *Use the Single-Link method and the dissimilarity matrix.*
- *Merge nodes that have the least dissimilarity*
- *Go on in a non-descending fashion*
- *Eventually all nodes belong to the same cluster*



A *Dendrogram* Shows How the Clusters are Merged Hierarchically

Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES
- Eventually each node forms a cluster on its own

More on Hierarchical Clustering Methods

- Major weakness of agglomerative clustering methods
  - do not scale well: time complexity of at least $O(n2)$, where n is the number of total objects
  - can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

7.5.2 BIRCH (1996)

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies, by Zhang, Ramakrishnan, Livny (SIGMOD'96)
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
  - Phase 2: use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness:* handles only numeric data, and sensitive to the order of the data record.

Rock Algorithm and CHAMELEON.

- ROCK: Robust Clustering using linKs,
  by S. Guha, R. Rastogi, K. Shim (ICDE'99).
  - Use links to measure similarity/proximity
  - Not distance based
  - Computational complexity:
- Basic ideas:
  - Similarity function and neighbors:

Let $T1 = \{1,2,3\}$, $T2=\{3,4,5\}$

7.5.3 Rock: Algorithm

- Links: The number of common neighbours for the two points.

{1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}

{1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}

{1,2,3} 3 {1,2,4}

- Algorithm
  - Draw random sample
  - Cluster with links
  - Label data in disk

7.5.4 CHAMELEON

- CHAMELEON: hierarchical clustering using dynamic modeling, by G. Karypis, E.H. Han and V. Kumar'99
- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters
- A two phase algorithm
  - 1. Use a graph partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
  - 2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters

AGGLOMERATIVE HIERARCHICAL CLUSTERING

Algorithms of hierarchical cluster analysis are divided into the two categories divisible algorithms and agglomerative algorithms. A *divisible algorithm* starts from the entire set of samples X and divides it into a partition of subsets, then divides each subset into smaller sets, and so on. Thus, a divisible algorithm generates a sequence of partitions that is ordered from a coarser one to a finer one. An *agglomerative algorithm* first regards each object as an initial cluster. The clusters are merged into a coarser partition, and the merging process proceeds until the trivial partition is obtained: all objects are in one large cluster. This process of clustering is a bottom-up process, where partitions from a finer one to a coarser one.

Most agglomerative hierarchical clustering algorithms are variants of the *single-link* or *complete-link* algorithms. In the single-link method, the distance between two clusters is the *minimum* of the distances between all pairs of samples drawn from the two clusters (one element from the first cluster, the other from the second). In the complete-link algorithm, the distance between two clusters is the *maximum* of all distances between all pairs drawn from the two clusters. A graphical illustration of these two distance measures is given.

The basic steps of the agglomerative clustering algorithm are the same. These steps are

1. Place each sample in its own cluster. Construct the list of inter-cluster distances for all distinct unordered pairs of samples, and sort this list in ascending order.
2. Step through the sorted list of distances, forming for each distinct threshold value d_k a graph of the samples where pairs samples closer than d_k are connected into a new cluster by a graph edge. If all the samples are members of a connected graph, stop. Otherwise, repeat this step.
3. The output of the algorithm is a nested hierarchy of graphs, which can be cut at the desired dissimilarity level forming a partition (clusters) identified by simple connected components in the corresponding sub graph.

Let us consider five points $\{x_1, x_2, x_3, x_4, x_5\}$ with the following coordinates as a two-dimensional sample for clustering:

For this example, we selected two-dimensional points because it is easier to graphically represent these points and to trace all the steps in the clustering algorithm.

The distances between these points using the Euclidian measure are

$d(x_1 , x_2) = 2$, $d(x_1, x_3) = 2.5$, $d(x_1, x_4) = 5.39$, $d(x_1, x_5) = 5$

$d(x_1 , x_3) = 1.5$, $d(x_2, x_4) = 5$, $d(x_2, x_5) = 5.29$,

$d(x_3 , x_4) = 3.5$, $d(x_3, x_5) = 4.03$, $d(x_4, x_5) = 2$

The distances between points as clusters in the first iteration are the same for both single-link and complete-link clustering. Further computation for these two algorithms is different. Using agglomerative single-link clustering, the following

steps are performed to create a cluster and to represent the cluster structure as a dendrogram.

Hierarchical and Non-Hierarchical Clustering

There are two main types of clustering techniques, those that create a hierarchy of clusters and those that do not. The hierarchical clustering techniques create a hierarchy of clusters from small to big. The main reason for this is that, as was already stated, clustering is an unsupervised learning technique, and as such, there is no absolutely correct answer. For this reason and depending on the particular application of the clustering, fewer or greater numbers of clusters may be desired. With a hierarchy of clusters defined it is possible to choose the number of clusters that are desired. At the extreme it is possible to have as many clusters as there are records in the database. In this case the records within the cluster are optimally similar to each other (since there is only one) and certainly different from the other clusters. But of course such a clustering technique misses the point in the sense that the idea of clustering is to find useful patters in the database that summarize it and make it easier to understand. Any clustering algorithm that ends up with as many clusters as there are records has not helped the user understand the data any better. Thus one of the main points about clustering is that there be many fewer clusters than there are original records. Exactly how many clusters should be formed is a matter of interpretation. The advantage of hierarchical clustering methods is that they allow the end user to choose from either many clusters or only a few.

The hierarchy of clusters is usually viewed as a tree where the smallest clusters merge together to create the next highest level of clusters and those at that level merge together to create the next highest level of clusters. Figure 1.5 below shows how several clusters might form a hierarchy. When a hierarchy of clusters like this is created the user can determine what the right number of clusters is that adequately summarizes the data while still providing useful information (at the other extreme a single cluster containing all the records is a great summarization but does not contain enough specific information to be useful).

This hierarchy of clusters is created through the algorithm that builds the clusters. There are two main types of hierarchical clustering algorithms:

- Agglomerative - Agglomerative clustering techniques start with as many clusters as there are records where each cluster contains just one record. The clusters that are nearest each other are merged together to form the next

largest cluster. This merging is continued until a hierarchy of clusters is built with just a single cluster containing all the records at the top of the hierarchy.

- Divisive - Divisive clustering techniques take the opposite approach from agglomerative techniques. These techniques start with all the records in one cluster and then try to split that cluster into smaller pieces and then in turn to try to split those smaller pieces.

Of the two the agglomerative techniques are the most commonly used for clustering and have more algorithms developed for them. We'll talk about these in more detail in the next section. The non-hierarchical techniques in general are faster to create from the historical database but require that the user make some decision about the number of clusters desired or the minimum "nearness" required for two records to be within the same cluster. These non-hierarchical techniques often times are run multiple times starting off with some arbitrary or even random clustering and then iteratively improving the clustering by shuffling some records around. Or these techniques sometimes create clusters that are created with only one pass through the database adding records to existing clusters when they exist and creating new clusters when no existing cluster is a good candidate for the given record. Because the definition of which clusters are formed can depend on these initial choices of which starting clusters should be chosen or even how many clusters these techniques can be less repeatable than the hierarchical techniques and can sometimes create either too many or too few clusters because the number of clusters is predetermined by the user not determined solely by the patterns inherent in the database.

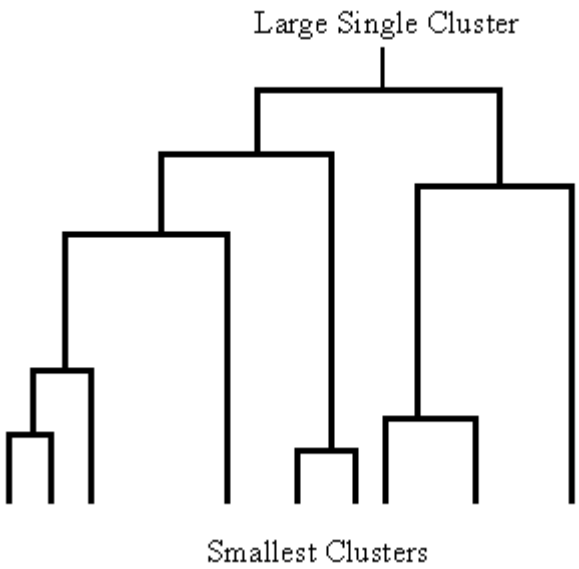


Figure 1.5 ***Diagram showing a hierarchy of clusters. Clusters at the lowest level are merged together to form larger clusters at the next level of the hierarchy.***

Non-Hierarchical Clustering

There are two main non-hierarchical clustering techniques. Both of them are very fast to compute on the database but have some drawbacks. The first are the single pass methods. They derive their name from the fact that the database must only be passed through once in order to create the clusters (i.e. each record is only read from the database once). The other class of techniques are called reallocation methods. They get their name from the movement or "reallocation" of records from one cluster to another in order to create better clusters. The reallocation techniques do use multiple passes through the database but are relatively fast in comparison to the hierarchical techniques.

Hierarchical Clustering

Hierarchical clustering has the advantage over non-hierarchical techniques in that the clusters are defined solely by the data (not by the users predetermining the number of clusters) and that the number of clusters can be increased or decreased by simple moving up and down the hierarchy.

The hierarchy is created by starting either at the top (one cluster that includes all records) and subdividing (divisive clustering) or by starting at the bottom with as many clusters as there are records and merging (agglomerative clustering). Usually the merging and subdividing are done two clusters at a time.

The main distinction between the techniques is their ability to favor long, scraggly clusters that are linked together record by record, or to favor the detection of the more classical, compact or spherical cluster that was shown at the beginning of this section. It may seem strange to want to form these long snaking chain like clusters, but in some cases they are the patters that the user would like to have detected in the database. These are the times when the underlying space looks quite different from the spherical clusters and the clusters that should be formed are not based on the distance from the center of the cluster but instead based on the records being "linked" together. Consider the example shown in Figure 1.6 or in Figure 1.7. In these cases there are two clusters that are not very spherical in shape but could be detected by the single link technique.

When looking at the layout of the data in Figure1.6 there appears to be two relatively flat clusters running parallel to each along the income axis. Neither the complete link nor Ward's method would, however, return these two clusters to the user. These techniques rely on creating a "center" for each cluster and picking these centers so that they average distance of each record from this center is minimized. Points that are very distant from these centers would necessarily fall into a different cluster.

.

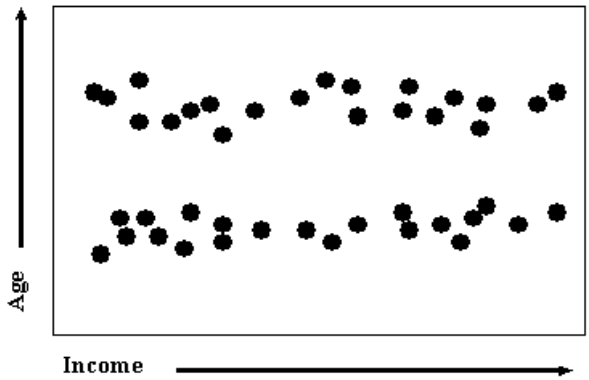


Figure 1.6 ***an example of elongated clusters which would not be recovered by the complete link or Ward's methods but would be by the single-link method.***

7.6 Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

7.6.1 Density-Based Clustering: Background

- Two parameters*:*
  - *Eps*: Maximum radius of the neighbour hood
  - *MinPts*: Minimum number of points in an Eps-neighbour hood of that point
- *NEps(p)*: *{q belongs to D | dist(p,q) <= Eps}*
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* wrt. *Eps*, *MinPts* if
  - 1) *p* belongs to *NEps(q)*
  - 2) core point condition:

  |NEps (q)| >= MinPts

Density-Based Clustering: Background (II)

- Density-reachable:
  - A point *p* is density-reachable from a point *q* wrt. *Eps*, *MinPts* if there is a chain of points *p1*, …, *pn*, *p1 = q*, *pn = p* such that *pi+1* is directly density-reachable from *pi*
- Density-connected
  - A point *p* is density-connected to a point *q* wrt. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* wrt. *Eps* and *MinPts*.

DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise

DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
- If *p* is a core point, a cluster is formed.
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed

7.6.2 OPTICS: A Cluster-Ordering Method (1999)

- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database wrt its density-based clustering structure
  - This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings

- Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
- Can be represented graphically or using visualization techniques

OPTICS: Some Extension from DBSCAN

- Index-based:
  - k = number of dimensions
  - N = 20
  - p = 75%
  - M = N(1-p) = 5
- Complexity: O(*kN2*)
- Core Distance
- Reachability Distance

Max (core-distance (o), d (o, p))
r(p1, o) = 2.8cm. r(p2,o) = 4cm
DENCLUE: using density functions

- DENsity-based CLUstEring by Hinneburg & Keim (KDD'98)
- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (faster than DBSCAN by a factor of up to 45)
  - But needs a large number of parameters

7.6.3 Denclue: Technical Essence

- Uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure.
- Influence function: describes the impact of a data point within its neighborhood.
- Overall density of the data space can be calculated as the sum of the influence function of all data points.
- Clusters can be determined mathematically by identifying density attractors.
- Density attractors are local maximal of the overall density function.

7.7 Grid-Based Methods

Using multi-resolution grid data structure

- Several interesting methods
  - STING (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
  - WaveCluster by Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
    - A multi-resolution clustering approach using wavelet method
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

7.7.1 STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution

STING: A Statistical Information Grid Approach (2)

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
  - *count*, *mean*, *s*, *min*, *max*
  - type of distribution—normal, *uniform*, etc.
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

STING: A Statistical Information Grid Approach (3)

- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- **Advantages:**
  - Query-independent, easy to parallelize, incremental update
  - *O(K),* where *K* is the number of grid cells at the lowest level
- **Disadvantages:**
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

7.7.2 WaveCluster (1998)

- Sheikholeslami, Chatterjee, and Zhang (VLDB'98)

- A multi-resolution clustering approach which applies wavelet transform to the feature space
  - A wavelet transform is a signal processing technique that decomposes a signal into different frequency sub-band.
- Both grid-based and density-based
- Input parameters:
  - # of grid cells for each dimension
  - the wavelet, and the # of applications of wavelet transform.

- **How to apply wavelet transform to find clusters**

  - Summaries the data by imposing a multidimensional grid structure onto data space
  - These multidimensional spatial data objects are represented in a n-dimensional feature space
  - Apply wavelet transform on feature space to find the dense regions in the feature space
  - Apply wavelet transform multiple times which result in clusters at different scales from fine to coarse

- **Why is wavelet transformation useful for clustering**
  - Unsupervised clustering

It uses hat-shape filters to emphasize region where points cluster, but simultaneously to suppress weaker information in their boundary

  - Effective removal of outliers
  - Multi-resolution
  - Cost efficiency
- Major features:
  - Complexity O(N)
  - Detect arbitrary shaped clusters at different scales
  - Not sensitive to noise, not sensitive to input order
  - Only applicable to low dimensional data
  -

7.8Model-Based Clustering Methods:

1.Attempt to optimize the fit between the data and some mathematical model

2. Statistical and AI approach

Conceptual clustering

3. A form of clustering in machine learning

4. Produces a classification scheme for a set of unlabeled objects

5. Finds characteristic description for each concept (class)
COBWEB (Fisher'87)
6. A popular a simple method of incremental conceptual learning
7. Creates a hierarchical clustering in the form of a classification tree
8. Each node refers to a concept and contains a probabilistic description of that concept

Other Model-Based Clustering Methods:

1. Neural network approaches

a. Represent each cluster as an exemplar, acting as a "prototype" of the cluster
b. New objects are distributed to the cluster whose exemplar is the most similar according to some distance measure

2. Competitive learning

a. Involves a hierarchical architecture of several units (neurons)
b. Neurons compete in a "winner-takes-all" fashion for the object currently being presented

7.9 CLIQUE (Clustering In QUEst)

- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98).
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both density-based and grid-based
  - It partitions each dimension into the same number of equal length interval
  - It partitions an m-dimensional data space into non-overlapping rectangular units
  - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - A cluster is a maximal set of connected dense units within a subspace

CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle

- Identify clusters:
  - Determine dense units in all subspaces of interests
  - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determination of minimal cover for each cluster

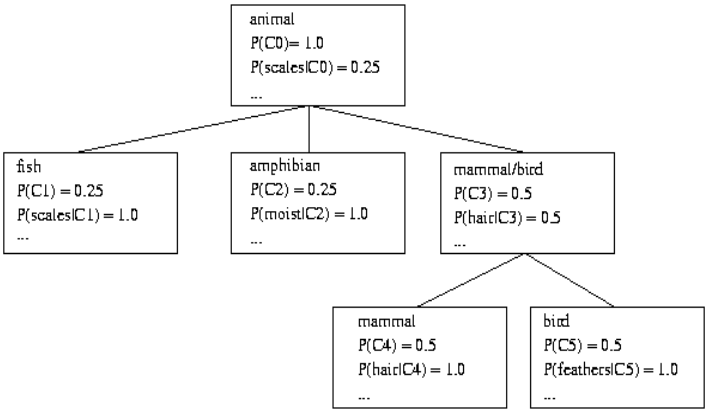
Strength and Weakness of *CLIQUE*

- Strength
  - It *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
  - It is *insensitive* to the order of records in input and does not presume some canonical data distribution
  - It scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
  - The accuracy of the clustering result may be degraded at the expense of simplicity of the method

Model-Based Clustering Methods

- Attempt to optimize the fit between the data and some mathematical model
- Statistical and AI approach
  - Conceptual clustering
    - A form of clustering in machine learning
    - Produces a classification scheme for a set of unlabeled objects
    - Finds characteristic description for each concept (class)
  - COBWEB (Fisher'87)
    - A popular a simple method of incremental conceptual learning
    - Creates a hierarchical clustering in the form of a classification tree
    - Each node refers to a concept and contains a probabilistic description of that concept

COBWEB Clustering Method
A classification tree



More on Statistical-Based Clustering

- Limitations of COBWEB
  - The assumption that the attributes are independent of each other is often too strong because correlation may exist
  - Not suitable for clustering large database data – skewed tree and expensive probability distributions
- CLASSIT
  - an extension of COBWEB for incremental clustering of continuous data
  - suffers similar problems as COBWEB
- AutoClass (Cheeseman and Stutz, 1996)
  - Uses Bayesian statistical analysis to estimate the number of clusters
  - Popular in industry

Other Model-Based Clustering Methods

- Neural network approaches
  - Represent each cluster as an exemplar, acting as a "prototype" of the cluster
  - New objects are distributed to the cluster whose exemplar is the most similar according to some distance measure.
- Competitive learning
  - Involves a hierarchical architecture of several units (neurons)
  - Neurons compete in a "winner-takes-all" fashion for the object currently being presented.

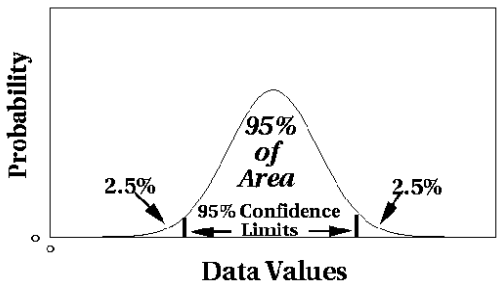
7.11 Outlier Analysis

What Is Outlier Discovery?

- What are outliers?

  - The set of objects are considerably dissimilar from the remainder of the data
  - Example: Sports: Michael Jordon, Wayne Gretzky, ...
- Problem
  - Find top n outlier points
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Customer segmentation
  - Medical analysis

Outlier Discovery: Statistical Approaches



- Assume a model underlying distribution that generates data set (e.g. normal distribution)
- Use discordance tests depending on
  - data distribution
  - distribution parameter (e.g., mean, variance)
  - number of expected outliers
- Drawbacks
  - most tests are for single attribute
  - In many cases, data distribution may not be known

Outlier Discovery: Distance-Based Approach

- Introduced to counter the main limitations imposed by statistical methods
  - We need multi-dimensional analysis without knowing data distribution.
- Distance-based outlier: A DB(p, D)-outlier is an object O in a dataset T such that at least a fraction p of the objects in T lies at a distance greater than D from O
- Algorithms for mining distance-based outliers
  - Index-based algorithm
  - Nested-loop algorithm
  - Cell-based algorithm

Outlier Discovery: Deviation-Based Approach

- Identifies outliers by examining the main characteristics of objects in a group

- Objects that "deviate" from this description are considered outliers
- sequential exception technique
  - simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects
- OLAP data cube technique
  - uses data cubes to identify regions of anomalies in large multidimensional data

Applications of data mining.

- Data mining is a young discipline with wide and diverse applications
  - There is still a nontrivial gap between general principles of data mining and domain-specific, effective data mining tools for particular applications
- Some application domains (covered in this chapter)
  - Biomedical and DNA data analysis
  - Financial data analysis
  - Retail industry
  - Telecommunication industry

Biomedical Data Mining and DNA Analysis

- DNA sequences: 4 basic building blocks (nucleotides): adenine (A), cytosine (C), guanine (G), and thymine (T).
- Gene: a sequence of hundreds of individual nucleotides arranged in a particular order
- Humans have around 100,000 genes
- Tremendous number of ways that the nucleotides can be ordered and sequenced to form distinct genes
- Semantic integration of heterogeneous, distributed genome databases
  - Current: highly distributed, uncontrolled generation and use of a wide variety of DNA data
  - Data cleaning and data integration methods developed in data mining will help

DNA Analysis: Examples

- Similarity search and comparison among DNA sequences
  - Compare the frequently occurring patterns of each class (e.g., diseased and healthy)
  - Identify gene sequence patterns that play roles in various diseases
- Association analysis: identification of co-occurring gene sequences
  - Most diseases are not triggered by a single gene but by a combination of genes acting together

  - Association analysis may help determine the kinds of genes that are likely to co-occur together in target samples
- Path analysis: linking genes to different disease development stages
  - Different genes may become active at different stages of the disease
  - Develop pharmaceutical interventions that target the different stages separately
- Visualization tools and genetic data analysis

Data Mining for Financial Data Analysis

- Financial data collected in banks and financial institutions are often relatively complete, reliable, and of high quality
- Design and construction of data warehouses for multidimensional data analysis and data mining
  - View the debt and revenue changes by month, by region, by sector, and by other factors
  - Access statistical information such as max, min, total, average, trend, etc.
- Loan payment prediction/consumer credit policy analysis
  - feature selection and attribute relevance ranking
  - Loan payment performance
  - Consumer credit rating

Financial Data Mining

- Classification and clustering of customers for targeted marketing
  - multidimensional segmentation by nearest-neighbor, classification, decision trees, etc. to identify customer groups or associate a new customer to an appropriate customer group
- Detection of money laundering and other financial crimes
  - integration of from multiple DBs (e.g., bank transactions, federal/state crime history DBs)
  - Tools: data visualization, linkage analysis, classification, clustering tools, outlier analysis, and sequential pattern analysis tools (find unusual access sequences)

Data Mining for Retail Industry

- Retail industry: huge amounts of data on sales, customer shopping history, etc.
- Applications of retail data mining
  - Identify customer buying behaviors
  - Discover customer shopping patterns and trends
  - Improve the quality of customer service

- Achieve better customer retention and satisfaction
- Enhance goods consumption ratios
- Design more effective goods transportation and distribution policies

Data Mining in Retail Industry: Examples

- Design and construction of data warehouses based on the benefits of data mining
  - Multidimensional analysis of sales, customers, products, time, and region
- Analysis of the effectiveness of sales campaigns
- Customer retention: Analysis of customer loyalty
  - Use customer loyalty card information to register sequences of purchases of particular customers
  - Use sequential pattern mining to investigate changes in customer consumption or loyalty
  - Suggest adjustments on the pricing and variety of goods
- Purchase recommendation and cross-reference of items

Data Mining for Telecomm. Industry

- A rapidly expanding and highly competitive industry and a great demand for data mining
  - Understand the business involved
  - Identify telecommunication patterns
  - Catch fraudulent activities
  - Make better use of resources
  - Improve the quality of service
- Multidimensional analysis of telecommunication data
  - Intrinsically multidimensional: calling-time, duration, location of caller, location of callee, type of call, etc.
- Fraudulent pattern analysis and the identification of unusual patterns
  - Identify potentially fraudulent users and their atypical usage patterns
  - Detect attempts to gain fraudulent entry to customer accounts
  - Discover unusual patterns which may need special attention
- Multidimensional association and sequential pattern analysis
  - Find usage patterns for a set of communication services by customer group, by month, etc.
  - Promote the sales of specific services
  - Improve the availability of particular services in a region
- Use of visualization tools in telecommunication data analysis

UNIT-VI

Mining Stream, Time-Series, and Sequence Data

8.1 Mining Data Streams

Tremendous and Potentially infinite volumes of data streams are often generated by real time surveillance systems, communication networks, Internet traffic, on-line transactions in the financial market or retail industry electric power grids, industry production processes, remote sensors and other dynamic environments.

