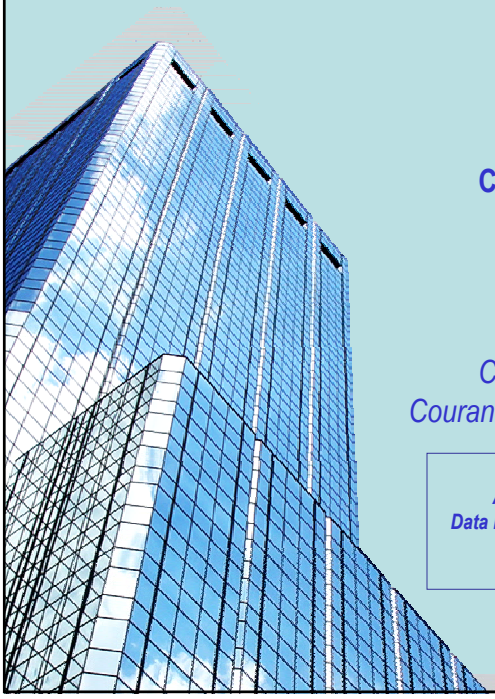


# Data Mining

## Session 7 – Main Theme
## Classification and Prediction


**Dr. Jean-Claude Franchitti**

*New York University*
*Computer Science Department*
*Courant Institute of Mathematical Sciences*


*Adapted from course textbook resources*
*Data Mining Concepts and Techniques (2nd Edition)*
*Jiawei Han and Micheline Kamber*

## Agenda

## What is the class about?

- Course description and syllabus:
  - » http://www.nyu.edu/classes/jcf/g22.3033-002/
  - » http://www.cs.nyu.edu/courses/spring10/G22.3033-002/index.html

- Textbooks:
  - » ***Data Mining: Concepts and Techniques (2nd Edition)***

Jiawei Han, Micheline Kamber
Morgan Kaufmann
ISBN-10: 1-55860-901-6, ISBN-13: 978-1-55860-901-3, (2006)

  - » ***Microsoft SQL Server 2008 Analysis Services Step by Step***

Scott Cameron
Microsoft Press
ISBN-10: 0-73562-620-0, ISBN-13: 978-0-73562-620-31 1st Edition (04/15/09)



## Session Agenda

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

## Icons / Metaphors


Information


Common Realization


Knowledge/Competency Pattern


Governance


Alignment


Solution Approach



## Agenda

1 Session Overview

2 Classification and Prediction

3 Summary and Conclusion

## Classification and Prediction – Sub-Topics

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



## 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

## Classification vs. Numeric Prediction