Unlike traditional data sets, **stream data** flow in and out of a computer system *continuously* and with varying update rates. They are *temporally ordered, fast changing, massive, and potentially infinite*. It may be impossible to store an entire data stream or to scan through it multiple times due to its tremendous volume. More-over, stream data tend to be of a rather low level of abstraction, whereas most analysts are interested in relatively high-level dynamic changes, such as trends and deviations.

- Data Streams
  - Data streams—continuous, ordered, changing, fast, huge amount
  - Traditional DBMS—data stored in finite, persistent data sets
- Characteristics
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive—single scan algorithm (*can only have one look*)
  - Store only the summary of the data seen thus far

Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing.

Stream Data Applications

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing
- Sensor, monitoring & surveillance: video streams, RFIDs
- Security monitoring
- Web logs and Web page click streams

8.1.1 Methodologies for Stream Data Processing

- Major challenges
    - Keep track of a large universe, e.g., pairs of IP address, not ages
- Methodology
    - Synopses (trade-off between accuracy and storage)
    - Use *synopsis data structure*, much smaller ($O(\log_k N)$ space) than their base data set ($O(N)$ space)
    - Compute an *approximate answer* within a *small error range* (factor ε of the actual answer)
- Major methods
    - Random sampling
    - Histograms
    - Sliding windows
    - Multi-resolution model
    - Sketches
    - Radomized algorithms

1) Random sampling

   - *Reservoir sampling*: maintain a set of *s* candidates in the reservoir, which form a true random sample of the element seen so far in the stream. As the data stream flow, every new element has a certain probability (*s*/N) of replacing an old element in the reservoir.

2) Sliding windows

   a. Make decisions based only on *recent data* of sliding window size *w*

   b. An element arriving at time *t* expires at time *t* + *w*

3) Histograms

   a. Approximate the frequency distribution of element values in a stream

   b. Partition data into a set of contiguous buckets

   c. Equal-width (equal value range for buckets) vs. V-optimal (minimizing frequency variance within each bucket)

4) Multi-resolution models

   a. Popular models: balanced binary trees, micro-clusters, and wavelets

5) Sketches

   a. Histograms and wavelets require multi-passes over the data but sketches can operate in a single pass

   b. Frequency moments of a stream A = $\{a_1, \ldots, a_N\}$, F_k: $F_k = \sum_{i=1}^{v} m_i^{\,k}$

   where v: the universe or domain size, m_i: the frequency of i in the sequence

   - Given N elts and v values, sketches can approximate F_0, F_1, F_2 in O(log v + log N) space

6) Randomized algorithms

   a. Monte Carlo algorithm: bound on running time but may not return correct result

$$P(|X - \mu| > k) \leq \frac{\sigma_2}{k_2}$$

b. Chebyshev's inequality:

   i. Let X be a random variable with mean μ and standard deviation σ

c. Chernoff bound: $P[X < (1+\delta)\mu|] < e_{-\mu\delta/4}^{2}$

   i. Let X be the sum of independent Poisson trials X_1, …, X_n, δ in (0, 1]

   ii. The probability decreases exponentially as we move from the mean.

7)

8.1.2 Stream OLAP and Stream Data Cubes

Stream data are generated continuously in a dynamic environment, with huge volume, infinite flow, and fast-changing behavior. It is impossible to store such data streams completly in a dataware house.

Multi-Dimensional Stream Analysis: Examples

- Analysis of Web click streams
  - Raw data at low levels: seconds, web page addresses, user IP addresses, …
  - Analysts want: changes, trends, unusual patterns, at reasonable levels of details
  - E.g., Average clicking traffic in North America on sports in the last 15 minutes is 40% higher than that in the last 24 hours."
- Analysis of power consumption streams
  - Raw data: power consumption flow for every household, every minute

- Patterns one may find: average hourly power consumption surges up 30% for manufacturing companies in Chicago in the last 2 hours today than that of the same day a week ago .

Time Dimension with Compressed Time Scale: Tiltled Frame

In stream data analysis, people are interested in recent changes at a fine scale but in long term changes at a coarse scale. Naturally, we can register time at different levels of granularity. The most recent time is registered at the finest granularity; the more distant time is registered at a coarser granularity; and the level of coarseness depends on the application requirements and on how old the time point is (from the current time). Such a time dimension model is called a **tilted time frame.**

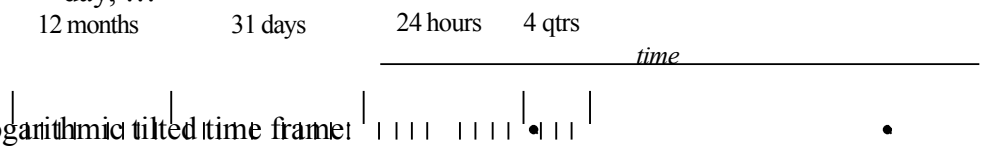
There are many possible ways to design a titled time frame. Here we introduce three models, as illustrated in Figure 8.1: (1) *natural tilted time frame model*, (2) *logarithmic tilted time frame model*, and (3) *progressive logarithmic tilted time frame model*.

- A tilted time frame
  - Different time granularities
    - second, minute, quarter, hour, day, week, …
- Critical layers
  - Minimum interest layer (m-layer)
  - Observation layer (o-layer)
  - User: watches at o-layer and occasionally needs to drill-down down to m-layer
- Partial materialization of stream cubes
  - Full materialization: too space and time consuming
  - No materialization: slow response at query time

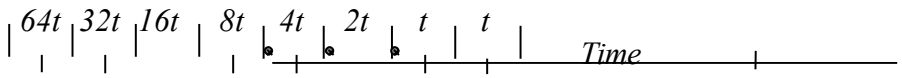
- Partial materialization: what do we mean "partial"

A Titled Time Model

- Natural tilted time frame:
  - Example: Minimal: quarter, then 4 quarters → 1 hour, 24 hours → day, …



- Logarithmic tilted time frame:
  - Example: Minimal: 1 minute, then 1, 2, 4, 8, 16, 32, …



A Titled Time Model

- Pyramidal tilted time frame:
  - Example: Suppose there are 5 frames and each takes maximal 3 snapshots
  - Given a snapshot number N, if N mod $2_d = 0$, insert into the frame number d. If there are more than 3 snapshots, "kick out" the oldest one.

Frame no.	Snapshots (by clock time)
0	69 67 65
1	70 66 62
2	68 60 52
3	56 40 24
4	48 16
5	64 32

Critical Layers

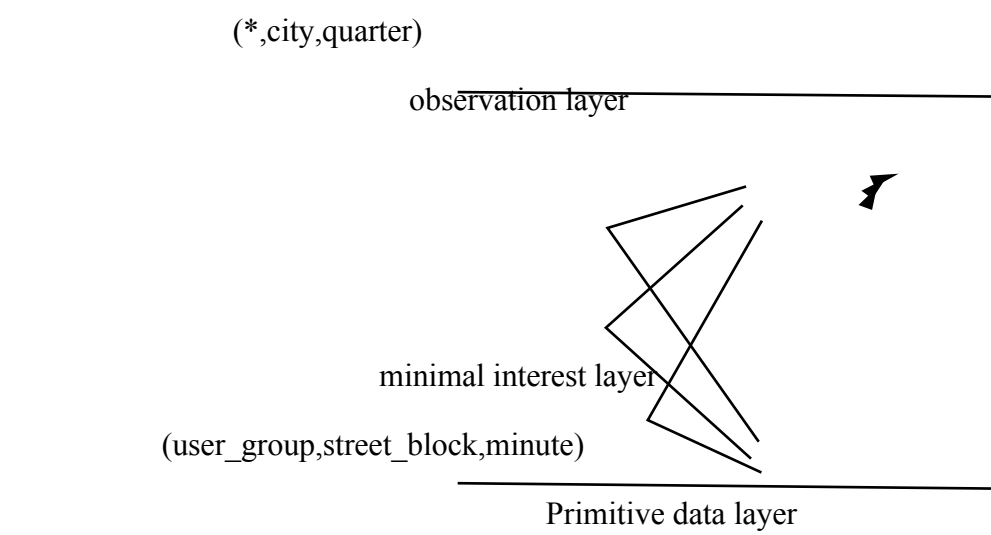


Fig: Two critical layers in a "power supply station" stream data cube

In many applications, it is beneficial to dynamically and incrementally compute and store two critical cuboids (or **layers**), which are determined based on their conceptual and computational importance in stream data analysis. The first layer, called the **minimal interest layer**, is the minimally interesting layer that an analyst would like to study. It is necessary to have such a layer because it is often neither cost effective nor interesting in practice to examine the minute details of stream data. The second layer, called the **observation layer**, is the layer at which an analyst (or an automated system) would like to continuously study the data. This can involve making decisions regarding the signaling of exceptions, or drilling down along certain paths to lower layers to find cells indicating data exceptions.

8.1.3 Frequent-Pattern Mining in Data Streams

- Frequent pattern mining is valuable in stream applications
  - e.g., network intrusion mining (Dokas, et al'02)
- Mining precise freq. patterns in stream data: unrealistic
  - Even store them in a compressed form, such as FPtree
- How to mine frequent patterns with good approximation?
  - Approximate frequent patterns (Manku & Motwani VLDB'02)
  - Keep only current frequent patterns? No changes can be detected
- Mining evolution freq. patterns (C. Giannella, J. Han, X. Yan, P.S. Yu, 2003)
  - Use tilted time window frame
  - Mining evolution and dramatic changes of frequent patterns

Space-saving computation of frequent and top-k elements (Metwally, Agrawal, and El Abbadi, ICDT'05)

Mining Approximate Frequent Patterns

- Mining precise freq. patterns in stream data: unrealistic
  - Even store them in a compressed form, such as FPtree
- Approximate answers are often sufficient (e.g., trend/pattern analysis)
  - Example: a router is interested in all flows:
    - whose frequency is at least 1% (σ) of the entire traffic stream seen so far
    - and feels that 1/10 of σ ($\varepsilon = 0.1\%$) error is comfortable
- How to mine frequent patterns with good approximation?
  - Lossy Counting Algorithm (Manku & Motwani, VLDB'02)
  - Major ideas: not tracing items until it becomes frequent
  - Adv: guaranteed error bound

Disadv: keep a large set of traces

Lossy Counting Algorithm

The Lossy Counting algorithm has three nice properties: (1) there are no false negatives, that is, there is no true frequent item that is not output; (2) false positives are quite "positive" as well, since the output items will have a frequency of at least N N; and (3) the frequency of a frequent item can be underestimated by at most N. For fre-quent items, this underestimation is only a small fraction of its true frequency, so this approximation is acceptable

- Strength
  - A simple idea
  - Can be extended to frequent itemsets
- Weakness:
  - Space Bound is not good
  - For frequent itemsets, they do scan each record many times
  - The output is based on all previous data. But sometimes, we are only interested in recent data

Classification for Dynamic Data Streams

1) Decision tree induction for stream data classification
   - VFDT (Very Fast Decision Tree)/CVFDT (Domingos, Hulten, Spencer, KDD00/KDD01)
2) Is decision-tree good for modeling fast changing data, e.g., stock market analysis?
3) Other stream classification methods
   - Instead of decision-trees, consider other models

- Naïve Bayesian
- Ensemble (Wang, Fan, Yu, Han. KDD'03)
- K-nearest neighbors (Aggarwal, Han, Wang, Yu. KDD'04)

- Tilted time framework, incremental updating, dynamic maintenance, and model construction
- Comparing of models to find changes

Hoeffding Tree

The **Hoeffding tree algorithm** is a decision tree learning method for stream data classi-fication. It was initially used to track Web clickstreams and construct models to predict which Web hosts and Web sites a user is likely to access. It typically runs in sublinear time and produces a nearly identical decision tree to that of traditional batch learners. It uses *Hoeffding trees*, which exploit the idea that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the *Hoeffding bound* (or *additive Chernoff bound*).

- Hoeffding Bound (Additive Chernoff Bound)

r: random variable

R: range of r

n: # independent observations

Mean of r is at least $r_{avg} - \varepsilon$, with probability $1 - d$

$$\varepsilon = \sqrt{\frac{R^2}{\ln(1/\delta)}}$$

- Hoeffding Tree Input

S: sequence of examples

X: attributes

G( ): evaluation function

d: desired accuracy

- Hoeffding Tree Algorithm

1. for each example in S
2. retrieve $G(X_a)$ and $G(X_b)$ //two highest $G(X_i)$
3. if ($G(X_a) - G(X_b) > \varepsilon$)
4. split on X_a
5. recurse to next node
6. break

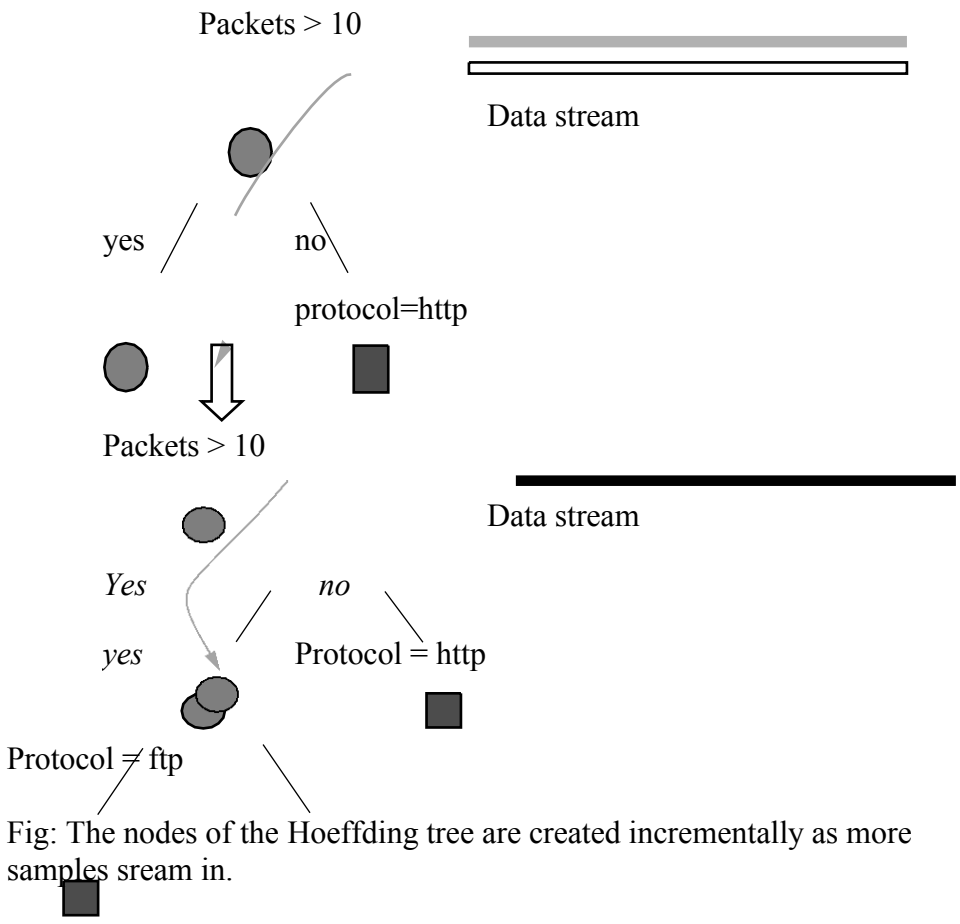


Fig: The nodes of the Hoeffding tree are created incrementally as more samples sream in.

Hoeffding Tree: Strengths and Weaknesses

1) Strengths
    - Scales better than traditional methods
        - Sublinear with sampling
        - Very small memory utilization
    - Incremental
        - Make class predictions in parallel
        - New examples are added as they come

There are, however, weaknesses to the Hoeffding tree algorithm. For example, the algorithm spends a great deal of time with attributes that have nearly identical splitting quality. In addition, the memory utilization can be further optimized. Finally, the algorithm cannot handle concept drift, because once a node is created, it can never change.

VFDT (Very Fast Decision Tree)

The **VFDT (Very Fast Decision Tree) algorithm** makes several modifications to the Hoeffding tree algorithm to improve both speed and memory utilization. The modifications include breaking near-ties during attribute selection more aggressively, computing the *G* function after a number of training examples, deactivating the least promising leaves whenever memory is running low, dropping poor splitting attributes, and improving the initialization method.

- Modifications to Hoeffding Tree
    - Near-ties broken more aggressively
    - G computed every n_{min}
    - Deactivates certain leaves to save memory
    - Poor attributes dropped
    - Initialize with traditional learner (helps learning curve)
- Compare to Hoeffding Tree: Better time and memory

- Compare to traditional decision tree
  - Similar accuracy
  - Better runtime with 1.61 million examples
    - 21 minutes for VFDT
    - 24 hours for C4.5
- Still does not handle concept drift

CVFDT (Concept-adapting VFDT)

Concept-adapting Very Fast Decision Tree algorithm uses a sliding window approach; however itv does not construct a new model from scratch each time.

- Concept Drift
  - Time-changing data streams
  - Incorporate new and eliminate old
- CVFDT
  - Increments count with new example
  - Decrement old example
    - Sliding window
    - Nodes assigned monotonically increasing IDs
  - Grows alternate subtrees
  - When alternate more accurate => replace old
  - O(w) better runtime than VFDT-window

A Classifier Ensemble Approach to Stream Data Classification

There are several reasons for involving more than one classifier. Decision trees are not necessarily the most natural method for han-dling concept drift. Specifically, if an attribute near the root of the tree in CVFDT no longer passes the Hoeffding bound, a large portion of the tree must be regrown. Many other classifiers, such as naïve Bayes, are not subject to this weakness. In addition, naïve Bayesian classifiers also supply relative probabilities along with the class

labels, which expresses the confidence of a decision. Furthermore, CVFDT's automatic elimination of old examples may not be prudent. Rather than keeping only the most up-to-date examples, the ensemble approach discards the least accurate classifiers. Experimentation shows that the ensemble approach achieves greater accuracy than any one of the single classifiers.

8.1.5 Clustering Evolving Data Streams

For effective clustering of stream data, several new methodologies have been devel-oped, as follows:

1) **Compute and store summaries of past data:** Due to limited memory space and fast response requirements, compute summaries of the previously seen data, store the relevant results, and use such summaries to compute important statistics when required.
2) **Apply a divide-and conquer strategy:** Divide data streams into chunks based on order of arrival, compute summaries for these chunks, and then merge the summaries. In this way, larger models can be built out of smaller building blocks.
3) **Incremental clustering of incoming data streams:** Because stream data enter the sys-tem continuously and incrementally, the clusters derived must be incrementally refined.
4) **Perform microclustering as well as macroclustering analysis:** Stream clusters can be computed in two steps: (1) compute and store summaries at the *microcluster* level, where microclusters are formed by applying a hierarchical bottom- up clustering algorithm (Section 7.5.1), and (2) compute *macroclusters* (such as by using another clustering algorithm to group the microclusters) at the user-specified level. This two step computation effectively compresses the data and often results in a sma ller margin of error.
5) **Explore multiple time granularity for the analysis of cluster evolution:** Because the more recent data often play a different role from that of the remote (i.e., ol der) data in stream data analysis, use a tilted time frame model to store snapshots of summarized data at different points in time.
6) **Divide stream clustering into on-line and off-line processes:** While data are streaming in, basic summaries of data snapshots should be computed, stor ed, and incremen-tally updated. Therefore, an on-line process is needed

to maintain such dynamically changing clusters. Meanwhile, a user may pose queries to ask about past, current, or evolving clusters. Such analysis can be performed off-line or as a process independent of on-line cluster maintenance.

CluStream: A Framework for Clustering Evolving Data Streams

CluStream is an algorithm for the clustering of evolving data streams based on user-specified, on-line clustering queries. It divides the clustering process into on-line and off line components.

The on-line component computes and stores summary statistics about the data stream using *microclusters*, and performs incremental on-line computation and maintenance of the microclusters. The off-line component does macro clustering and answers various user questions using the stored summary statistics, which are based on the tilted time frame model.

- Design goal
  - High quality for clustering evolving data streams with greater functionality
  - While keep the stream mining requirement in mind
    - One-pass over the original stream data
    - Limited space usage and high efficiency
- CluStream: A framework for clustering evolving data streams
  - Divide the clustering process into online and offline components
    - Online component: periodically stores summary statistics about the stream data
    - Offline component: answers various user questions based on the stored summary statistic

8.2 Mining Time-Series Data

A **time-series database** consists of sequences of values or events obtained over repeated measurements of time. The values are typically measured at equal time intervals (e.g., hourly, daily, weekly). Time-series databases are popular in many applications, such as stock market analysis, economic and sales fore-casting, budgetary analysis, utility studies, inventory studies, yield projections, work-load projections, process and quality control, observation of natural phenomena (such as atmosphere, temperature, wind, earthquake), scientific and engineering experiments, and medical treatments. A time-series database is also a sequence database. However, a **sequence database** is any database that consists of sequences of ordered events, with or without concrete notions of time. For example, Web page traversal sequences and cus-tomer shopping transaction sequences are sequence data, but they may not be time-series data.

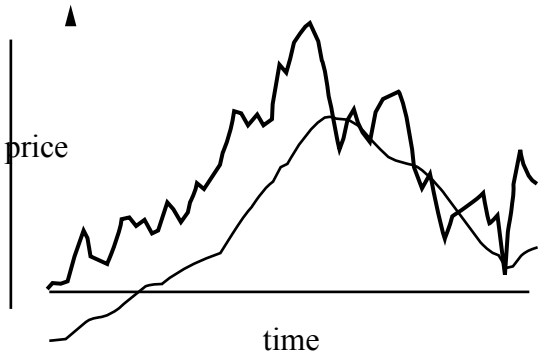
- Time-series database
  - Consists of sequences of values or events changing with time
  - Data is recorded at regular intervals
  - Characteristic time-series components
    - Trend, cycle, seasonal, irregular
- Applications
  - Financial: stock price, inflation
  - Industry: power consumption
  - Scientific: experiment results
  - Meteorological: precipitation

8.2.1 Trend Analysis

A time series involving a variable Y, representing, say, the daily closing price of a share in a stock market, can be viewed as a function of time t, that is, $Y = F(t)$. Such a function can be illustrated as a time-series graph, as shown in Figure 8.4, which describes a point moving with the passage of time.

In general there are two goals in time-series analysis: (1) *modeling time series* (i.e., to gain insight into the mechanisms or underlying forces that generate the time series), and (2) *forecasting time series* (i.e., to predict the future values of the time-series variables).Trend analysis consists of the following four major **components** or **movements** for characterizing time-series data:

1) **Trend or long-term movements**: These indicate the general direction in which a time-series graph is moving over a long interval of time. This movement is displayed by a **trend curve**, or a **trend line**. For example , the trend curve of Figure 8.4 is indicated by a dashed curve. Typical methods for determining a trend curve or trend line include the *weighted moving average method* and the *least squares method*, discussed later.
2) **Cyclic movements or cyclic variations**: These refer to the *cycles*, that is, the long-term oscillations about a trend line or curve, which may or may not be periodic. That is, the cycles need not necessarily follow exactly similar patterns after equal intervals of time.



all electronics stock _

10 day moving average---

-

Figure 8.4 Time-series data of the stock price of *AllElectronics* over time. The *trend* is shown with a dashed curve, calculated by a moving average

3) **Seasonal movements or seasonal variations**: These are systematic or calendar related. Examples include events that recur annually, such as the sudden increase in sales of chocolates and flowers before Valentine's Day or

of department store items before Christmas. The observed increase in water consumption in summer due to warm weather is another example. In these examples, seasonal movements are the identical or nearly identical patterns that a time series appears to follow during corresponding months of successive years.

4) **Irregular or random movements**: These characterize the sporadic motion of time series due to random or chance events, such as labor disputes, floods, or announced personnel changes within companies.

Categories of Time-Series Movements

1) Categories of Time-Series Movements
    - Long-term or trend movements (trend curve): general direction in which a time series is moving over a long interval of time.
    - Cyclic movements or cycle variations: long term oscillations about a trend line or curve.
        - e.g., business cycles, may or may not be periodic
    - Seasonal movements or seasonal variations
        - i.e, almost identical patterns that a time series appears to follow during corresponding months of successive years.
    - Irregular or random movements.
2) Time series analysis: decomposition of a time series into these four basic movements
    - Additive Modal: $TS = T + C + S + I$
    - Multiplicative Modal: $TS = T \times C \times S \times I$

Estimation of Trend Curve

1) The freehand method
    - Fit the curve by looking at the graph

- Costly and barely reliable for large-scaled data mining

2) The least-square method
   - Find the curve minimizing the sum of the squares of the deviation of points on the curve from the corresponding data points
3) The moving-average method
4) Moving average of order n

$$\frac{y_1 + y_2 + \cdots + y_n}{n}, \frac{y_2 + y_3 + \cdots + y_{n+1}}{n}, \frac{y_3 + y_4 + \cdots + y_{n+2}}{n}$$

   - Smoothes the data
   - Eliminates cyclic, seasonal and irregular movements
   - Loses the data at the beginning or end of a series
   - Sensitive to outliers (can be reduced by weighted moving average)

8.2.2 **Similarity Search in Time-Series Analysis**

1) Normal database query finds exact match
2) Similarity search finds data sequences that differ only slightly from the given query sequence
3) Two categories of similarity queries
   - Whole matching: find a sequence that is similar to the query sequence
   - Subsequence matching: find all pairs of similar sequences
4) Typical Applications
   - Financial market
   - Market basket data analysis
   - Scientific databases

- Medical diagnosis

Data Reduction and Transformation Techniques

Due to the tremendous size and high-dimensionality of time-series data, *data reduction* often serves as the first step in time-series analysis. Major strategies for data reduction include *attribute subset selection* (which removes irrelevant or redundant attributes or dimensions), *dimensionality reduction* (which typi-cally employs signal processing techniques to obtain a r educed version of the original data), and *numerosity reduction* (where data are replaced or estimated by alternative, smaller representations, such as histograms, clustering, and sampling). Because time series can be viewed as data of very high dimensionality where each point of time can be viewed as a dimension, dimensionality reduction is our major concern here.

1) Many techniques for signal analysis require the data to be in the frequency domain.

2) Usually data-independent transformations are used
   - The transformation matrix is determined a priori
     - discrete Fourier transform (DFT)
     - discrete wavelet transform (DWT)
3) The distance between two signals in the time domain is the same as their Euclidean distance in the frequency domain

Several dimensionality reduction techniques can be used in time-series analysis. Examples include (1) the *discrete Fourier transform* (*DFT*) as the classical data reduction technique, (2) more recently developed *discrete wavelet transforms (DWT)*, (3) Singu-lar Value Decomposition (SVD) based on Principle Components Analysis (PCA), and (4) random projection-based sketch techniques which can also give a good-quality synopsis of data.

Many techniques for signal analysis require the data to be in the *frequency domain*. Therefore, **distance-preserving orthonormal transformations** are often used to trans-form the data from the time domain to the frequency domain.

Indexing Methods for Similarity Search

For efficient accessing, a *multidimensional index* can be constructed using the first few Fourier coefficients. When a similarity query is submitted to the system, the index can be used to retrieve the sequences that are at most a certain small distance away from the query sequence. Postprocessing is then performed by computing the actual distance between sequences in the time domain and discarding any false matches.

For subsequence matching, each sequence can be broken down into a set of "pieces" of *windows* with length *w*. In one approach, the features of the subsequence inside each window are then extracted. Each sequence is mapped to a "trail" in the feature space. The trail of each sequence is divided into "subtrails," each represented by a minimum bounding rectangle. A multipiece assembly algorithm can then be used to search for longer sequence matches.

Similarity Search Methods

For similarity analysis of time-series data, Euclidean distance is typically used as a similarity measure. Here, the smaller the distance between two sets of time-series data, the more similar are the two series. However, we cannot directly apply the Euclidean distance. Instead, we need to consider differences in the *baseline* and *scale* (or amplitude) of our two series. For example, one stock's value may have a baseline of around $20 and fluctuate with a relatively large amplitude (such as between $15 and $25), while another could have a baseline of around $100 and fluctuate with a relatively small amplitude (such as between $90 and $110). The distance from one baseline to another is referred to as the **offset**.

Steps for Performing a Similarity Search

1) Atomic matching
    - Find all pairs of gap-free windows of a small length that are similar
2) Window stitching
    - Stitch similar windows to form pairs of large similar subsequences allowing gaps between atomic matches
3) Subsequence Ordering
    - Linearly order the subsequence matches to determine whether enough similar pieces exist.

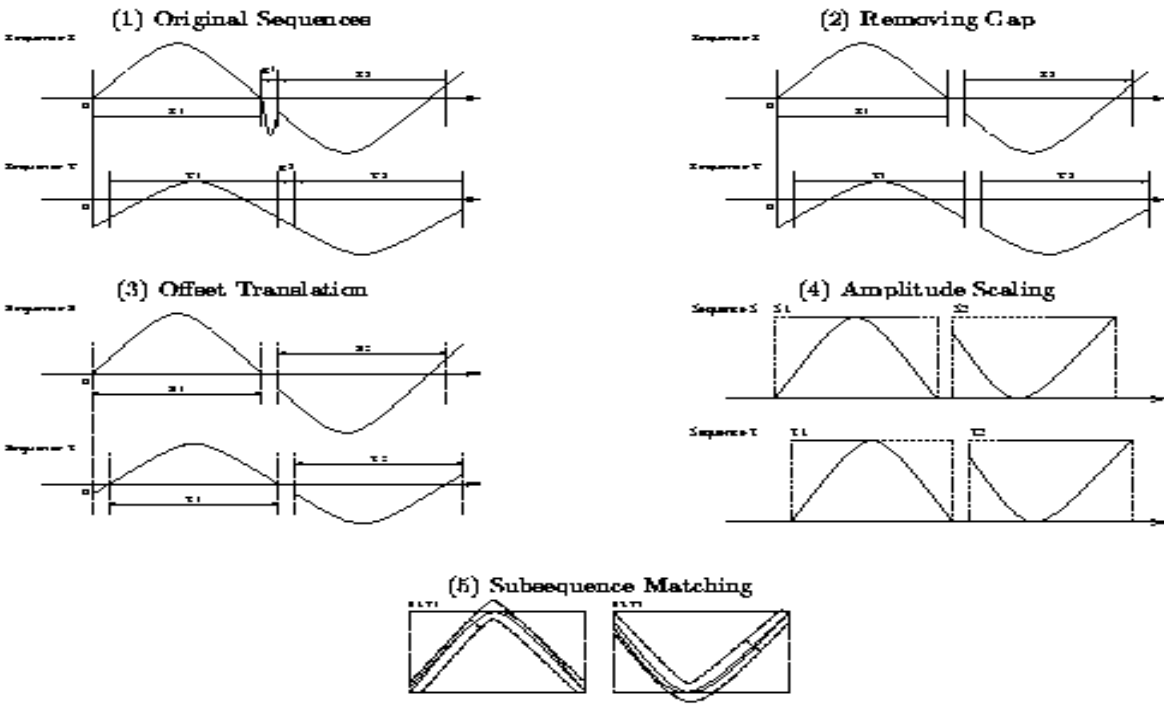


Figure 8.5 Subsequence matching in time-series data:

Query Languages for Time Sequences

1) Time-sequence query language

   - Should be able to specify sophisticated queries like

Find all of the sequences that are similar to some sequence in class *A*, but not similar to any sequence in class *B*.

   - Should be able to support various kinds of queries: range queries, all-pair queries, and nearest neighbor queries.

2) Shape definition language

   - Allows users to define and query the overall shape of time sequences

- Uses human readable series of sequence transitions or macros
- Ignores the specific details
    - E.g., the pattern up, Up, UP can be used to describe increasing degrees of rising slopes

8.3 Mining Sequence Patterns in Transactional Databases

A **sequence database** consists of sequences of ordered elements or events, recorded with or without a concrete notion of time. There are many applications involving sequence data. Typical examples include customer shopping sequences, Web clickstreams, bio-logical sequences, sequences of events in science and engineering, and in natural and social developments.

8.3.1 Sequential Pattern Mining: Concepts and Primitives

Sequential pattern mining is the mining of frequently occurring ordered events or subsequences as patterns. An example of a sequential pattern is "*Customers who buy a Canon digital camera are likely to buy an HP color printer within a month*." For retail data, sequential patterns are useful for shelf placement and promotions. This industry, as well as telecommunications and other businesses, may also use sequential patterns for targeted marketing, customer retention, and many other tasks. Other areas in which sequential patterns can be applied include Web access pat-tern analysis, weather prediction, production processes, and network intrusion detection.

Table 8.1: A Sequence database

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Given support threshold min_sup=2,<(ab)c> is a sequential pattern.

8.3.2 Scalable Methods for Mining Sequential Patterns

Sequential pattern mining is computationally challenging because such mining may generate and/or test a combinatorially explosive number of intermediate subsequences.

"*How can we develop efficient and scalable methods for sequential pattern mining?*" Recent developments have made progress in two directions: (1) efficient methods for mining the *full set* of sequential patterns, and (2) efficient methods for mining only the *set of closed* sequential patterns, where a sequential pattern *s* is **closed** if there exists no sequential pattern *s*0 where *s*0 is a proper supersequence of *s*, and *s*0 has the same (frequency) support as *s*.6

Three such approaches for sequential pattern mining, represented by the algorithms GSP, SPADE, and PrefixSpan, respectively. GSP adopts a *candidate generate-and-test* approach using *horizonal data format* (where the data are represented as h*sequence ID* : *sequence of itemsets*i, as usual, where each itemset is an event). SPADE adopts a candidate generate-and-test approach using *vertical data format* (where the data are represented as h*itemset* : (*sequence ID*, *event ID*)i). The vertical data format can be obtained by transforming from a horizontally formatted sequence database in just one scan. PrefixSpan is a *pat-tern growth* method, which does not require candidate generation.

All three approaches either directly or indirectly explore the **Apriori property**, stated as follows: *every nonempty subsequence of a sequential pattern is a sequential pattern*. (Recall that for a pattern to be called sequential, it must be frequent. That is, it must sat-isfy minimum support.) The Apriori property is antimonotonic (or downward-closed) in that, if a sequence cannot pass a test (e.g., regarding minimum support), all of its supersequences will also fail the test. Use of this property to prune the search space can help make the discovery of sequential patterns more efficient.

GSP: A Sequential Pattern Mining Algorithm Based on Candidate Generate-and-Test

GSP (Generalized Sequential Pattern) is an extension of their seminal algorithm for frequent itemset mining, known as Apriori. GSP uses the downward- closure property of sequential patterns and adopts a multiple-pass, candidate generate-and-test approach.

1) GSP (Generalized Sequential Pattern) mining algorithm

- proposed by Agrawal and Srikant, EDBT'96

2) Outline of the method
   - Initially, every item in DB is a candidate of length-1
   - for each level (i.e., sequences of length-k) do
     - scan database to collect support count for each candidate sequence
     - generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
   - repeat until no frequent sequence or no candidate can be found
3) Major strength: Candidate pruning by Apriori.

Examine GSP using an example

Initial candidates: all singleton sequences

- <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>

Scan database once, count support for candidates

SPADE: An Apriori-Based Vertical Data Format Sequential Pattern Mining Algorithm

SPADE (Sequential Pattern Discovery using Equivalent Class) developed by Zaki 2001.The Apriori like sequential pattern mining approach (based on candidate generate-and-test) can also be explored by mapping a sequence database into vertical data format. In **vertical data format**, the database becomes a set of tuples of the form *itemset* : (*sequence ID*, *event ID*). The **event identifier** serves as a timestamp within a sequence. The *event ID* of the *i*th itemset (or event) in a sequence is *i*. Note than an itemset can occur in more than one sequence. The set of (*sequence ID*, *event ID*) pairs for a given itemset forms the **ID list** of the itemset.

The mapping from horizontal to vertical format requires one scan of the database. A major advantage of using this format is that we can determine the support of any *k*-sequence by simply joining the ID lists of any two of its (*k* 1)-length subsequences. The length of the resulting ID list (i.e., unique *sequence ID* values) is equal to the support of the *k*-sequence, which tells us

whether the sequence is frequent.

PrefixSpan: Prefix-Projected Sequential Pattern Growth

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a

SID	EID
1	1
1	2
1	3
2	1
2	4
3	2
4	3

b

SID
1
2
3
3
4

ab

SID	EID (a)	EID(b
1	1	2
2	1	3
3	2	5
4	3	5

aba

SID	EID (a)	EID(b
1	1	2
2	1	3

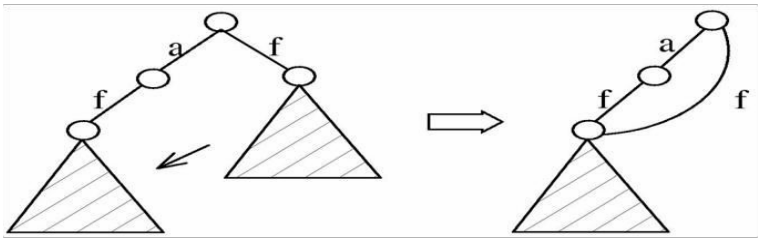
.

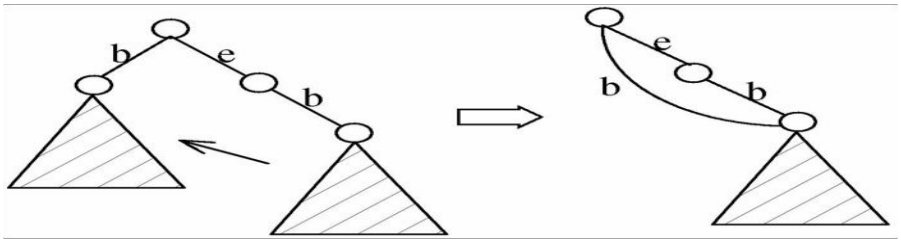
Pattern growth is a method of frequent-pattern mining that does not require candi-date generation. The technique originated in the FP-growth algorithm for

transaction databases. *Pattern growth* is a method of frequent-pattern mining that does not require candidate generation.The general idea of this approach is as follows: it finds the frequent single items, then compresses this information into a *frequent-pattern tree*, or *FP-tree*. The FP-tree is used to generate a set of projected databases, each associated with one frequent item. Each of these databases is mined separately. The algorithm builds prefix patterns, which it concatenates with suffix patterns to find frequent pat-terns, avoiding candidate generation. Here, we look at **PrefixSpan**, which extends the pattern-growth approach to instead mine sequential patterns.

Mining Closed Sequential Patterns

- A closed sequential pattern *s*: there exists no superpattern *s'* such that *s'* כ *s*, and *s'* and *s* have the same support.
- Motivation: reduces the number of (redundant) patterns but attains the same expressive power.
- Using Backward Subpattern and Backward Superpattern pruning to prune redundant search space.





CloSpan is an efficient closed sequential pattern mining method. The method is based on a property of sequence databases, called **equivalence of projected databases**, stated as follows: *Two projected sequence databases, S*j = *S*j ,v*(i*:*e*:; *is a subsequence of), are equivalent if and only if the total number of items in S*j *is equal to the total number of items in S*j .

Mining Multidimensional, Multilevel Sequential Patterns

Sequence identifiers (representing individual customers, for example) and sequence items (such as products bought) are often associated with additional pieces of information. Sequential pattern mining should take advantage of such additional information to discover interesting patterns in multidimensional, multilevel information space. Take customer shopping transactions, for instance. In a sequence database for such data, the additional information associated with sequence IDs could include customer age, address, group, and profession. Information associated with items could include item category, brand, model type, model number, place manufactured, and manufacture date. Mining *multidimensional, multilevel* sequential patterns is the discovery of interesting patterns in such a broad dimensional space, at different levels of detail.

8.3.3 Constraint-Based Mining of Sequential Patterns

constraint-based mining, which incorporates user specified constraints to reduce the search space and derive only patterns that are of interest to the user.Constraints can be expressed in many forms. They may specify desired relation-ships between attributes, attribute values, or aggregates within the resulting patterns mined. Regular expressions can also be used as constraints in the form of "pattern templates," which specify the desired form of the patterns to be mined. The key idea to note is that these kinds of constraints can be used *during* the mining process to confine the search space, thereby improving (1) the efficiency of the mining and (2) the interestingness of the resulting patterns found. This idea is also referred to as *"pushing the constraints deep into the mining process."*

1) Constraint-based sequential pattern mining
   - Constraints: User-specified, for focused mining of desired patterns
   - How to explore efficient mining with constraints? — Optimization
2) Classification of constraints
   - Anti-monotone: E.g., value_sum(S) < 150, min(S) > 10
   - Monotone: E.g., count (S) > 5, $S \supseteq$ {PC, digital_camera}
   - Succinct: E.g., length(S) $\geq$ 10, $S \in$ {Pentium, MS/Office, MS/Money}
   - Convertible: E.g., value_avg(S) < 25, profit_sum (S) > 160, max(S)/avg(S) < 2, median(S) – min(S) > 5
   - Inconvertible: E.g., avg(S) – median(S) = 0

8.3.4 Periodicity Analysis for Time-Related Sequence Data

Periodicity analysis is the mining of periodic patterns, that is, the search for recurring patterns in time-related sequence data. Periodicity analysis can be applied to many important areas. For example, seasons, tides, planet trajectories, daily power consumptions, daily traffic patterns, and weekly TV programs all present certain periodic patterns. Periodicity analysis is often performed over time-series data, which consists of sequences of values or events typically measured at equal time intervals (e.g., hourly, daily, weekly). It can also be applied to other time-related sequence data where the value or event may occur at a nonequal time interval or at any time (e.g., on-line transactions). Moreover, the items to be analyzed can be numerical data, such as daily temperature or power consumption fluctuations, or categorical data (events), such as purchasing a product or watching a game.

The problem of mining periodic patterns can be viewed from different perspectives. Based on the coverage of the pattern, we can categorize periodic patterns into *full* versus *partial* periodic patterns:

1) Full periodicity
   - Every point in time contributes (precisely or approximately) to the periodicity
2) Partial periodicit: A more general notion
   - Only some segments contribute to the periodicity
     - Jim reads NY Times 7:00-7:30 am every week day
3) Cyclic association rules
   - Associations which form cycles
4) Methods
   - Full periodicity: FFT, other statistical analysis methods.
   - Partial and cyclic periodicity: Variations of Apriori-like mining methods.

8.4 Mining Sequence Patterns in Biological Data

Bioinformatics is a promising young field that applies computer technology in molecular biology and develops algorithms and methods to manage and analyze biological data. Because DNA and protein sequences are essential biological

data and exist in huge volumes as well, it is important to develop effective methods to compare and align biological sequences and discover biosequence patterns.

In the case of DNA, these components or "building blocks" are four **nucleotides** (also called *bases*), namely adenine (A), cytosine (C), guanine (G), and thymine (T). In the case of proteins, the components are 20 **amino acids**, denoted by 20 different letters of the alphabet. A gene is a sequence of typically hundreds of individual nucleotides arranged in a particular order. A **genome** is the complete set of genes of an organism. When proteins are needed, the corresponding genes are transcribed into RNA. RNA is a chain of nucleotides. DNA directs the synthesis of a variety of RNA molecules, each with a unique role in cellular function.

An **alignment** is the process of lining up sequences to achieve a maximal level of identity, which also expresses the degree of similarity between sequences. Two sequences are **homologous** if they share a common ancestor. The degree of similarity obtained by sequence alignment can be useful in determining the possibility of homology between two sequences. Such an alignment also helps determine the relative Positions of multiple species in an evolution tree, which is called a **phylogenetic tree**.

8.4.1 Alignment of Biological Sequences

The problem of alignment of biological sequences can be described as follows: *Given two or more input biological sequences, identify similar sequences with long conserved sub-sequences*. If the number of sequences to be aligned is exactly two, it is called **pairwise sequence alignment**; otherwise, it is **multiple sequence alignment**. The sequences to be compared and aligned can be either nucleotides (DNA/RNA) or amino acids (proteins). For nucleotides, two symbols align if they are identical. However, for amino acids, two symbols align if they are identical, or if one can be derived from the other by substitutions that are likely to occur in nature. There are two kinds of alignments: *local alignments* versus *global alignments*. The former means that only portions of the sequences are aligned, whereas the latter requires alignment over the entire length of the sequences.

Pairwise Alignment

```
HEAGAWGHEE
PAWHEAE
```

```
HEAGAWGHE-E
  |   |  ||  |
P-A--W-HEAE
```

```
HEAGAWGHE-E
    ||  ||  |
--P-AW-HEAE
```

1) Which one is better? → Scoring alignments
2) To compare two sequence alignments, calculate a score
   - PAM (Percent Accepted Mutation) or BLOSUM (Blocks Substitution Matrix) (*substitution*) matrices: Calculate matches and mismatches, considering amino acid substitution
   - Gap penalty: Initiating a gap
   - Gap extension penalty: Extending a gap

	A	E	G	H	W
A	5	-1	0	-2	-3
E	-1	6	-3	0	-3
H	-2	0	-2	10	-3
P	-1	-1	-2	-2	-4
W	-3	-3	-3	-3	15

Gap penalty:-8
Gap extension:-8

```
HEAGAWGHE-E
  ||   ||  |
```

```
--P-AW-HEAE

(-8) + (-8) + (-1) + 5 + 15 + (-8)
     + 10 + 6 + (-8) + 6 = 9

HEAGAWGHE-E

P-A--W-HEAE
```

The BLAST Local Alignment Algorithm

The **BLAST** algorithm was first developed by Altschul, Gish, Miller, et al. around 1990 at the National Center for Biotechnology Information (NCBI). The software, its tutori-als, and a wealth of other information can be accessed at *www.ncbi.nlm.nih.gov/BLAST/*. BLAST finds regions of local similarity between biosequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical signif-icance of matches. BLAST can be used to infer functional and evolutionary relationships between sequences as well as to help identify members of gene families.

1) Approach (BLAST) (Altschul et al. 1990, developed by NCBI)
    - View sequences as sequences of short words (*k*-tuple)
        - DNA: 11 bases, protein: 3 amino acids.
    - Create hash table of neighborhood (closely-matching) words.
    - Use statistics to set threshold for "closeness".
    - Start from exact matches to neighborhood words.
2) Motivation
    - Good alignments should contain many close matches.
    - Statistics can determine which matches are significant.
        - Much more sensitive than % identity.
    - Hashing can find matches in O(n) time.
    - Extending matches in both directions finds alignment.
        - Yields high-scoring/maximum segment pairs (HSP/MSP).

Multiple Sequence Alignment Methods

Multiple sequence alignment is usually performed on a set of sequences of amino acids that are believed to have similar structures. The goal is to find common patterns that are conserved among all the sequences being considered.

The alignment of multiple sequences has many applications. First, such an alignment may assist in the identification of highly conserved residues (amino acids), which are likely to be essential sites for structure and function. This will guide or help pairwise alignment as well. Second, it will help build gene or protein families using conserved regions, forming a basis for phylogenetic analysis (i.e., the inference of evolutionary relationships between genes). Third, conserved regions can be used to develop primers for amplifying DNA sequences and probes for DNA microarray analysis.

There are two major approaches for approximate multiple sequence alignment. The first method reduces a multiple alignment to a series of pairwise alignments and then combines the result. The popular **Feng-Doolittle alignment** method belongs to this approach. Feng-Doolittle alignment first computes all of the possible pairwise align-ments by dynamic programming and converts or normalizes alignment scores to distances. It then constructs a "guide tree" by clustering and performs progressive alignment based on the guide tree in a bottom-up manner. Following this approach, a multiple alignment tool, Clustal W, and its variants have been developed as software packages for multiple sequence alignments. The software handles a variety of input/output formats and provides displays for visual inspection. The second multiple sequence alignment method uses hidden Markov models (HMMs).

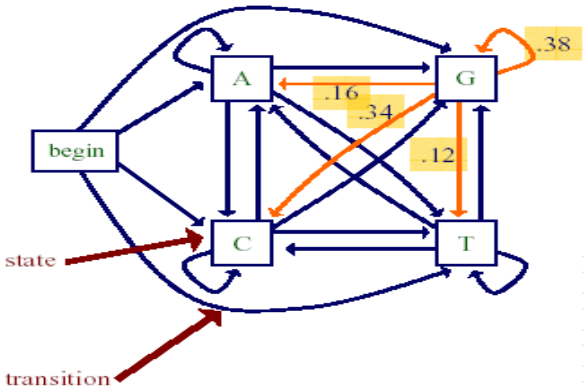
8.4.2 Hidden Markov Model for Biological Sequence Analysis

1) There are many cases in which we would like to represent the statistical regularities of some class of sequences.
   - genes
   - various regulatory sites in DNA (e.g., where RNA polymerase and transcription factors bind)
   - proteins in a given family
2) Markov models are well suited to this type of task.

Markov Chain

A **Markov chain** is a model that generates sequences in which the probability of a sym-
bol depends only on the previous symbol. Figure 8.9 is an example Markov chain model. A Markov chain model is defined by (a) a set of *states*, Q, which emit symbols and (b)

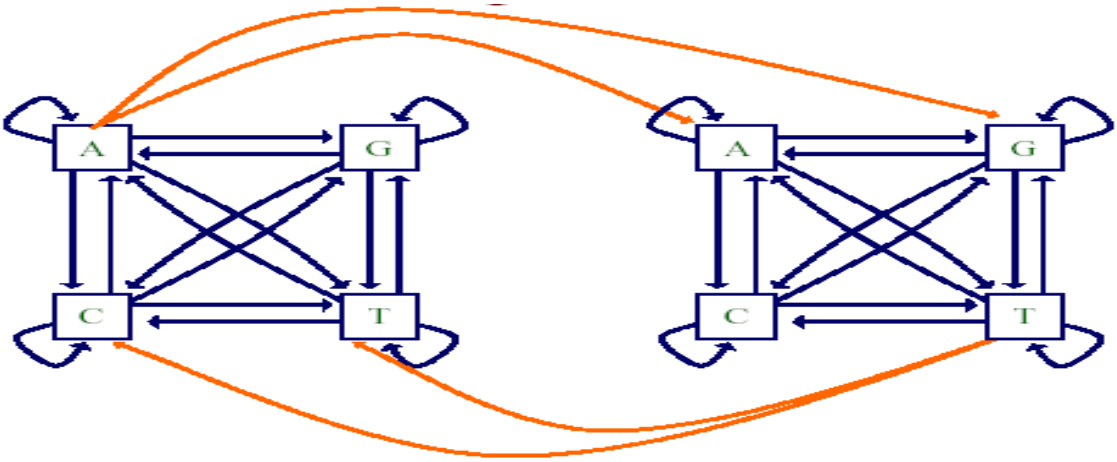
a set of *transitions* between states. States are represented by circles and transitions are rep-resented by arrows. Each transition has an associated **transition probability**, *aij*, which represents the conditional probability of going to state *j* in the next step, given that the current state is *i*. The sum of all transition probabilities from a given state must equal 1, that is, *j*2*Q aij* = 1 for all *j* 2 *Q*. If an arc is not shown, it is assumed to have a 0 prob-ability. The transition probabilities can also be written as a *transition matrix*, *A* = f*aij*g.



Hidden Markov Model

A Hidden Markov Model (HMM) is defined by

1) a set of states, *Q*
2) a set of transitions, where transition probability *akl* = *P*(*i* = *l*j *i* 1 = *k*) is the prob-ability of transitioning from state *k* to state *l* for *k*, *l* 2 *Q*
3) an **emission probability**, *ek*(*b*) = *P*(*xi* = *b*j *i* = *k*), for each state, *k*, and each symbol, *b*, where *ek*(*b*) is the probability of seeing symbol *b* in state *k*. The sum of all emission probabilities at a given state must equal 1, that is, *b* *ek* = 1 for each state, *k*.



Tasks using hidden Markov models include:

Evaluation: Given a sequence, x, determine the probability, $P(x)$, of obtaining x in the model.

Decoding: Given a sequence, determine the most probable path through the model that produced the sequence.

Learning: Given a model and a set of training sequences, find the model parameters (i.e., the transition and emission probabilities) that explain the training sequences with relatively high probability. The goal is to find a model that generalizes well to sequences we have not seen before.

Baum-Welch Algorithm

When the paths for the training sequences are unknown, there is no longer a direct

closed-
form equation for the estimated parameter values. An iterative procedure must be used, like the **Baum-Welch algorithm**. The Baum-Welch algorithm is a special case of the EM algorithm,which is a family of algorithms for learning probabilistic models in problems that involve hidden states.

Graph Mining, Social Network Analysis, and Multirelational Data Mining

9.1 Graph Mining

Graphs become increasingly important in modeling complicated

structures, such as circuits, images, chemical compounds, protein structures, biological networks, social networks, the Web, workflows, and XML documents. Many graph search algorithms have been developed in chemical informatics, computer vision, video indexing, and text retrieval. With the increasing demand on the analysis of large amounts of structured data, graph mining has become an active and important theme in data mining.

9.1.1 Methods for Mining Frequent Subgraphs

We denote the **vertex set** of a graph g by $V(g)$ and the **edge set** by $E(g)$. A label func-tion, L, maps a vertex or an edge to a label. A graph g is a **subgraph** of another graph $g0$ if there exists a subgraph isomorphism from g to $g0$. Given a labeled graph data set, $D = f G1; G2; :::; Gng$, we define $support(g)$ (or $frequency(g)$) as the percentage (or number) of graphs in D where g is a subgraph. A **frequent graph** is a graph whose sup-port is no less than a minimum support threshold, *min sup*.

Fig 9.1 : A sample graph dataset

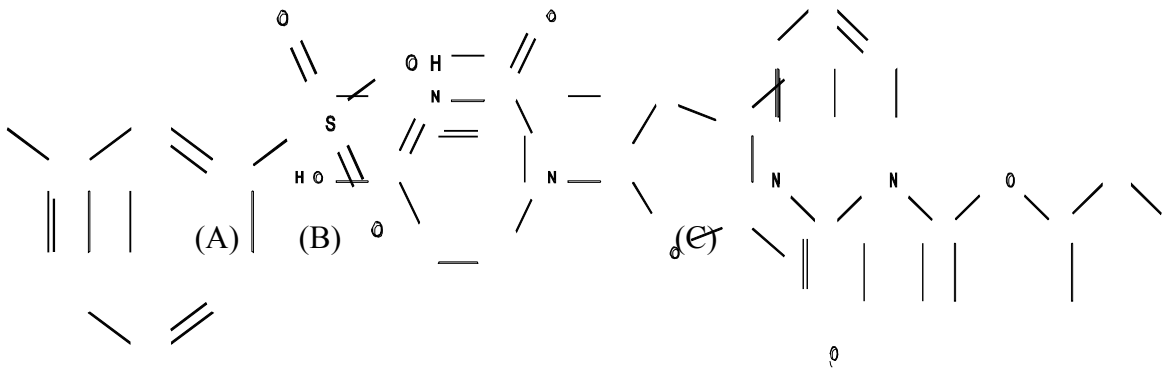
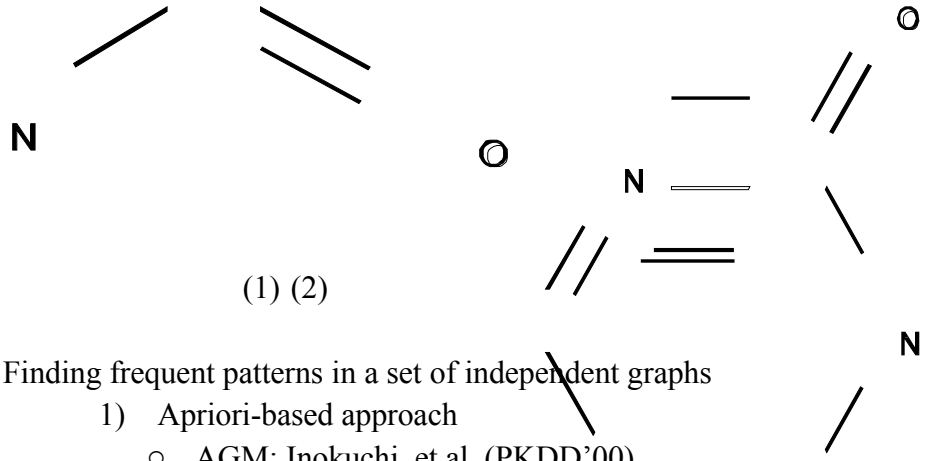


Fig 2: Frequent Patterns
(min support is 2)



(1) (2)

Finding frequent patterns in a set of independent graphs

1) Apriori-based approach
   - AGM: Inokuchi, et al. (PKDD'00)
   - FSG: Kuramochi and Karypis (ICDM'01)
   - PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
   - FFSM: Huan, et al. (ICDM'03)
2) Pattern-growth approach
   - MoFa, Borgelt and Berthold (ICDM'02)
   - gSpan: Yan and Han (ICDM'02)
   - Gaston: Nijssen and Kok (KDD'04)
3) Close pattern mining
   - CLOSEGRAPH: Yan & Han (KDD'03)

Apriori-based Approach

Apriori-based frequent substructure mining algorithms share similar characteristics with Apriori-based frequent itemset mining algorithms. The search for frequent graphs starts with graphs of small "size," and proceeds in a bottom-up manner by generating candidates having an extra vertex, edge, or path. The definition of graph size depends on the algorithm used.

The general framework of Apriori-based methods for frequent substructure mining is outlined in Figure 9.3. We refer to this algorithm as Apriori Graph. *Sk* is the frequent sub-structure set of size *k*. We will clarify the definition of graph size when we describe specific Apriori –based methods further below. Apriori Graph adopts a *level-wise* mining methodology. At each iteration, the size of newly discovered frequent substructures is increased by one. These new substructures are first generated by joining two similar but slightly different frequent sub graphs that were discovered in the previous call to Apriori

Graph. This candidate generation procedure is outlined on line 4. The frequency of the newly formed graphs is then checked. Those found to be frequent are used to generate larger candidates in the next round.

- AGM (Apriori-based Graph Mining), Inokuchi, et al. PKDD'00.

- It generates new graphs with one more node.

AGM: The AGM algorithm uses a *vertex-based candidate generation* method that increases the substructure size by one vertex at each iteration of AprioriGraph. 2 size-k frequent patterns are joined only if they have the same size-(k-1) subgraph. Graph size is the number of vertices in the graph. The newly formed candidate includes the the size-(k-1) subgraph and the additional two vertices from the two patterns. Because it is undertermined whether there is an edge connecting the additional 2 vertices, both of the options are included in the candidate set.

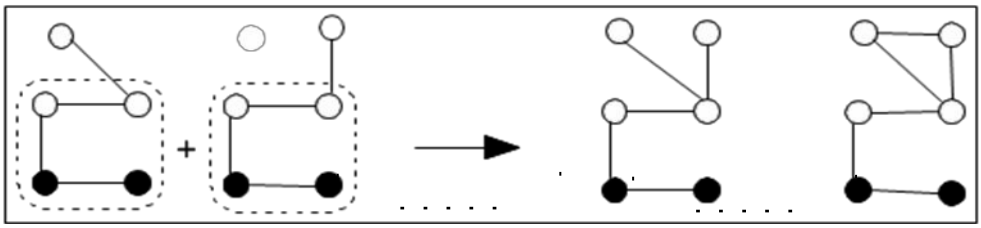


Figure 9.4 AGM: Two substructures joined by two chains.

- FSG (Frquent SubGraph mining), Kuramochi and Karypis, ICDM'01
  - generates new graphs with one more edge

FSG: The FSG algorithm adopts an *edge-based candidate generation* strategy that increases the substructure size by one edge in each call of Apriori Graph. two k-patterns are joined in and only if they share the same sub graph having k-1 edges. The new candidate includes the core and the additional two edges.

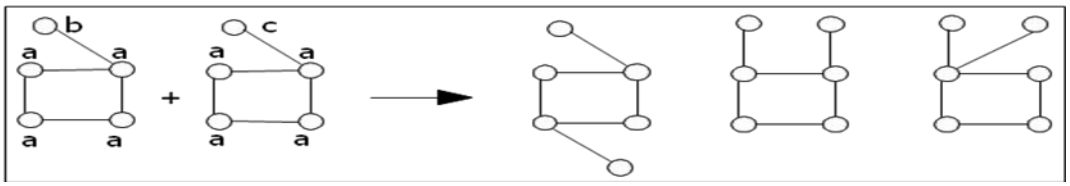


Figure 9.5 FSG: Two substructure patterns and their potential candidates.

Pattern-Growth Approach

Pattern GrowthGraph is simple, but not efficient. The bottleneck is at the inefficiency of extending a graph. The same graph can be discovered many times. For example, there may exist n different (n 1)-edge graphs that can be extended to the same n-edge graph. The repeated discovery of the same graph is computationally inefficient. We call a graph that is discovered a second time a **duplicate graph**. Although line 1 of PatternGrowthGraph gets rid of duplicate graphs, the generation and detection of duplicate graphs may increase the workload.

Aprior based approach uses the breadth-first search strategy. It checks all size-k subgraphs before moving on to size k+1 subgraphs. In contrast, frequent-pattern growth based method can use breadth-based search as well as depth-based search and the depth-first approach consumes less memory.

A simple idea of frequent pattern growth method could be to recursively find frequent patterns by extending a small frequent pattern G until all frequent graphs with G embedded are discovered. However, the same graph can be discovered many times.

9.1.2 Mining Variant and Constrained Substructure Patterns

The frequent subgraph mining discussed in the previous section handles only one spe-cial kind of graphs: *labeled, undirected, connected simple graphs without any specific con-straints*. That is, we assume that the database to be mined contains a set of graphs, each consisting of a set of labeled vertices and labeled but undirected edges, with no other constraints. However, many applications or users may need to enforce various kinds of *constraints* on the patterns to be mined or seek *variant substructure patterns*. For example, we may like to mine patterns, each of which contains certain specificvertices/edges, or where the total number of vertices / edges is within a specified range.

Mining closed Frequent Substructures

The first important variation of a frequent substructure is the **closed frequent substruc-ture**. Take mining frequent subgraphs as an example. As with frequent itemset mining and sequential pattern mining, mining graph patterns may generate an explosive n umber of patterns. This is particularly true for dense data sets, because all of the subgraphs of a frequent graph are also frequent. This is an inherent problem, because according to the Apriori property, all the subgraphs of a frequent substructure must be frequent. A large graph pattern may generate an exponential number of frequent subgraphs. For exam-ple, among 423 confirmed active chemical compounds in an AIDS antiviral screen data set, there are nearly 1 million frequent graph patterns whose support is at least 5%. This renders the further analysis on frequent graphs nearly impossible.

Extension of Pattern-Growth Approach: Mining Alternative Substructure Patterns

A typical pattern-growth graph mining algorithm, such as gSpan or Close Graph, mines labeled, connected, undirected frequent or closed subgraph patterns. Such a graph mining framework can easily be extended for mining *alternative substructure patterns*. Here we discuss a few such alternatives.

First, the method can be extended for **mining unlabeled or partially labeled graphs**. Each vertex and each edge in our previously discussed graphs contain labels. Alternatively, if none of the vertices and edges in a graph are labeled, the graph is **unlabeled**. A graph is **partially labeled** if only some of the edges and/or vertices are labeled. To handle such cases, we can build a label set that contains the original label set and a new empty label. Label is assigned to vertices and edges that do not have labels.

Second, we examine whether gSpan can be extended to **mining nonsimple graphs**. A **nonsimple graph** may have a *self-loop* (i.e., an edge joins a vertex to itself) and *multiple edges* (i.e., several edges connecting two of the same vertices). In gSpan, we always first grow backward edges and then forward edges. In order to accommodate self-loops, the growing order should be changed to *backward edges, self-loops, and forward edges.*

Third, we see how gSpan can be extended to handle **mining directed graphs**. In a directed graph, each edge of the graph has a defined direction. If we use a 5-tuple, $(i; j; li; l(i;j); lj)$, to represent an undirected edge, then for directed edges, a new state is introduced to form a 6-tuple, $(i; j; d; li; l(i;j); lj)$, where d represents the direction of an edge. Let $d = +1$ be the direction from i (vi) to j (vj), whereas $d = 1$ is that from j (vj) to i (vi).

Fourth, the method can also be extended to **mining disconnected graphs**. There are two cases to be considered: (1) the graphs in the data set may be disconnected, and (2) the graph patterns may be disconnected. Finally, if we view a tree as a degenerated graph, it is straightforward to extend the method to **mining frequent subtrees**.

Constraint-Based Mining of Substructure Patterns

Rather than developing many case-specific substructure mining algorithms, it is more appropriate to set up a general framework of constraint-based substructure mining so that systematic strategies can be developed to push constraints deep into the mining process.

1. Element, set, or subgraph containment constraint. Suppose a user requires that the mined patterns contain a particular set of subgraphs. This is a **succinct constraint**, which can be pushed deep into the beginning of the mining process. That is, we can take the given set of sub graphs as a query, perform selection first using the constraint, and then mine on the selected data set by growing (i.e., extending) the patterns from the given set of sub graphs

2. Geometric constraint. A geometric constraint can be that the angle between each pair of connected edges must be within a range, written as "$CG = min\ angle\ \_\ angle(e1; e2; v; v1; v2)\ \_\ max\ angle$," where two edges $e1$ and $e2$ are connected at vertex v with the two vertices at the other ends as $v1$ and $v2$, respectively.

3. **Value-sum constraint**. For example, such a constraint can be that the sum of (positive) weights on the edges, *Sume*, be within a range *low* and *high*. This constraint can be split into two constraints, *Sume* _ *low* and *Sume* _ *high*. The former is a **monotonic constraint**, because once it is satisfied, further "growth" on the graph by adding more edges will always satisfy the constraint. The latter is an **antimonotonic constraint**, because once the condition is not satisfied, further growth of *Sume* will never satisfy

it.

Mining Approximate Frequent Substructures

The principle of minimum description length is adopted in a substructure discovery system called SUBDUE, which mines approximate frequent substructures. It looks for a substructure pattern that can best compress a graph set based on the Minimum Description Length (MDL) principle, which essentially states that the simplest representation is preferred. SUBDUE adopts a constrained beam search method. It grows a single vertex incrementally by expanding a node in it. At each expansion, it searches for the best total description length: the description length of the pattern and the description length of the graph set with all the instances of the pattern condensed into single nodes. SUBDUE performs approximate matching to allow slight variations of substructures, thus supporting the discovery of approximate substructures.

Mining Coherent Substructures

A frequent substructure G is a **coherent subgraph** if the mutual information between G and each of its own subgraphs is above some threshold. The number of coherent substructures is significantly smaller than that of frequent substructures. Thus, mining coherent substructures can efficiently prune redundant patterns (i.e., patterns that are similar to each other and have similar support). A promising method was developed for mining such substructures.

Mining Dense Substructures

In the analysis of graph pattern mining, researchers have found that there exists a specific kind of graph structure, called a **relational graph**, where each node label is used only once per graph. The relational graph is widely used in modeling and analyzing massive networks (e.g., biological networks, social networks, transportation networks, and the World Wide Web). In biological networks, nodes represent objects like genes, proteins, and enzymes, whereas edges encode the relationships, such as control, reaction, and correlation, between these objects.

9.1.3 Applications: Graph Indexing, Similarity Search, Classification, and Clustering

Graph Indexing with Discriminative Frequent Substructures

Indexing is essential for efficient search and query processing in database and infor-mation systems. Technology has evolved from single-dimensional to multidimensional indexing, claiming a broad spectrum of successful applications, including relational database systems and spatiotemporal, time-series, multimedia, text-, and Web-based information systems. However, the traditional indexing approach encounters challenges in databases involving complex objects, like graphs, because a graph may contain an exponential number of subgraphs. It is ineffective to build an index based on vertices or edges, because such features are nonselective and unable to distinguish graphs. On the other hand, building index structures based on subgraphs may lead to an explosive number of index entries.

A method called gIndex was developed to build a compact and effective graph index structure. It takes frequent and discriminative substructures as index features. Frequent substructures are ideal candidates because they explore the shared structures in the data and are relatively stable to database updates.

Substructure Similarity Search in Graph Databases

Bioinformatics and chem-informatics applications involve query-based search in massive, complex structural data. Even with a graph index, such search can encounter challenges because it is often too restrictive to search for an exact match of an index entry.

A feature-based structural filtering algorithm, called Grafil (**Gra**ph Similarity **Fil**ter-ing), was developed to filter graphs efficientl y in large-scale graph databases. Grafil models each query graph as a set of features and transforms the edge deletions into "feature misses" in the query graph. It is shown that using too many features will not leverage the filtering performance. Therefore, a multifilter composition strategy is developed, where each filter uses a distinct and complementary subset of the features. The filters are constructed by a hierarchical, one-dimensional clustering algorithm that groups features with similar selectivity into a *feature set*.

Classification and Cluster Analysis Using Graph Patterns

The discovered frequent graph patterns and/or their variants can be used as features for graph classification. First, we mine frequent graph patterns in the training set. The features that are frequent in one class but rather infrequent in the other class(es) should be considered as highly discriminative features. Such features will then be used for model construction. To achieve high-quality classification, we can adjust

the thresholds on frequency, discriminativeness, and graph connectivity based on the data, the number and quality of the features generated, and the classification accuracy. Various classification approaches, including support vector machines, naïve Bayesian, and associative classification, can be used in such graph-based classification. Similarly, cluster analysis can be explored with mined graph patterns. The set of graphs that share a large set of similar graph patterns should be considered as highly similar and should be grouped into similar clusters. The graphs that do not belong to any cluster or that are far away from the derived clusters can be considered as *outliers*. Thus outliers can be considered as a by-product of cluster analysis.

9.2 Social Network Analysis

The notion of social networks, where relationships between entities are represented as *links* in a graph, has attracted increasing attention in the past decades. Thus social network analysis, from a data mining perspective, is also called *link analysis* or *link mining*.

9.2.1 What Is a Social Network?

From the point of view of data mining, a **social network** is a heterogeneous and multire-lational data set represented by a graph. The graph is typically very large, with **nodes** corresponding to *objects* and **edges** corresponding to *links* representing relationships or interactions between objects. Both nodes and links have *attributes*. Objects may have class labels. Links can be one-directional and are not required to be binary.

Social networks need not be social in context. There are many real-world instances of technological, business, economic, and biologic social networks. Examples include electrical power grids, telephone call graphs, the spread of computer viruses, the World Wide Web, and coauthorship and citation networks of scientists. Customer networks and collaborative filtering problems (where product recommendations are made based on the preferences of other customers) are other examples. In biology, examples range from epidemiological networks, cellular and metabolic networks, and food webs, to the neural network of the nematode worm Caenorhabditis elegans (the only creature whose neural network has been completely mapped).

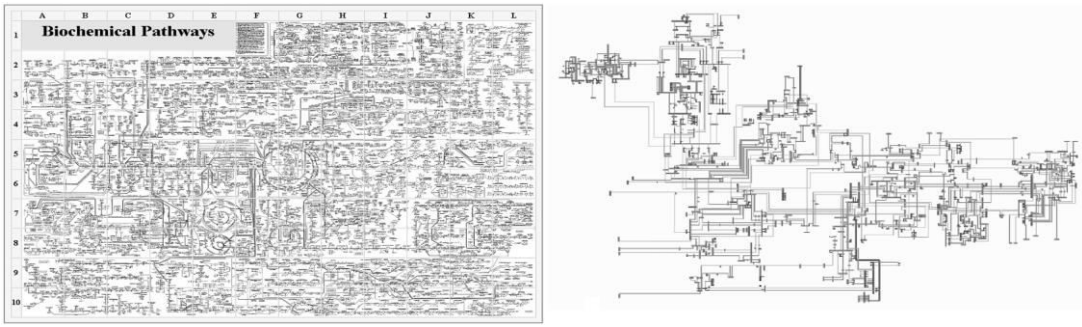


Figure 9.16 Real-world examples of social networks: (a) science coauthor network, (b) connected pages on a part of the Internet, (c) biochemical pathway network, and (d) New York state electric power grid.

9.2.2 Characteristics of Social Networks

Social networks are rarely static. Their graph representations evolve as nodes and edges are added or deleted over time. In general, social networks tend to exhibit the following phenomena:

1. Densification power law: Previously, it was believed that as a network evolves, the number of degrees grows linearly in the number of nodes. This was known as the constant average degree assumption. However, extensive experiments have shown that, on the contrary, networks become increasingly *dense* over time with the average degree increasing (and hence, the number of edges growing super linearly in the number of nodes). The densification follows the **densification power law** (or **growth power law**), which states

$$e(t) \propto n(t)^a; \tag{9.1}$$

where $e(t)$ and $n(t)$, respectively, represent the number of edges and nodes of the graph at time t, and the exponent a generally lies strictly between 1 and 2. Note that if $a = 1$, this corresponds to constant average degree over time, whereas $a = 2$ corresponds to an extremely dense graph where each node has edges to a constant fraction of all nodes.

2.Shrinking diameter: It has been experimentally shown that the effective diameter tends to *decrease* as the network grows. This contradicts an earlier belief that the diameter slowly increases as a function of network size

3.Heavy-tailed out-degree and in-degree distributions: The number of out-degrees for a node tends to follow a heavy-tailed distribution by observing the **power law**, $1=na$, where n is the rank of the node in the order of decreasing out-degrees and typically, $0 < a < 2$ (Figure 9.17).

9.2.3 Link Mining: Tasks and Challenges

Traditional methods of machine learning and data mining, taking, as input, a random sample of homogenous objects from a single relation, may not be appropriate here. The data comprising social networks tend to be heterogeneous, multi relational, and semi-structured. As a result, a new field of research has emerged called **link mining**. Link mining is a confluence of research in social net-works, link analysis, hypertext and Web mining, graph mining, relational learning, and inductive logic programming. It embodies descriptive and predictive modeling. Here, we list these tasks with examples from various domains:

1.Link-based object classification. In traditional classification methods, objects are clas-sified based on the attributes that describe them. Link-based classification predicts the category of an object based not only on its attributes, but also on its links, and on the attributes of linked objects.
Web page classification is a well-recognized example of link-based classification. It predicts the category of a Web page based on word occurrence (words that occur on the page) and *anchor text* (the hyperlink words, that is, the words you click on when you click on a link), both of which serve as attributes.

2.Object type prediction. This predicts the type of an object, based on its attributes and its links, and on the attributes of objects linked to it. In the bibliographic domain, we may want to predict the venue type of a publication as either conference, journal, or workshop.

3. Link type prediction. This predicts the type or purpose of a link, based on properties of the objects involved. Given epidemiological data, for instance, we may try to predict whether two people who know each other are family members, coworkers, or acquaintances.

4. Predicting link existence. Unlike link type prediction, where we know a connection exists between two objects and we want to predict its type, instead we may want to predict whether a link exists between two objects. Examples include predicting whether there will be a link between two Web pages.

5. Link cardinality estimation. There are two forms of link cardinality estimation. First, we may predict the number of links to an object. This is useful, for instance, in predicting the authoritativeness of a Web page based on the number of links to it (in-links). Similarly, the number of out-links can be used to identify Web pages that act as *hubs*, where a hub is one or a set of Web pages that point to many authoritative pages of the same topic

6. Object reconciliation. In object reconciliation, the task is to predict whether two objects are, in fact, the same, based on their attributes and links. This task is common in information extraction, duplication elimination, object consolidation, and citation matching, and is also known as *record linkage* or *identity uncertainty*. Examples include predicting whether two websites are mirrors of each other, whether two citations actually refer to the same paper, and whether two apparent disease strains are really the same.

7. Group detection . Group detection is a clustering task. It predicts when a set of objects belong to the same group or cluster, based on their attributes as well as their link structure. An area of application is the identification of *Web communities*, where a Web community is a collection of Web pages that focus on a particular theme or topic. A similar example in the bibliographic domain is the identification of research communities.

8. Subgraph detection. Subgraph identification finds characteristic subgraphs within networks. An example from biology is the discovery of subgraphs corresponding to protein structures. In chemistry, we can search for subgraphs representing chemical substructures.

9. Metadata mining. Metadata are data about data. Metadata provide semi-

structured data about unstructured data, ranging from text and Web data to multimedia data-bases. It is useful for data integration tasks in many domains.

The implementation of these tasks, however, invokes many challenges. We examine several of these challenges here:

1. Logical versus statistical dependencies. Two types of dependencies reside in the graph—*link structures* (representing the logical relationship between objects) and *probabilistic dependencies* (representing statistical relationships, such as correlation between attributes of objects where, typically, such objects are logically related).

2. Feature construction. In link-based classification, we consider the attributes of an object as well as the attributes of objects linked to it. In addition, the links may also have attributes. The goal of *feature construction* is to construct a single feature rep-resenting these attributes. This can involve feature selection and feature aggregation. In *feature selection*, only the most discriminating features are included.2 *Feature aggregation* takes a multiset of values over the set of related objects and returns a summary of it.

3. Instances versus classes. This alludes to whether the model refers explicitly to indi-viduals or to classes (generic categories) of individuals. An advantage of the former model is that it may be used to connect particular individuals with high probability.

4. Collective classification and collective consolidation. Consider training a model for classification, based on a set of class-labeled objects. Traditional classification methods consider only the attributes of the objects.

5. Effective use of labeled and unlabeled data. A recent strategy in learning is to incorporate a mix of both labeled and unlabeled data. Unlabeled data can help infer the object attribute distribution. Links between unlabeled (test) data allow us to use attributes of linked objects.

6. Link prediction. A challenge in link prediction is that the prior probability of a par-ticular link between objects is typically extremely low.

7. Closed versus open world assumption. Most traditional approaches assume that

we know all the potential entities in the domain. This "closed world" assumption is unrealistic in real-world applications.

8. Community mining from multi relational networks. Typical work on social network analysis includes the discovery of groups of objects that share similar properties. This is known as *community mining*. Web page linkage is an example, where a discovered community may be a set of Web pages on a particular topic.

These challenges will continue to stimulate much research in link mining.

9.2.4 *Mining on Social Networks*

Link Prediction: What Edges Will Be Added to the Network?

Social networks are dynamic. New links appear, indicating new interactions between objects. In the **link prediction problem**, we are given a snapshot of a social network at time t and wish to *predict the edges that will be added to the network during the interval from time t to a given future time, t*0.

Approaches to link prediction have been proposed based on several measures for analyzing the "proximity" of nodes in a network. Many measures originate from tech-niques in graph theory and social network analysis. The general methodology is as follows: All methods assign a connection weight, *score*(***X, Y***), to pairs of nodes, ***X*** and ***Y***, based on the given proximity measure and input graph, G. A ranked list in decreasing order of *score*(***X, Y***) is produced. This gives the predicted new links in decreasing order of confidence. The predictions can be evaluated based on real observations on experimental data sets.

Mining Customer Networks for Viral Marketing

Viral marketing is an application of social network mining that explores how individuals can influence the buying behavior of others. Traditionally, companies have employed **direct marketing** or **mass marketing.** These approaches, however, neglect the influence that customers can have on the purchasing decisions of others. For example, consider a person who decides to see a particular movie and persuades a group of friends to see the same film. **Viral marketing** aims to optimize the positive word-of-mouth effect among customers. It can choose to spend more money marketing to an individual if that person has many social connections. Thus, by considering the interactions between customers, viral marketing may obtain higher profits than traditional marketing, which ignores such interactions.

The growth of the Internet over the past two decades has led to the availability of many social networks that can be mined for the purposes of viral marketing. Examples include e-mail mailing lists, UseNet groups, on-line forums, instant relay chat (IRC), instant messaging, collaborative filtering systems, and knowledge-sharing sites. **Knowledge-sharing sites** (such as Epinions at *www.epinions.com*) allow users to offer advice or rate products to help others, typically for free. Users can rate the usefulness or "trustworthiness" of a review, and may possibly rate other reviewers as well. In this way, a network of trust relationships between users (known as a "web of trust") evolves, representing a social network for mining.

Mining Newsgroups Using Networks

A typical news-group posting consists of one or more quoted lines from another posting

followed by the opinion of the author. Such quoted responses form "quotation links" and create a network in which the vertices represent individuals and the links "responded-to" relationships. An interesting phenomenon is that people more frequently respond to a message when they *disagree* than when they *agree*. This behavior exists in many newsgroups and is in sharp contrast to the Web page link graph, where linkage is an indicator of agreement or common interest. Based on this behavior, one can effectively classify and partition authors in the newsgroup into *opposite camps* by analyzing the graph structure of the responses.

This newsgroup classification process can be performed using a graph-theoretic approach. The *quotation network* (or *graph)* can be constructed by building a quo-tation link between person i and person j if i has quoted from an earlier posting written by j. We can consider any bipartition of the vertices into two sets: F represents those *for* an issue and A represents those *against* it.

Community Mining from Multirelational Networks

With the growth of the Web, *community mining* has attracted increasing attention. A great deal of such work has focused on mining implicit communities of Web pages, of scientific literature from the Web, and of document citations. In principle, a **com-munity** can be defined as a group of objects sharing some common properties. **Com-munity mining** can be thought of as subgraph identification. For example, in Web page linkage, two Web pages (objects) are related if there is a hyperlink between them. A graph of Web page linkages can be mined to identify a community or set of Web pages on a particular topic.

Most techniques for graph mining and community mining are based on a homogenous graph, that is, they assume only one kind of relationship exists between the objects. However, in real social networks, there are always various kinds of relationships between the objects. Each relation can be viewed as a **relation network**. In this sense, the multiple relations form a **multirelational social network** (also referred to as a **heterogeneous social network**). Each kind of relation may play a distinct role in a particular task. Here, the different relation graphs can provide us with different communities.

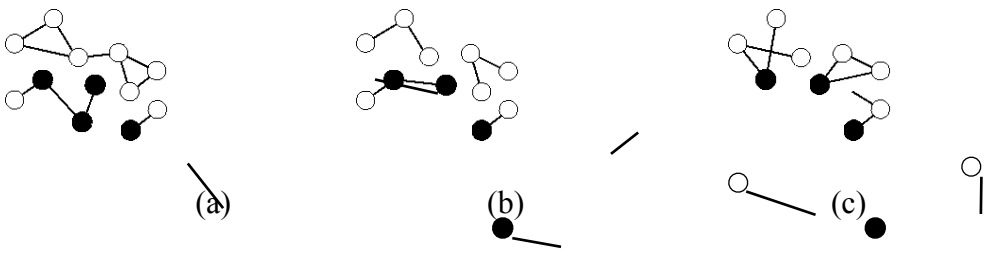


Figure 9.18 There are three relations in the network. The four colored objects are required to belong to the same community, according to a user query.

As an example, consider the network in Figure 9.18, which has three different relations, shown as (a), (b), and (c), respectively. Suppose a user requires that the four colored objects belong to the same community and specifies this with a query. Clearly, the relative importance of each of the three relations differs with respect to the user's information need. Of the three relations, we see that (a) is the most relevant to the user's need and is thus the most important, while (b) comes in second. Relation (c) can be seen as noise in regards to the user's information need. Traditional social network analysis does not distinguish these relations. The different relations are treated equally. They are simply combined together for describing the structure between objects. Unfortunately, in this example, relation (c) has a negative effect for this purpose. However, if we combine these relations according to their importance, relation (c) can be easily excluded, leaving relations (a) and (b) to be used to discover the community structure, which is consistent with the user's requirement.

9.3 Multi Relational Data Mining

Relational databases are the most popular repository for *structured* data. In a relational database, multiple relations are linked together via entity-relationship links. Many classification approaches (such as neural networks and support vector machines) can only be applied to data represented in single, "flat" relational form— that is, they expect data in a single table. However, many real-world applications, such as credit card fraud detection, loan applications, and biological data analysis, involve decision-making processes based on information stored in multiple relations in a relational database. Thus, multirelational data mining has become a field of strategic importance.

9.3.1 What Is Multirelational Data Mining?

Multirelational data mining (MRDM) methods search for patterns that involve multi-ple tables (relations) from a relational database. Consider the multirelational schema of Figure 9.19, which defines a financial database. Each table or relation represents an entity or a relationship, described by a set of attributes. Links between relations show the relation-ship between them. One method to apply traditional data mining methods (which assume that the data reside in a single table) is propositionalization, which converts multiple rela-tional data into a single flat data relation, using joins and aggregations. This, however, could lead to the generation of a huge, undesirable "universal relation" (involving all of the attributes). Furthermore, it can result in the loss of information, including essential semantic information represented by the links in the database design.

Multirelational data mining aims to discover knowledge directly from relational data. There are different multirelational data

mining tasks, including multirelational classi-fication, clustering, and frequent pattern mining. Multirelational classification aims to build a classification model that utilizes information in different relations. Multirelational clustering aims to group tuples into clusters using their own attributes as well as tuples related to them in different relations. Multirelational frequent pattern mining aims at finding patterns

involving interconnected items in different relations.

In a database for multirelational classification, there is one **target relation**, *Rt*, whose tuples are called **target tuples** and are associated with class labels. The other relations are *nontarget relations*. Each relation may have one *primary key* (which uniquely identifies tuples in the relation) and several *foreign keys* (where a primary key in one relation can be linked to the foreign key in another). If we assume a two-class problem, then we pick one class as the **positive** class and the other as the **negative** class. The most important task for building an accurate multirelational classifier is to find relevant features in different relations that help distinguish *positive* and *negative* target tuples.

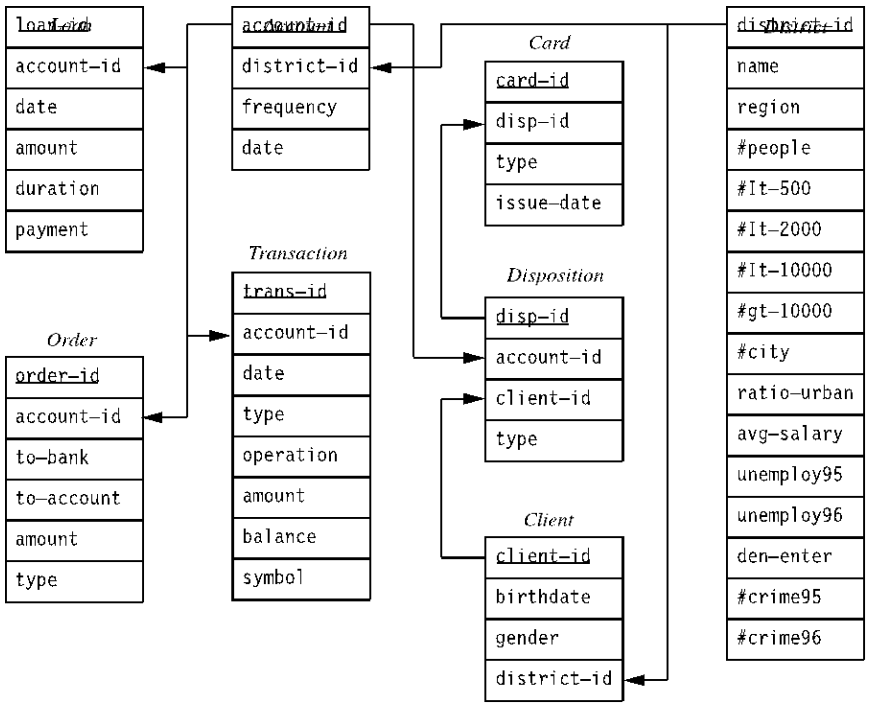


Figure 9.19 A financial database (from [PKDD CUP 99]).

9.3.2 *ILP Approach to Multirelational Classification*

Inductive Logic Programming (ILP) is the most widely used category of approaches to multirelational classification. There are many ILP approaches. In general, they aim to find hypotheses of a certain format that can predict the class labels of target tuples, based on background knowledge (i.e., the information stored in all relations). The ILP problem is defined as follows: *Given background knowledge B, a set of positive examples P, and a set of negative examples N, find a hypothesis H such that: (1)* $\forall t \in P : H[B \models t$ *(completeness), and (2)* $\forall t \in N : H[B \not\models t$ *(consistency), where* $\models$ *stands for logical implication.*

Well-known ILP systems include FOIL, Golem, and Progol. FOIL is a top-down learner, which builds rules that cover many positive examples and few negative ones. Golem is a bottom-up learner, which performs generalizations from the most specific rules. Progol uses a combined search strategy. Recent approaches, like TILDE, Mr- SMOTI, and RPTs, use the idea of C4.5 and inductively construct decision trees from relational data.

9.3.3 Tuple ID Propagation

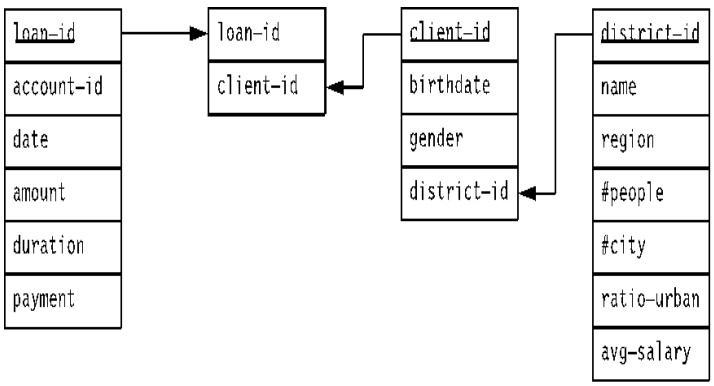
Tuple ID propagation is a technique for performing virtual join, which greatly improves efficiency of multirelational classification. Instead of physically joining relations, they are virtually joined by attaching the IDs of target tuples to tuples in non target relations. In this way the predicates can be evaluated as if a physical join were performed. Tuple ID propagation is flexible and efficient, because IDs can easily be propagated between any two relations, requiring only small amounts of data transfer and extra storage space. By doing so, predicates in different relations can be evaluated with little redundant computation.

9.3.4 Multirelational Classification Using Tuple ID Propagation

In this section we introduce **CrossMine**, an approach that uses tuple ID propagation for multirelational classification. To better integrate the information of ID propagation, CrossMine uses *complex predicates* as elements of rules. A complex predicate, p, contains two parts:

1. *prop-path*: This indicates how to propagate IDs. For example, the path "*Loan*: *account ID* !*Account*:*account ID*" indicates propagating IDs from *Loan* to *Account* using *account ID*. If no ID propagation is involved, *prop-path* is empty.

2. *constraint*: This is a predicate indicating the constraint on the relation to which the IDs are propagated. It can be either categorical or numerical.



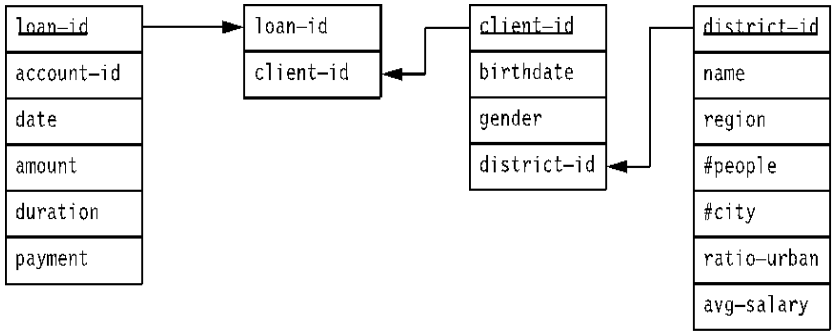


Fig 9.24 Another example database

9.3.5 Multirelational Clustering with User Guidance

Multirelational clustering is the process of partitioning data objects into a set of clusters based on their similarity, utilizing information in multiple relations. In this section we will introduce **CrossClus** (Cross-relational Clustering with user guidance), an algorithm for multirelational clustering that explores how to utilize e user guidance in clustering and tuple ID propagation to avoid physical joins.

One major challenge in multirelational clustering is that there are too many attributes in different relations, and usually only a small portion of them are relevant to a specific clustering task. Consider the computer science department database of Figure 9.25. In order to cluster students, attributes cover many different aspects of information, such as courses taken by students, publications of students, advisors and research groups of students, and so on. A user is usually interested in clustering students using a certain.

UNIT-VII

Mining Object, Spatial, Multimedia, Text, and Web Data

10.1 Multidimensional Analysis and Descriptive Mining of Complex Data Objects

Each object in a class is associated with (1) an object-identifier, (2) a set of attributes that may contain sophisticated data structures, set- or list-valued data, class composition hierarchies, multimedia data, and (3) a set of methods that specify the computational routines or rules associated with the object class. There has been extensive research in the field of database systems on how to efficiently index, store, access, and manipulate complex objects in object- relational and object-oriented database systems. Technologies handling these issues are discussed in many books on database systems, especially on object-oriented and object-relational database system

This includes two major tasks: (1) construct multidimensional data warehouses for complex object data and perform online analytical processing (OLAP) in such data warehouses, and (2) develop effective and scalable methods for mining knowledge from object databases and/or data warehouses. The second task is largely covered by the mining of specific kinds of data (such as spatial, temporal, sequence, graph- or tree-structured, text, and multimedia data), since these data form the major new kinds of complex data objects.

A major limitation of many commercial data warehouse and OLAP tools for multidimensional database analysis is their restriction on the allowable data types for dimensions and measures. Most data cube implementations confine dimensions to nonnumeric data, and measures to simple, aggregated values. To introduce data mining and multidimensional data analysis for complex objects, this section examines how to perform generalization on complex structured objects and construct object cubes for OLAP and mining in object databases. To facilitate generalization and induction in object-relational and object-oriented databases, it is important to study how each component of such databases can be generalized, and how the generalized data can be used for multidimensional data analysis and data mining.

10.1.1 Generalization of Structured Data

An important feature of object-relational and object-oriented databases is their capability of storing, accessing, and modeling complex structure-valued data, such as set- and list-valued data and data with nested structures. “How can generalization be performed on such data?” Let’s start by looking at the generalization of set-valued, list-valued, and sequence-valued attributes. A set-valued attribute may be of homogeneous or heterogeneous type.

Typically, set-valued data can be generalized by (1) generalization of each value in the set to its corresponding higher-level concept, or (2) derivation of the general behavior of the set, such as the number of elements in the set, the types or value ranges in the set, the weighted average for numerical data, or the major clusters formed by the set. Moreover, generalization can be

performed by applying different generalization operators to explore alternative generalization paths. In this case, the result of generalization is a heterogeneous set.

Example 10.1 Generalization of a set-valued attribute. Suppose that the hobby of a person is a set-valued attribute containing the set of values {tennis, hockey, soccer, violin, SimCity}. This set can be generalized to a set of high-level concepts, such as {sports, music, computer games} or into the number 5 (i.e., the number of hobbies in the set). Moreover, a count can be associated with a generalized value to indicate how many elements are generalized to that value, as in
{sports(3), music(1), computer games(1)}, where sports(3)indicates three kinds of sports, and so on. A set-valued attribute may be generalized to a set-valued or a single-valued attribute; a single-valued attribute may be generalized to a set-valued attribute if the values form a lattice or "hierarchy" or if the generalization follows different paths. Further generalizations on such a generalized set-valued attribute should follow the generalization path of each value in the set.

List-valued attributes and sequence-valued attributes can be generalized in a manner similar to that for set-valued attributes except that the order of the elements in the list or sequence should be preserved in the generalization. Each value in the list can be generalized into its corresponding higher-level concept. Alternatively, a list can be generalized according to its general behavior, such as the length of the list, the type of list elements, the value range, the weighted average value for numerical data, or by dropping unimportant elements in the list. A list may be generalized into a list, a set, or a single value.

10.1.2 Aggregation and Approximation in Spatial and Multimedia Data Generalization

Aggregation and approximation are another important means of generalization. They are especially useful for generalizing attributes with large sets of values, complex structures, and spatial or multimedia data.

Let's take spatial data as an example. We would like to generalize detailed geographic points into clustered regions, such as business, residential, industrial, or agricultural areas, according to land usage. Such generalization often requires the merge of a set of geographic areas by spatial operations, such as spatial union or spatial.

In a spatial merge, it is necessary to not only merge the regions of similar types within the same general class but also to compute the total areas, average density, or other aggregate functions while ignoring some scattered regions with different types if they are unimportant to the study. Other spatial operators, such as spatial-union, spatial-overlapping, and

spatial- intersection (which may require the merging of scattered small regions into large, clustered regions) can also use spatial aggregation and approximation as data generalization operators.

A multimedia database may contain complex texts, graphics, images, video fragments, maps, voice, music, and other forms of audio/video information. Multimedia data are typically

stored as sequences of bytes with variable lengths, and segments of data are linked together or indexed in a multidimensional way for easy reference.

10.1.3 Generalization of Object Identifiers and Class/Subclass Hierarchies

"How can object identifiers be generalized?" At first glance, it may seem impossible to generalize an object identifier. It remains unchanged even after structural reorganization of the data. However, since objects in an object-oriented database are organized into classes, which in turn are organized into class/subclass hierarchies, the generalization of an object can be performed by referring to its associated hierarchy. Thus, an object identifier can be generalized as follows. First, the object identifier is generalized to the identifier of the lowest subclass to which the object belongs. The identifier of this subclass can then, in turn, be generalized to a higher level class/subclass identifier by climbing up the class/subclass hierarchy. Similarly, a class or a subclass can be generalized to its corresponding superclass(es) by climbing up its associated class/subclass hierarchy.

10.1.4 Generalization of Class Composition Hierarchies

An attribute of an object may be composed of or described by another object, some of whose attributes may be in turn composed of or described by other objects, thus forming a class composition hierarchy. Generalization on a class composition hierarchy can be viewed as generalization on a set of nested structured data (which are possibly infinite, if the nesting is recursive).

10.1.5 Construction and Mining of Object Cubes

So, how can class-based generalization be performed for a large set of objects?" For classbased generalization, the attribute-oriented induction method developed. Generalization can continue until the resulting class contains a small number of generalized objects that can be summarized as a concise, generalized rule in high-level terms.

the generalization of multidimensional attributes of a complex object class can be performed by examining each attribute (or dimension), generalizing each attribute to simple- valued data, and constructing a multidimensional data cube, called an object cube. Once an object cube is constructed, multidimensional analysis and data mining can be performed on it in a manner similar to that for relational data cubes.

10.1.6 Generalization-Based Mining of Plan Databases by Divide-and-Conquer

A plan consists of a variable sequence of actions. A plan database, or simply a plan base, is a large collection of plans. Plan mining is the task of mining significant patterns or knowledge from a plan base. Plan mining can be used to discover travel patterns of business passengers in an air flight

database or to find significant patterns from the sequences of actions in the repair of automobiles. Plan mining is different from sequential pattern mining, where a large number of

frequently occurring sequences are mined at a very detailed level. Instead, plan mining is the extraction of important or significant generalized (sequential) patterns from a plan base.

10.2 Spatial Data Mining

A spatial database stores a large amount of space-related data, such as maps, preprocessed remote sensing or medical imaging data, and VLSI chip layout data. Spatial databases have many features distinguishing them from relational databases. They carry topological and/or distance information, usually organized by sophisticated, multidimensional spatial indexing structures that are accessed by spatial data access methods and often require spatial reasoning, geometric computation, and spatial knowledge representation techniques.

Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial databases. Such mining demands an integration of data mining with spatial database technologies. It can be used for understanding spatial data, discovering spatial relationships and relationships between spatial and non spatial data, constructing spatial knowledge bases, reorganizing spatial databases, and optimizing spatial queries.

Statistical spatial data analysis has been a popular approach to analyzing spatial data and exploring geographic information. The term geostatistics is often associated with continuous geographic space, patial statistics is often associated with discrete space.

In a statistical model that handles nonspatial data, one usually assumes statistical independence among different portions of data. However, different from traditional data sets, there is no such independence among spatially distributed data because in reality, spatial objects are often interrelated, or more exactly spatially co-located, in the sense that the closer the two objects are located, the more likely they share similar properties.

People even consider this as the first law of geography: “Everything is related to everything else, but nearby things are more related than distant things.” Such a property of close interdependency across nearby space leads to the notion of spatial autocorrelation.

10.2.1 Spatial Data Cube Construction and Spatial OLAP

“Can we construct a spatial data warehouse?” Yes, as with relational data, we can integrate
spatial data to construct a data warehouse that facilitates spatial data mining.

A spatial data warehouse is a subject-oriented, integrated, time-

variant, and nonvolatile collection of both spatial and non spatial data in support of spatial data mining and spatial-data related decision-making processes.

There are three types of dimensions in a spatial data cube:

A non spatial dimension contains only non spatial data.

Non spatial dimensions temperature and precipitation can be constructed for the warehouse in since each contains non spatial data whose generalizations are non spatial (such as "hot" for temperature and "wet" for precipitation).

A spatial-to-non spatial dimension is a dimension whose primitive-level data are spatial but whose generalization, starting at a certain high level, becomes non spatial. For example, the spatial dimension city relays geographic data for the U.S. map.

Suppose that the dimension's spatial representation of, say, Seattle is generalized to the string "pacific northwest." Although "pacific northwest" is a spatial concept, its representation is not spatial (since, in our example, it is a string). It therefore plays the role of a non spatial dimension.

Aspatial-to-spatial dimension is a dimension whose primitive level and all of its high level generalized data are spatial. For example, the dimension equi temperature region contains spatial data, as do all of its generalizations, such as with regions covering 0-5 degrees (Celsius), 5-10 degrees, and so on.

We distinguish two types of measures in a spatial data cube:

1)A numerical measure contains only numerical data. For example, one measure in a spatial data warehouse could be the monthly revenue of a region, so that a roll-up may compute the total revenue by year, by county, and so on. Numerical measures can be further classified into distributive, algebraic, and holistic

2) A spatial measure contains a collection of pointers to spatial objects. For example, in a generalization (or roll-up) in the spatial data cube of Example 10.5, the regions with the same range of temperature and precipitation will be grouped into the same cell, and the measure so formed contains a collection of pointers to those regions.

10.2.2 Mining Spatial Association and Co-location Patterns

For mining spatial associations related to the spatial predicate close to,

we can first collect the candidates that pass the minimum support threshold by Applying certain rough spatial evaluation algorithms, for example, using an MBR structure (which registers only two spatial points rather than a set of complex polygons), and

Evaluating the relaxed spatial predicate, g close to, which is a generalized close to covering a broader context that includes close to, touch,

and intersect.

10.2.3 Spatial Clustering Methods Spatial data

clustering identifies clusters, or densely populated regions, according to some distance measurement in a large, multidimensional data set.

10.2.4 Spatial Classification and Spatial Trend Analysis

Spatial classification analyzes spatial objects to derive classification schemes in relevance to certain spatial properties, such as the neighborhood of a district, highway, or river.

Spatial trend analysis deals with another issue: the detection of changes and trends along a spatial dimension. Typically, trend analysis detects changes with time, such as the changes of temporal patterns in time-series data. Spatial trend analysis replaces time with space and studies the trend of non spatial or spatial data changing with space.

10.2.5 Mining Raster Databases

Spatial database systems usually handle vector data that consist of points, lines, polygons (regions), and their compositions, such as networks or partitions. Typical examples of such data include maps, design graphs, and 3-D representations of the arrangement of the chains of protein molecules. However, a huge amount of space-related data are in digital raster (image) forms, such as satellite images, remote sensing data, and computer tomography. It is important to explore data mining in raster or image databases. Methods for mining raster and image data are examined in the following section regarding the mining of multimedia data.

10.3 Multimedia Data Mining

A multimedia database system stores and manages a large collection of multimedia data, such as audio, video, image, graphics, speech, text, document, and hypertext data, which contain text, text markups, and linkages. Multimedia database systems are increasingly common owing to the popular use of audio video equipment, digital cameras, CD-ROMs, and the Internet. Typical multimedia database systems include NASA's EOS (Earth Observation System), various kinds of image and audio-video databases, and Internet databases.

10.3.1 Similarity Search in Multimedia Data

For similarity searching in multimedia data, we consider two main families of multimedia indexing and retrieval systems: (1) description-based retrieval systems, which build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation; and (2) content-based retrieval systems, which support retrieval based on the image content, such as color histogram, texture, pattern, image topology, and the

shape of objects and their layouts and locations within the image.

Image-sample-based queries find all of the images that are similar to the given image sample. This search compares the feature vector (or signature) extracted from the sample with the feature

vectors of images that have already been extracted and indexed in the image database. Based on this comparison, images that are close to the sample image are returned.

Image feature specification queries specify or sketch image features like color, texture, or shape, which are translated into a feature vector to be matched with the feature vectors of the images in the database. Content-based retrieval has wide applications, including medical diagnosis, weather prediction, TV production,

Web search engines for images, and e-commerce. Some systems, such as QBIC (Query By Image Content), support both sample-based and image feature specification queries. There are also systems that support both content based and description-based retrieval.

Several approaches have been proposed and studied for similarity-based retrieval in image databases, based on image signature:

1)Color histogram–based signature:

In this approach, the signature of an image includes color histograms based on the color composition of an image regardless of its scale orientation. This method does not contain any information about shape, image topology, or texture. Thus, two images with similar color composition but that contain very different shapes or textures may be identified as similar, although they could be completely unrelated semantically.

2)Multi feature composed signature:

In this approach, the signature of an image includes a composition of multiple features: color histogram, shape, image topology, and texture. The extracted image features are stored as metadata, and images are indexed based on such metadata. Often, separate distance functions can be defined for each feature and subsequently combined to derive the overall results. Multidimensional content-based search often uses one or a few probe features to search for images containing such (similar) features. It can therefore be used to search for similar images. This is the most popularly used approach in practice.

3)Wavelet-based signature:

This approach uses the dominant wavelet coefficients of an image as its signature. Wavelets capture shape, texture, and image topology information in a single unified framework.1 This improves efficiency and reduces the need for providing multiple search primitives (unlike the second method above). However, since this method computes a single signature for an entire image, it may fail to identify images containing similar objects where the objects *differ* in location or size.

4)Wavelet-based signature with region-based granularity:

In this approach, the computation and comparison of signatures are at the granularity of regions, not the entire image. This is based on the observation that similar images may contain similar regions, but a region in one image could be a translation or scaling of a matching region in the other. Therefore, a similarity measure between the query image Q and a target image T can be

defined in terms of the fraction of the area of the two images covered by matching pairs of regions from *Q* and *T*. Such a region-based similarity search can find images containing similar objects, where these objects may be translated or scaled.

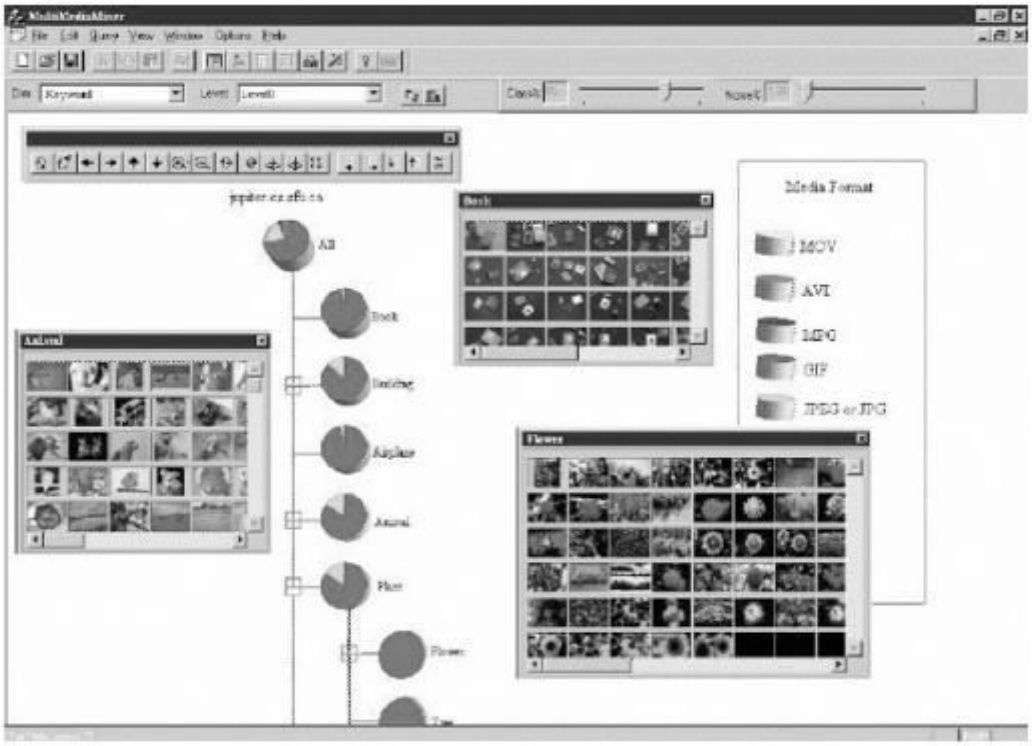
10.3.2 Multidimensional Analysis of Multimedia Data

A multimedia data cube can contain additional dimensions and measures for multimedia information, such as color, texture, and shape.

The example database tested in the Multi Media Miner system is constructed as follows. Each image contains two descriptors: a *feature descriptor* and a *layout descriptor*.

The original image is not stored directly in the database; only its descriptors are stored. The description information encompasses fields like image file name, image URL, image type (e.g., gif, tiff, jpeg,mpeg, bmp, avi), a list of all known Web pages referring to the image (i.e., parent URLs), a list of keywords, and a thumbnail used by the user interface for image and video browsing.

The **feature descriptor** is a set of vectors for each visual characteristic. The **layout descriptor** contains a color layout vector and an edge layout vector.



An output of the *Classifier* module of MultiMediaMiner.

10.3.3 Classification and Prediction Analysis of Multimedia Data

Classification and predictive modeling have been used for mining multimedia data, especially in scientific research, such as astronomy, seismology, and geo scientific research.

For Example:

Classification and prediction analysis of astronomy data. Taking sky images that have been carefully classified by astronomers as the training set, we can construct models for the recognition of galaxies, stars, and other stellar objects, based on properties like magnitudes, areas, intensity, image moments, and orientation. A large number of sky images taken by telescopes or space probes can then be tested against the constructed models in order to identify new celestial bodies. Similar studies have successfully been performed to identify volcanoes on Venus.

10.3.4 Mining Associations in Multimedia Data

Association rules involving multimedia objects can be mined in image and video databases. At least three categories can be observed:

Associations between image content and non image content features:

A rule like "If at least 50% of the upper part of the picture is blue, then it is likely to represent sky" belongs to this category since it links the image content to the keyword sky.

Associations among image contents that are not related to spatial relationships:

A rule like "If a picture contains two blue squares, then it is likely to contain one red circle as well" belongs to this category since the associations are all regarding image contents.

Associations among image contents related to spatial relationships:

A rule like "If a red triangle is between two yellow squares, then it is likely a big oval-shaped object is underneath" belongs to this category since it associates objects in the image with spatial relationships.

"What are the differences between mining association rules in multimedia databases versus in transaction databases?"

1)An image may contain multiple objects, each with many features such as color, shape, texture, keyword, and spatial location, Such a multi resolution mining strategy substantially reduces the overall data mining cost without loss of the quality and completeness of data mining results. This leads to an efficient methodology for mining frequent item sets and associations in large multimedia databases.

2) Because a picture containing multiple recurrent objects is an important feature in image analysis, recurrence of the same objects should not be ignored in association analysis.

3)There often exist important spatial relationships among multimedia objects, such as above, beneath, between, nearby, left-of, and so on.

10.3.5 Audio and Video Data Mining

To facilitate the recording, search, and analysis of audio and video information from multimedia data, industry and standardization committees have made great strides toward developing a set of standards for multimedia information description and compression. For example, MPEG-k (developed by MPEG: Moving Picture Experts Group) and JPEG are typical video compression schemes. The most recently released MPEG-7,formally named "Multimedia Content Description Interface," is a standard for

describing the multimedia content data. It

supports some degree of interpretation of the information meaning, which can be passed onto, or accessed by, a device or a computer.

The MPEG committee standardizes the following elements in MPEG-7: (1) a set of descriptors, where each descriptor defines the syntax and semantics of a feature, such as color, shape, texture, image topology, motion, or title; (2) a set of descriptor schemes, where each scheme specifies the structure and semantics of the relationships between its components (descriptors or description schemes); (3) a set of coding schemes for the descriptors, and (4) a description definition language (DDL) to specify schemes and descriptors. Such standardization greatly facilitates content-based video retrieval and video data mining.

10.4 Text Mining

Information is stored in text databases (or document databases), which consist of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, e-mail messages, and Web pages. Text databases are rapidly growing due to the increasing amount of information available in electronic form, such as electronic publications, various kinds of electronic documents, e-mail, and the World Wide Web (which can also be viewed as a huge, interconnected, dynamic text database). Nowadays most of the information in government, industry, business, and other institutions are stored electronically, in the form of text databases.

Data stored in most text databases are semi structured data in that they are neither completely unstructured nor completely structured. For example, a document may contain a few structured fields, such as title, authors, publication date, category, and so on, but also contain some largely unstructured text components, such as abstract and contents.

10.4.1 Text Data Analysis and Information Retrieval

Information retrieval (IR) is a field that has been developing in parallel with database systems for many years. Due to the abundance of text information, information retrieval has found many applications. There exist many information retrieval systems, such as on-line library catalog systems, on-line document management systems, and the more recently developed Web search engines.

Basic Measures for Text Retrieval: Precision and Recall

Precision: This is the percentage of retrieved documents that are in fact relevant to the query
(i.e., “correct” responses). It is formally defined as

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}.$$

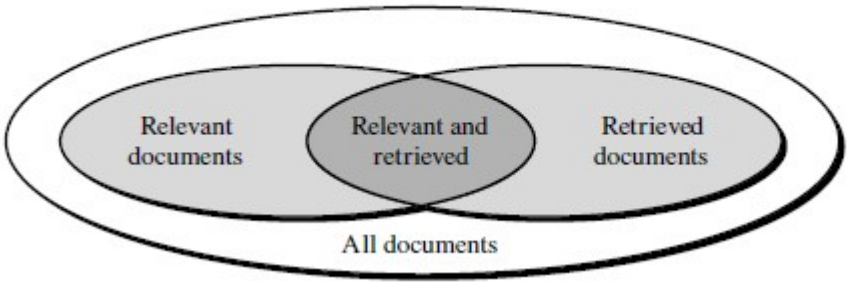
Recall: This is the percentage of documents that are relevant to the query and were, in fact, retrieved. It is formally defined as

$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}.$$

F-score, which is defined as the harmonic mean of recall and precision:

$$F\_score = \frac{recall \times precision}{(recall + precision)/2}.$$

The harmonic mean discourages a system that sacrifices one measure for another too drastically.



Relationship between the set of relevant documents and the set of retrieved documents.

Text Retrieval Methods

Retrieval methods fall into two categories: They generally either view the retrieval problem as a document
selection problem or as a document ranking problem

In document selection methods, the query is regarded as specifying constraints for selecting relevant documents.

Document ranking methods use the query to rank all documents in the order of relevance. How can we model a document to facilitate information retrieval?" Starting with a set of d documents and a set of t terms, we can model each document as a vector v in the t dimensional space R t , which is why this method is called the vector-space model. Let the term frequency be the number of occurrences of term t in the document d, that is, freq(d; t). The (weighted) term-frequency matrix TF(d; t) measures the association of a term t with respect to the given document d: it is generally defined as 0 if the document
does not contain the term, and nonzero otherwise. There are many ways to define the term- weighting for the nonzero entries in such a vector. For example, we can simply set TF(d; t) = 1 if the termt occurs in the document d, or use the term frequency freq(d; t), or the relative term frequency, that is, the term frequency versus the total number of occurrences of all the terms in the document. There are also other ways to normalize the term frequency. For example, the Cornell SMART system uses the following formula to compute the (normalized) term frequency:

$$TF(d; t) = \begin{cases} 0 & \text{if } freq(d; t) = 0 \\ 1 + \log(1 + \log(freq(d; t))) & \text{otherwise} \end{cases}$$

Inverse document frequency (IDF), that represents the scaling factor, or the importance, of a term t. If a term t occurs in many documents.

where d is the document collection, and dt is the set of documents containing term t. If jdt j_jdj, the term t will have a large IDF scaling factor and vice versa.

$$IDF(t) = \log \frac{1+|d|}{|d_t|},$$

In a complete vector-space model, TF and IDF are combined together, which forms the TF-IDF measure:

$$TF\text{-}IDF(d,t) = TF(d,t) \times IDF(t).$$

Term frequency and inverse document frequency. Table 10.5 shows a term frequency matrix where each row represents a document vector, each column represents a term, and each entry registers $freq(di; t\ j)$, the number of occurrences of term $t\ j$ in document di. Based on this table we can calculate the TF-IDF value of a term in a document. For example, for $t6$ in $d4$, we have

$$TF(d_4,t_6) = 1 + \log(1 + \log(15)) = 1.3377$$

$$IDF(t_6) = \log \frac{1+5}{3} = 0.301.$$

$$TF\text{-}IDF(d_4,t_6) = 1.3377 \times 0.301 = 0.403$$

A representative metric is the cosine measure, defined as follows. Let $v1$ and $v2$ be two document vectors. Their cosine similarity is defined as

$$sim(v_1,v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|},$$

where the inner product $v_1 \cdot v_2$ is the standard vector dot product, defined as $\Sigma_{i=1}^{v} v_{1i}v_{2i}$, and the norm $|v_1|$ in the denominator is defined as $|v_1| = \sqrt{v_1 \cdot v_1}$.

A term frequency matrix showing the frequency of terms per document.

document/term	t_1	t_2	t_3	t_4	t_5	t_6	t_7
d_1	0	4	10	8	0	5	0
d_2	5	19	7	16	0	0	32
d_3	15	0	0	4	9	0	17
d_4	22	3	12	0	5	15	0
d_5	0	7	0	9	2	4	12

Text Indexing Techniques

There are several popular text retrieval indexing techniques, including *inverted indices* and *signature files*.

An inverted index is an index structure that maintains two hash indexed or B+-tree indexed tables: *document table* and *term table*, where

a) *document table* consists of a set of document records, each containing two fields: *doc id* and *posting list*, where *posting list* is a list of terms (or pointers to terms) that occur in the document, sorted according to some relevance measure.

b) *term table* consists of a set of term records, each containing two fields: *term id* and *posting list*, where *posting list* specifies a list of document identifiers in which the term appears.

A signature file is a file that stores a *signature* record for each document in the database. Each signature has a fixed size of *b* bits representing terms. A simple encoding scheme goes as follows. Each bit of a document signature is initialized to 0. A bit is set to 1 if the termit represents appears in the document. A signature *S*1 matches another signature *S*2 if each bit that is set in signature *S*2 is also set in *S*1.

Query Processing Techniques

Once an inverted index is created for a document collection, a retrieval system can answer a keyword query quickly by looking up which documents contain the query keywords. Specifically, we will maintain a score accumulator for each document and update these accumulators as we go through each query term.

For each query term, we will fetch all of the documents that match the term and increase their scores. accumulators as we go through each query term. For each query term, we will fetch all of the documents that match the term and increase their scores.

Dimensionality reduction techniques such as latent semantic indexing, probabilistic latent semantic analysis, and locality preserving indexing

Latent Semantic Indexing

Latent semantic indexing (LSI) is one of the most popular algorithms for document dimensionality reduction. It is fundamentally based on SVD (singular value decomposition). Suppose the *rank* of the term-document X is r, then LSI decomposes X using SVD as follows:

$$X = U\Sigma V^T,$$

$$a_{opt} = \arg\min_{a} \|X - aa^T X\|^2 = \arg\max_{a} a^T XX^T a,$$

with the constraint,

$$a^T a = 1.$$

since XX^T is symmetric, the basis functions of LSI are orthogonal.

Locality Preserving Indexing

The basic idea of LPI is to preserve the locality information (i.e., if two documents are near each other in the original document space, LPI tries to

keep these two documents close together in the reduced dimensionality space).

$$a_{opt} = \arg\min_{a} \sum_{i,j} \left(a^T x_i - a^T x_j\right)^2 S_{ij} = \arg\min_{a} a^T X L X^T a,$$

Given the document set ***x1***; : : : ;***xn*** 2 R*m*, LPI constructs a similarity matrix *S* 2 R*n*_*n*. The transformation vectors of LPI can be obtained by solving the following minimization problem:

$$a^T X D X^T a = 1,$$

where *L* = *D* *S* is the *Graph Laplacian* and *Dii* = å*j* *Si j*. *Dii* measures the local density around ***xi***. LPI constructs the similarity matrix *S* as where *L* = *D*-*S* is the *Graph Laplacian* and *Dii* = å*j* *Si j*. *Dii* measures the local density around ***xi***. LPI constructs the similarity matrix *S* as

$$S_{ij} = \begin{cases} \frac{x_i^T x_j}{\|x_i^T x_j\|}, & \text{if } x_i \text{ is among the } p \text{ nearest neighbors of } x_j \\ & \text{or } x_j \text{ is among the } p \text{ nearest neighbors of } x_i \\ 0, & \text{otherwise.} \end{cases}$$

"close" then y_i $(=a^T x_i)$ and y_j $(=a^T x_j)$ are close as well. Finally, the basis functions of LPI are the eigenvectors associated with the smallest eigenvalues of the following generalized eigen-problem:

$$XLX^T a = \lambda XDX^T a.$$

Probabilistic Latent Semantic Indexing

The probabilistic latent semantic indexing (PLSI) method is similar to LSI, but achieves dimensionality reduction through a probabilistic mixture model. Specifically, we assume there are *k* latent common themes in the document collection, and each is characterized by a multinomial word distribution

$$p_{d_i}(w) = \sum_{j=1}^{k} [\pi_{d_i,j} p(w|\theta_j)]$$

Formally, let *C* = f*d*1;*d*2; : : : ;*dn*g be a collection of *n* documents. Let q1; : : : ;q*k* be *k* theme multinomial distributions. A word *w* in document *di* is regarded as a sample of the following mixture model.

where $\pi_{d_i,j}$ is a document-specific mixing weight for the *j*-th aspect theme, and $\sum_{j=1}^{k} \pi_{d_i,j} = 1$.

The log-likelihood of the collection *C* is

$$\log p(C|\Lambda) = \sum_{i=1}^{n} \sum_{w \in V} [c(w,d_i) \log(\sum_{j=1}^{k} (\pi_{d_i,j} p(w|\theta_j)))],$$

10.4.3 Text Mining Approaches

The major approaches, based on the kinds of data they take as input, are:

(1) the keyword-based approach, where the input is a set of keywords or terms in the documents,

(2) the tagging approach, where the input is a set of tags, and (3) the information-extraction approach, which inputs semantic information, such as events, facts, or entities uncovered by information extraction.

Various text mining tasks can be performed on the extracted keywords, tags, or semantic information. These include document clustering, classification, information extraction, association analysis, and trend analysis .We examine a few such tasks in the following discussion.

1)Keyword-Based
Association Analysis
2)Document
Classification Analysis

10.5 Mining the World Wide Web

The World Wide Web serves as a huge ,widely distributed, global information service center for news, advertisements, consumer information, financial management, education, government, e- commerce, and many other information services. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing rich sources for data mining. However, based on the following observations, the Web also poses great challenges for effective resource and knowledge discovery.

i)The Web seems to be too huge for effective data warehousing and data mining.
ii) The complexity of Web pages is far greater than that of any traditional text document Collection
iii) The Web is a highly dynamic information source.
iv) The Web serves a broad diversity of user communities
v) Only a small portion of the information on the Web is truly relevant or useful.

10.5.1 Mining the Web Page Layout Structure

The basic structure of a Web page is its DOM (Document Object Model) structure. The DOM structure of a Web page is a tree structure, where every HTML tag in the page corresponds to a node in the DOM tree.

```
<tr>
<td><img src='t1.jpg'></td><td><img src="t2.jpg"></td>
<td><img src='t3.jpg'></td><td><img src="t4.jpg"></td>
</tr>
<tr>
<td>Timber Wolf</td><td>Giraffes</td>
<td>Elephant Sunrise</td><td>Prowling Fox</td>
</tr>
```

(a) Part of HTML source (only keep the backbone)

(b) The DOM tree structure (The picture area and caption area are two different TR nodes)

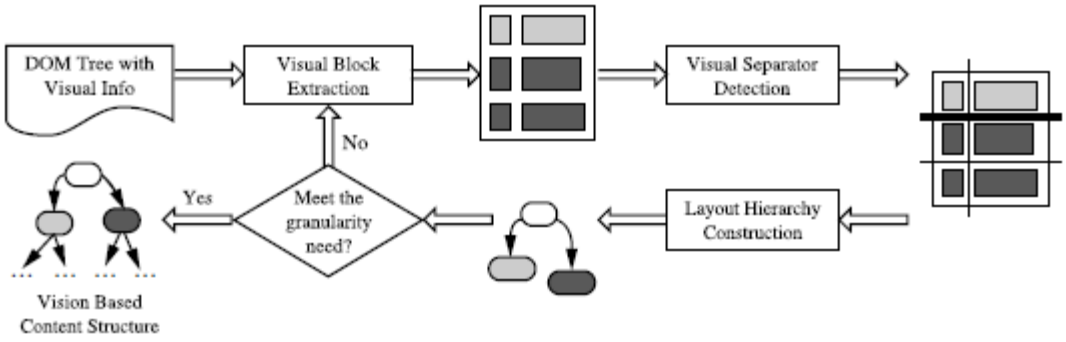
The HTML source and DOM tree structure of a sample page. It is difficult to extract the correct semantic content structure of the page.

There are many index-based Web search engines. These search the Web, index Web pages, and build and store huge keyword-based indices that help locate sets of Web pages containing certain keywords. a simple keyword-based search engine suffers from several deficiencies. First, a topic of any breadth can easily contain hundreds of thousands of documents. This can lead to a huge number of document entries returned by a search engine, many of which are only marginally relevant to the topic or may contain materials of poor quality. Second, many documents that are highly relevant to a topic may not contain keywords defining them. This is referred to as the *polysemy* problem

10.5.2 Mining the Web's Link Structures to Identify Authoritative Web Pages

But how can a search engine automatically identify authoritative Web pages for my topic?" Interestingly, the secrecy of authority is hiding in Web page linkages. The Web consists not only of pages, but also of hyperlinks pointing from one page to another. This idea has motivated some interesting studies on mining authoritative pages on the Web These properties of Web link structures have led researchers to consider another important category of Web pages called a hub. A hub is one or a set of Web pages that provides
collections of links to authorities. Hub pages may not be prominent, or there may exist few links pointing to them.

An algorithm using hubs, called HITS (Hyperlink-Induced Topic Search), was developed as follows. First, HITS uses the query terms to collect a starting set of, say, 200 pages from an index-based search engine. These pages form the root set. Since many of these pages are presumably relevant to the search topic, some of them should contain links to most of the prominent authorities. Therefore, the root set can be expanded into a base set by including all of the pages that the root-set pages link to and all of the pages that link to a page in the root set, up to a designated size cutoff such as 1,000 to 5,000 pages



The process flow of vision-based page segmentation algorithm.

Partition using VIPS (The image with their surrounding text are accurately identified)

We first associate a non-negative authorityweight, *ap*, and a non-negative hubweight, *hp*, with each page *p* in the base set, and initialize all *a* and *h* values to a uniform constant

$$a_p = \Sigma_{(q \text{ such that } q \to p)}\, h_q$$

$$h_p = \Sigma_{(q \text{ such that } q \leftarrow p)}\, a_q$$

These equations can be written in matrix form as follows. Let us number the pages f1;2; : : : ;*n*g and define their adjacency matrix *A* to be an *n_n* matrix where *A*(*i*; *j*) is 1 if page *i* links to page *j*, or 0 otherwise. Similarly, we define the authority weight vector $\boldsymbol{a}$ = (*a*1;*a*2; : : : ;*an*), and the hub weight vector $\boldsymbol{h}$ = (*h*1;*h*2; : : : ;*hn*). Thus, we have

$$\boldsymbol{h} = A \cdot \boldsymbol{a}$$

$$\boldsymbol{a} = A^T \cdot \boldsymbol{h},$$

where *AT* is the transposition of matrix *A*. Unfolding these two equations *k* times, we have

$$\boldsymbol{h} = A \cdot \boldsymbol{a} = AA^T\boldsymbol{h} = (AA^T)\boldsymbol{h} = (AA^T)^2\boldsymbol{h} = \cdots = (AA^T)^k\boldsymbol{h}$$

$$\boldsymbol{a} = A^T \cdot \boldsymbol{h} = A^TA\boldsymbol{a} = (A^TA)\boldsymbol{a} = (A^TA)^2\boldsymbol{a} = \cdots = (A^TA)^k\boldsymbol{a}.$$

The graph model in block-level link analysis is induced from two kinds of relationships ,that is,
block-to-page (link structure) and *page-to-block* (page layout). The block-to-page relationship is obtained from link analysis. Let Z denote the block-to-page matrix with dimension n_k. Z can be formally defined as follows:

$$Z_{ij} = \begin{cases} 1/s_i, & \text{if there is a link from block } i \text{ to page } j \\ 0, & \text{otherwise,} \end{cases}$$

where si is the number of pages to which block i links. $Zi\,j$ can also be viewed as a probability of jumping from block i to page j

The page-to-block relationships are obtained from page layout analysis. Let X denote the page- to-block matrix with dimension k_n.

$$X_{ij} = \begin{cases} f_{p_i}(b_j), & \text{if } b_j \in p_i \\ 0, & \text{otherwise,} \end{cases}$$

where *f* is a function that assigns to every block *b* in page *p* an importance value. Specifically, the bigger *fp*(*b*) is, the more important the block *b* is. Function *f* is empirically defined below,

$$f_p(b) = \alpha \times \frac{\text{the size of block } b}{\text{the distance between the center of } b \text{ and the center of the screen}},$$

where a is a normalization factor to make the sum of *fp*(*b*) to be 1, that is,

$$\sum_{b \in p} f_p(b) = 1$$

fp(b) can also be viewed as a probability that the user is focused on the block b when viewing the page p

block-to-page and page-to-block relations, a new Web page graph that incorporates the block importance information can be defined as

$$W_{\wp} = XZ,$$

where *X* is a *k_n* page-to-block matrix, and *Z* is a *n_k* block-to-page matrix. Thus *W*P is a *k_k*
page-to-page matrix

10.5.3 Mining Multimedia Data on the Web

Web-based multimedia data are embedded on the Web page and are associated with text and link information. These texts and links can also be regarded as features of the multimedia data.

To construct a Web-image graph, in addition to the *block-to-page* and *page-to-block* relations, we need to consider a new relation: block-to-image relation. Let *Y* denote the block-to-image matrix with dimension *n_m*.

$$Y_{ij} = \begin{cases} 1/s_i, & \text{if } I_j \in b_i \\ 0, & \text{otherwise,} \end{cases}$$

where *si* is the number of images contained in the image block *bi*.

$$W_{\mathcal{B}} = (1-t)ZX + tD^{-1}U,$$

where *t* is a suitable constant. *D* is a diagonal matrix, *Dii* = å*jUi j*. *Ui j* is 0 if block *i* and block *j* are contained in two different Web pages; otherwise, it is set to the *DOC* (*degree of coherence*, a property of the block,)

$$W_I = Y^T W_{\mathcal{B}} Y,$$

*W*I is an *m_m* matrix. If two images *i* and *j* are in the same block, say *b*, then *W*I (*i*; *j*) = *W*B(*b*;*b*)
= 0.

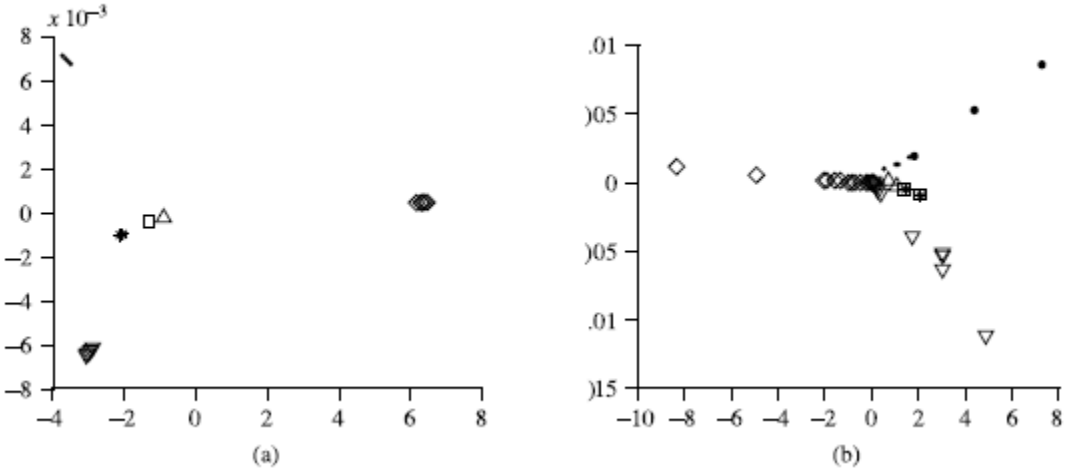
$$W_I = tD^{-1}Y^T Y + (1-t)Y^T W_{\mathcal{B}} Y,$$

t is a suitable constant, and D is a diagonal matrix, $Dii = \sum_j (YTY)_{ij}$.

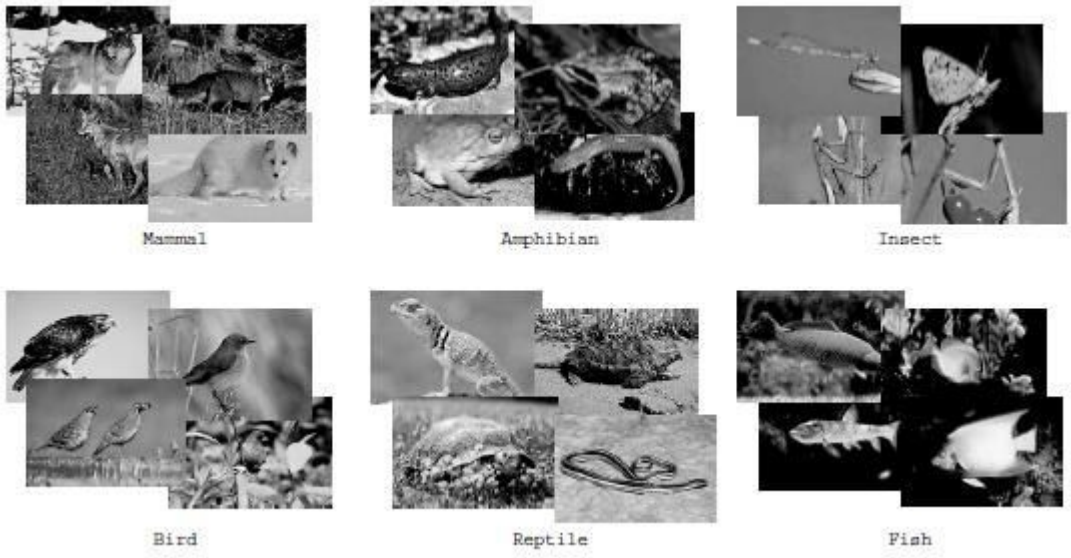
10.5.4 Automatic Classification of Web Documents

In the automatic classification of Web documents, each document is assigned a class label from a set of predefined topic categories, based on a set of examples of pre classified documents. Keyword-based document classification can be used for Web document classification. Such a term-based classification scheme has shown good results in Web page classification.

There have been extensive research activities on the construction and use of the semantic Web, a Web information infrastructure that is expected to bring structure to the Web based on the semantic meaning of the contents of Web pages. Web document classification by Web mining will help in the automatic extraction of the semantic meaning of Web pages and build up ontology for the semantic Web.



2-D embedding of the WWW images. (a) The image graph is constructed using block-level link analysis. Each color (shape) represents a semantic category. Clearly, they are well separated. (b) The image graph was constructed based on traditional perspective that the hyperlinks are considered from pages to pages. The image graph was induced from the page-to-page and page-to-image relationships.



Six image categories.

10.5.5 Web Usage Mining

Besides mining Web contents and Web linkage structures, another important task for Web mining is Web usage mining, which mines Weblog records to discover user access patterns of Web pages.

A Web server usually registers a (Web) log entry, or Weblog entry, for every access of a Web page. It includes the URL requested, the IP address from which the request originated,
and a timestamp.

In developing techniques for Web usage mining, we may consider the following.

First,Although it is encouraging and exciting to imagine the various potential applications of Weblog file analysis, it is important to know that the success of such applications depends on what and how much valid and reliable knowledge can be discovered from the large raw log data. Often, raw Weblog data need to be cleaned, condensed, and transformed in order to retrieve and analyze significant and useful information.

Second, with the available URL, time, IP address, and Web page content information, a multidimensional view can be constructed on the Weblog database, and multidimensional OLAP analysis can be performed to find the top N users, top N accessed Web pages, most frequently accessed time periods, and so on, which will help discover potential customers, users, markets, and others.

Third, data mining can be performed on Web log records to find association patterns, sequential patterns, and trends of Web accessing.

11.1 Data Mining Applications

Data mining is a young discipline with wide and diverse applications.There is still a nontrivial gap between general principles of data mining and domain-specific, effective data mining tools for particular applications

Some application domains (covered in this chapter)

1) Biomedical and DNA data analysis
2) Financial data analysis
3) Retail industry

Telecommunication industry

Biomedical Data Mining and DNA Analysis

1) DNA sequences: 4 basic building blocks (nucleotides): adenine (A), cytosine (C), guanine (G), and thymine (T).
2) Gene: a sequence of hundreds of individual nucleotides arranged in a particular order
3) Humans have around 100,000 genes
4) Tremendous number of ways that the nucleotides can be ordered and sequenced to form distinct genes
5) Semantic integration of heterogeneous, distributed genome databases

DNA Analysis: Examples

1) Similarity search and comparison among DNA sequences
   - Compare the frequently occurring patterns of each class (e.g., diseased and healthy)
   - Identify gene sequence patterns that play roles in various diseases
2) Association analysis: identification of co-occurring gene sequences
   - Most diseases are not triggered by a single gene but by a combination of genes acting together
   - Association analysis may help determine the kinds of genes that are likely to co-occur together in target samples
3) Path analysis: linking genes to different disease development stages

- Different genes may become active at different stages of the disease
- Develop pharmaceutical interventions that target the different stages separately

4) Visualization tools and genetic data analysis

11.1.1 Data Mining for Financial Data Analysis

1) Financial data collected in banks and financial institutions are often relatively complete, reliable, and of high quality.

2) Design and construction of data warehouses for multidimensional data analysis and data mining.

- View the debt and revenue changes by month, by region, by sector, and by other factors
- Access statistical information such as max, min, total, average, trend, etc.

■ Loan payment prediction/consumer credit policy analysis

- feature selection and attribute relevance ranking
- Loan payment performance
- Consumer credit rating

Financial Data Mining

1) Classification and clustering of customers for targeted marketing.

■ multidimensional segmentation by nearest-neighbor, classification, decision trees, etc. to identify customer groups or associate a new customer to an appropriate customer group

2) Detection of money laundering and other financial crimes

■ integration of from multiple DBs (e.g., bank transactions, federal/state crime history DBs)

Tools: data visualization, linkage analysis, classification, clustering tools, outlier analysis, and sequential pattern analysis tools (find unusual access sequences)

11.1.2 Data Mining for Retail Industry

1) Retail industry: huge amounts of data on sales, customer shopping history, etc.

2) Applications of retail data mining

- Identify customer buying behaviors
- Discover customer shopping patterns and trends
- Improve the quality of customer service

- Achieve better customer retention and satisfaction
- Enhance goods consumption ratios
- Design more effective goods transportation and distribution policies

Data Mining in Retail Industry: Examples

1) Design and construction of data warehouses based on the benefits of data mining
   - Multidimensional analysis of sales, customers, products, time, and region
2) Analysis of the effectiveness of sales campaigns
3) Customer retention: Analysis of customer loyalty
   - Use customer loyalty card information to register sequences of purchases of particular customers
   - Use sequential pattern mining to investigate changes in customer consumption or loyalty
   - suggest adjustments on the pricing and variety of goods
4) Purchase recommendation and cross-reference of items

11.1.3 Data Mining for Telecomm. Industry

1) A rapidly expanding and highly competitive industry and a great demand for data mining
   - Understand the business involved
   - Identify telecommunication patterns
   - Catch fraudulent activities
   - Make better use of resources
   - Improve the quality of service

2) Multidimensional analysis of telecommunication data

   Intrinsically multidimensional: calling-time, duration, location of caller, location of callee, type of call, etc

3) Fraudulent pattern analysis and the identification of unusual patterns
   - Identify potentially fraudulent users and their atypical usage patterns
   - Detect attempts to gain fraudulent entry to customer accounts
   - Discover unusual patterns which may need special attention

4) Multidimensional association and sequential pattern analysis

- Find usage patterns for a set of communication services by customer group, by month, etc.
- Promote the sales of specific services
- Improve the availability of particular services in a region

5) Use of visualization tools in telecommunication data analysis

11.2 Data mining system products and research prototypes

11.2.1 How to choose a data mining system?

1) Commercial data mining systems have little in common
   - Different data mining functionality or methodology
   - May even work with completely different kinds of data sets
2) Need multiple dimensional view in selection
3) Data types: relational, transactional, text, time sequence, spatial?
4) System issues
   - running on only one or on several operating systems?
   - a client/server architecture?
   - Provide Web-based interfaces and allow XML data as input and/or output?
5) Data sources
   - ASCII text files, multiple relational data sources
   - support ODBC connections (OLE DB, JDBC)?
6) Data mining functions and methodologies
   - One vs. multiple data mining functions
   - One vs. variety of methods per function
     - More data mining functions and methods per function provide the user with greater flexibility and analysis power
7) Coupling with DB and/or data warehouse systems

- Four forms of coupling: no coupling, loose coupling, semitight coupling, and tight coupling
    - Ideally, a data mining system should be tightly coupled with a database system

8) Scalability
    - Row (or database size) scalability
    - Column (or dimension) scalability
    - Curse of dimensionality: it is much more challenging to make a system column scalable that row scalable

9) Visualization tools
    - "A picture is worth a thousand words"
    - Visualization categories: data visualization, mining result visualization, mining process visualization, and visual data mining

10) Data mining query language and graphical user interface
    - Easy-to-use and high-quality graphical user interface
    - Essential for user-guided, highly interactive data mining

11.2.2 Examples of Data Mining Systems

1) IBM Intelligent Miner
    - A wide range of data mining algorithms
    - Scalable mining algorithms
    - Toolkits: neural network algorithms, statistical methods, data preparation, and data visualization tools
    - Tight integration with IBM's DB2 relational database system

2) SAS Enterprise Miner
    - A variety of statistical analysis tools
    - Data warehouse tools and multiple data mining algorithms

3) Mirosoft SQLServer 2000

- Integrate DB and OLAP with mining
- Support OLEDB for DM standard

4) SGI MineSet
   - Multiple data mining algorithms and advanced statistics
   - Advanced visualization tools
5) Clementine (SPSS)
   - An integrated data mining development environment for end-users and developers
   - Multiple data mining algorithms and visualization tools
6) DBMiner (DBMiner Technology Inc.)
   - Multiple data mining modules: discovery-driven OLAP analysis, association, classification, and clustering
   - Efficient, association and sequential-pattern mining functions, and visual classification tool
   - Mining both relational databases and data warehouses

11.3 Additional Themes on Data mining

11.3.1 Theoretical Foundations of Data Mining

1) Data reduction
   - The basis of data mining is to reduce the data representation
   - Trades accuracy for speed in response
2) Data compression
   - The basis of data mining is to compress the given data by encoding in terms of bits, association rules, decision trees, clusters, etc.
3) Pattern discovery
   - The basis of data mining is to discover patterns occurring in the database, such as associations, classification models, sequential patterns, etc.
4) Probability theory

- The basis of data mining is to discover joint probability distributions of random variables

5) Microeconomic view
    - A view of utility: the task of data mining is finding patterns that are interesting only to the extent in that they can be used in the decision-making process of some enterprise
6) Inductive databases
    - Data mining is the problem of performing inductive logic on databases,
    - The task is to query the data and the theory (i.e., patterns) of the database
    - Popular among many researchers in database systems

Data Mining and Intelligent Query Answering

1) Query answering
    - Direct query answering: returns exactly what is being asked
    - Intelligent (or cooperative) query answering: analyzes the intent of the query and provides generalized, neighborhood or associated information relevant to the query
2) Some users may not have a clear idea of exactly what to mine or what is contained in the database.
3) Intelligent query answering analyzes the user's intent and answers queries in an intelligent way.
4) A general framework for the integration of data mining and intelligent query answering
    - Data query: finds concrete data stored in a database.
    - Knowledge query: finds rules, patterns, and other kinds of knowledge in a database.
5) Ex. Three ways to improve on-line shopping service
    - Informative query answering by providing summary information
    - Suggestion of additional items based on association analysis Product promotion by sequential pattern mining

11.3.2 .2 Scientific and Statistical Data Mining

1. There are many well-established statistical techniques for data analysis, particularly for numeric data

- applied extensively to data from scientific experiments and data from economics and the social sciences

2. Regression

- predict the value of a response (dependent) variable from one or more predictor (independent) variables where the variables are numeric
- forms of regression: linear, multiple, weighted, polynomial, nonparametric, and robust.

3. Generalized linear models

- allow a categorical response variable (or some transformation of it) to be related to a set of predictor variables
- similar to the modeling of a numeric response variable using linear regression include logistic regression and Poisson regression

4. Regression trees

- Binary trees used for classification and prediction
- Similar to decision trees: Tests are performed at the internal nodes
- Difference is at the leaf level
    - In a decision tree a majority voting is performed to assign a class label to the leaf
    - In a regression tree the mean of the objective attribute is computed and used as the predicted value

5. Analysis of variance

- Analyze experimental data for two or more populations described by a numeric response variable and one or more categorical variables (factors)

6. Mixed-effect models

- For analyzing grouped data, i.e. data that can be classified according to one or more grouping variables

- Typically describe relationships between a response variable and some covariates in data grouped according to one or more factors

7. Factor analysis

- determine which vars are combined to generate a given factor
- e.g., for many psychiatric data, one can indirectly measure other quantities (such as test scores) that reflect the factor of interest

8. Discriminant analysis

- predict a categorical response variable, commonly used in social science
- Attempts to determine several discriminant functions (linear combinations of the independent variables) that discriminate among the groups defined by the response variable

9. Time series: many methods such as autoregression, ARIMA (Autoregressive integrated moving-average modeling), long memory time-series modeling

10. Survival analysis

- predict the probability that a patient undergoing a medical treatment would survive at least to time *t* (life span prediction)

11. Quality control

- display group summary charts

11.3.3 Visual Data Mining and Audio Data mining

1. Visualization: use of computer graphics to create visual images which aid in the understanding of complex, often massive representations of data

2. Visual Data Mining: the process of discovering implicit but useful knowledge from large data sets using visualization techniques

3. Purpose of Visualization

- Gain insight into an information space by mapping data onto graphical primitives
- Provide qualitative oveiew of large data sets
- Search for patterns, trends, structure, irregularities, relationships among data.

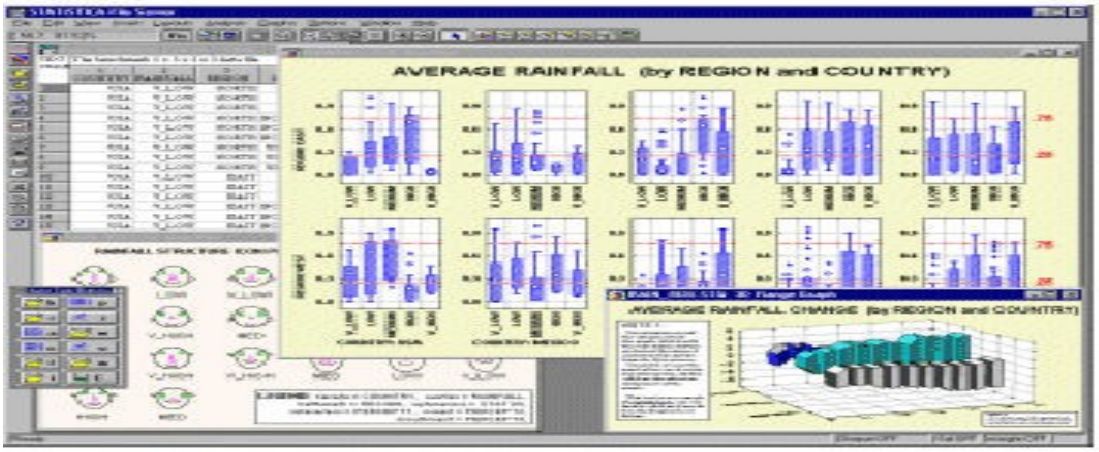
- Help find interesting regions and suitable parameters for further quantitative analysis.
- Provide a visual proof of computer representations derived

- *Integration of visualization and data mining*
  - data visualization
  - data mining result visualization
  - data mining process visualization
  - interactive visual data mining

1) Data visualization

- Data in a database or data warehouse can be viewed
  - at different levels of granularity or abstraction
  - as different combinations of attributes or dimensions
- *Data can be presented in various visual forms*

Fig 11.1 : Boxplots from Statsoft: multiple variable combinations

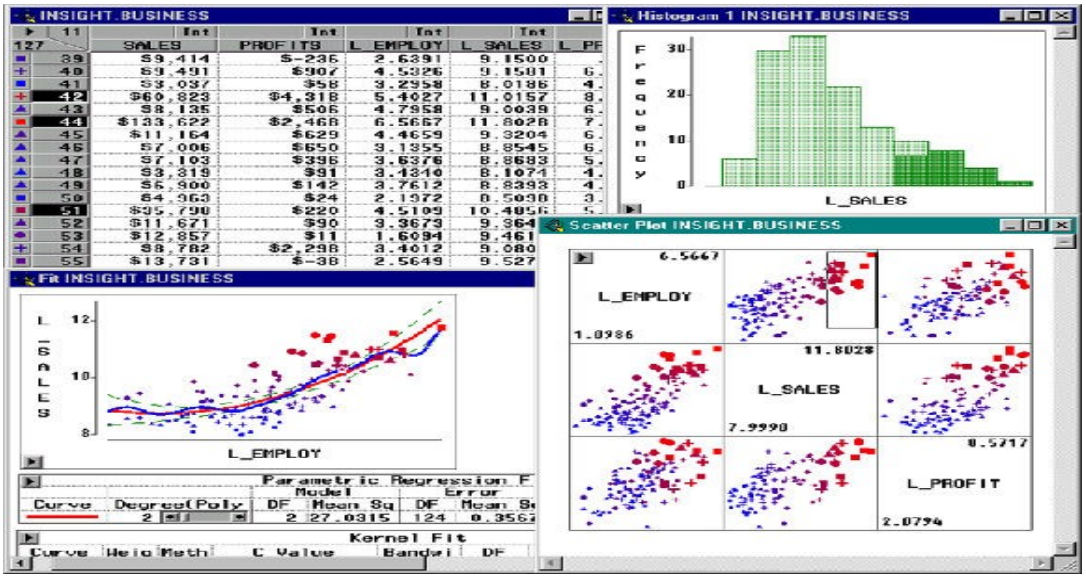


2) Data Mining Result Visualization

- Presentation of the results or knowledge obtained from data mining in visual forms

- Examples
  - Scatter plots and boxplots (obtained from descriptive data mining)
  - Decision trees
  - Association rules
  - Clusters
  - Outliers
  - Generalized rules

Fig 11.2 : Visualization of data mining results in SAS Enterprise Miner: scatter plots



Audio Data Mining

- Uses audio signals to indicate the patterns of data or the features of data mining results
- An interesting alternative to visual mining
- An inverse task of mining audio (such as music) databases which is to find patterns from audio data
- Visual data mining may disclose interesting patterns using graphical displays, but requires users to concentrate on watching patterns

- Instead, transform patterns into sound and music and listen to pitches, rhythms, tune, and melody in order to identify anything interesting or unusual

11.3.4 Data Mining and Collaborative Filtering

A **collaborative filtering** approach is commonly used, in which products are recommended based on the options of other customers.

Fig 11.3: Visualization of association rules in MineSet 3.0

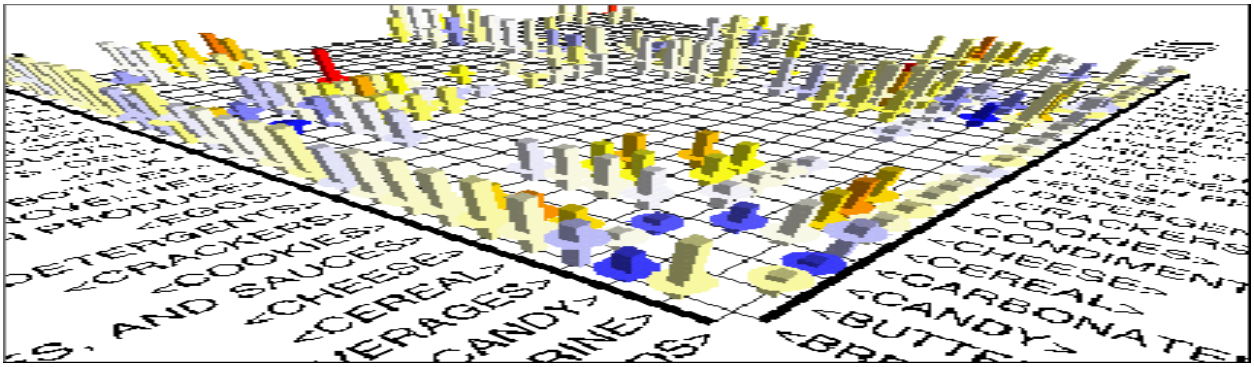


Fig 11.4 : Visualization of a decision tree in MineSet 3.0

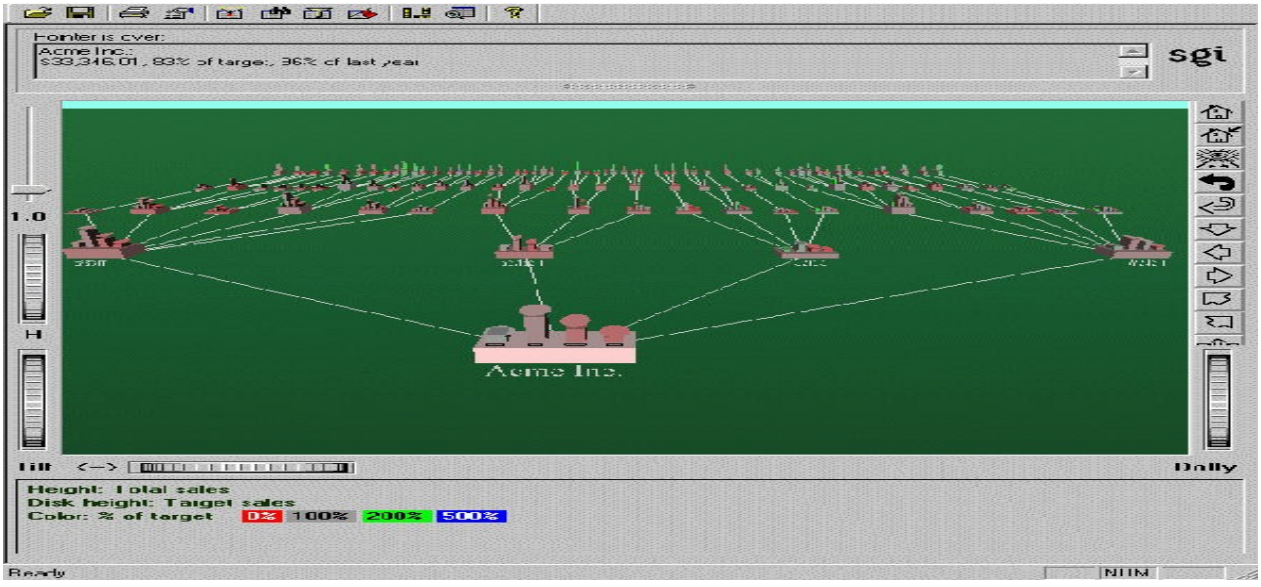
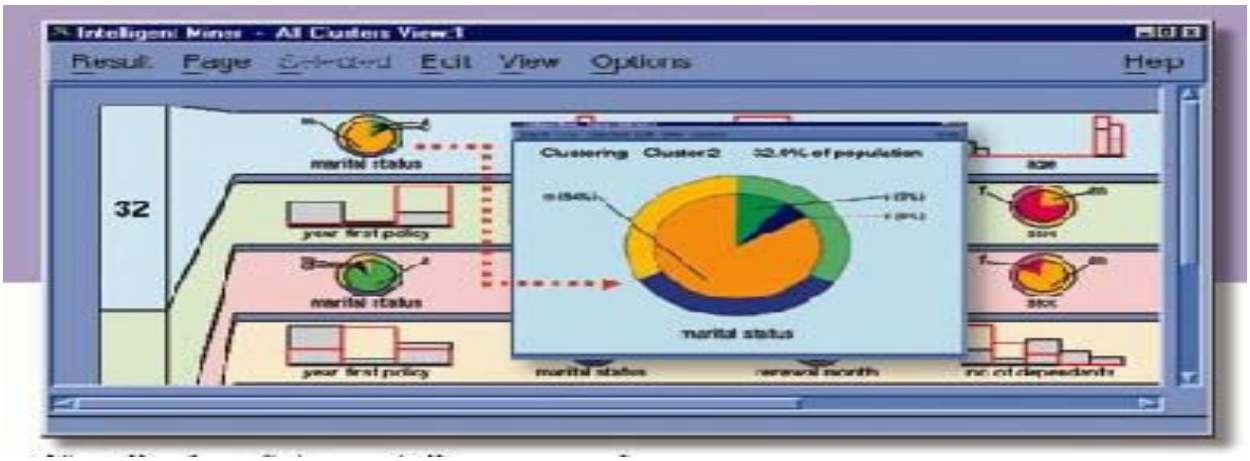


Fig 11.5 : Visualization of cluster groupings in IBM Intelligent Miner

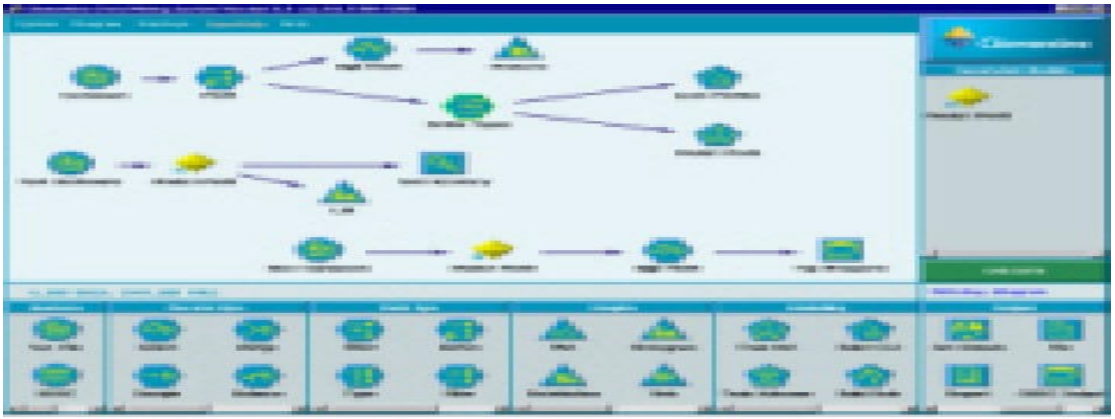


Data Mining Process Visualization

- Presentation of the various processes of data mining in visual forms so that users can see
  - How the data are extracted
  - From which database or data warehouse they are extracted
  - How the selected data are cleaned, integrated, preprocessed, and mined
  - Which method is selected at data mining
  - Where the results are stored

How they may be viewed

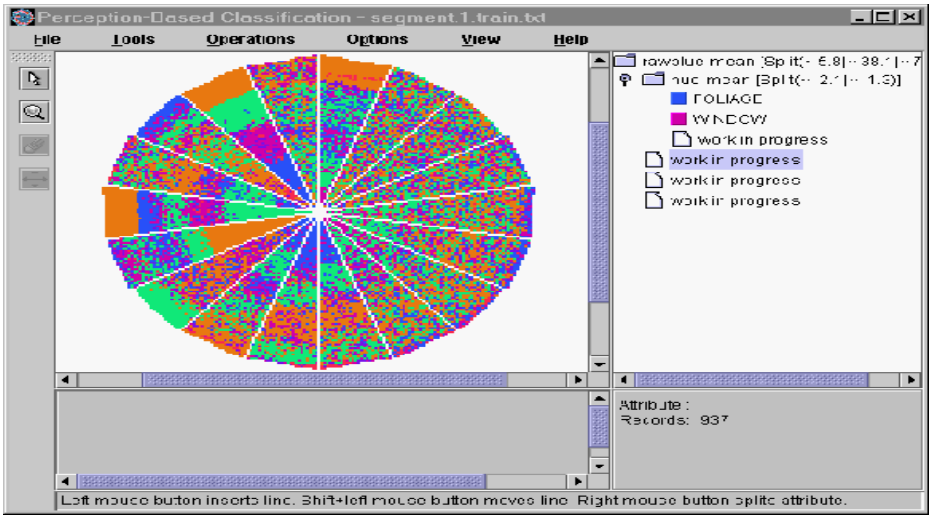
Fig 11.6 : Visualization of Data Mining Processes by Clementine



Interactive Visual Data Mining

- Using visualization tools in the data mining process to help users make smart data mining decisions.
- *Example*
  - Display the data distribution in a set of attributes using colored sectors or columns (depending on whether the whole space is represented by either a circle or a set of columns).
  - Use the display to which sector should first be selected for classification and where a good split point for this sector may be.

Fig 11.7: Perception based classification (PBC): An interactive visual mining approach



11.4 Social impact of data mining

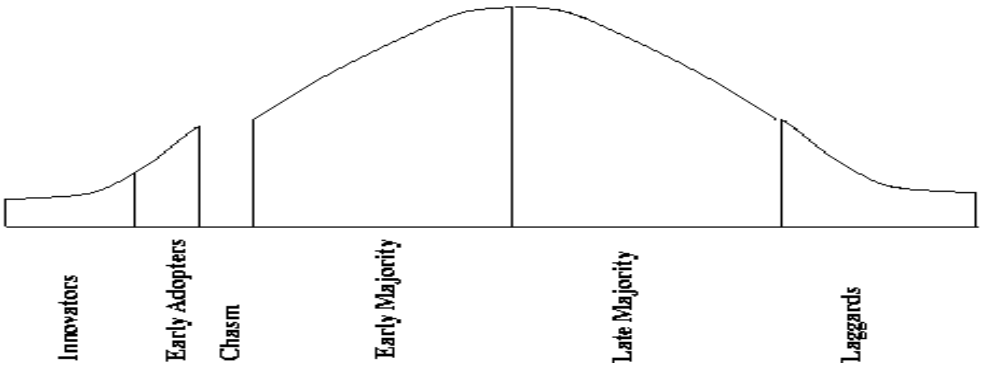
Interactive Visual Mining by Perception-Based Classification (PBC)

Is Data Mining a Hype or Will It Be Persistent?

- Data mining is a technology
- Technological life cycle
  - Innovators
  - Early adopters
  - Chasm

- Early majority
- Late majority

Life Cycle of Technology Adoption



- Data mining is at Chasm!?
  - Existing data mining systems are too generic
  - Need business-specific data mining solutions and smooth integration of business logic with data mining functions

Social Impacts: Threat to Privacy and Data Security?

- Is data mining a threat to privacy and data security?
  - "Big Brother", "Big Banker", and "Big Business" are carefully watching you
  - Profiling information is collected every time
    - You use your credit card, debit card, supermarket loyalty card, or frequent flyer card, or apply for any of the above
    - You surf the Web, reply to an Internet newsgroup, subscribe to a magazine, rent a video, join a club, fill out a contest entry form,
    - You pay for prescription drugs, or present you medical care number when visiting the doctor
  - Collection of personal data may be beneficial for companies and consumers, there is also potential for misuse

Protect Privacy and Data Security

- *Fair information practices*
  - International guidelines for data privacy protection

- Cover aspects relating to data collection, purpose, use, quality, openness, individual participation, and accountability
- Purpose specification and use limitation
- Openness: Individuals have the right to know what information is collected about them, who has access to the data, and how the data are being used

- *Develop and use data security-enhancing techniques*
  - Blind signatures
  - Biometric encryption
  - Anonymous databases

11.5 Trends in Data Mining

1. Application exploration

- development of application-specific data mining system
- Invisible data mining (mining as built-in function)

2. Scalable data mining methods

- Constraint-based mining: use of constraints to guide data mining systems in their search for interesting patterns

3. Integration of data mining with database systems, data warehouse systems, and Web database systems

4. Standardization of data mining language

- A standard will facilitate systematic development, improve interoperability, and promote the education and use of data mining systems in industry and society.

5. *Visual data mining*

6. New methods for mining complex types of data

- More research is required towards the integration of data mining methods with existing data analysis techniques for the complex types of data .

7. *Web mining*

- Privacy protection and information security in data mining.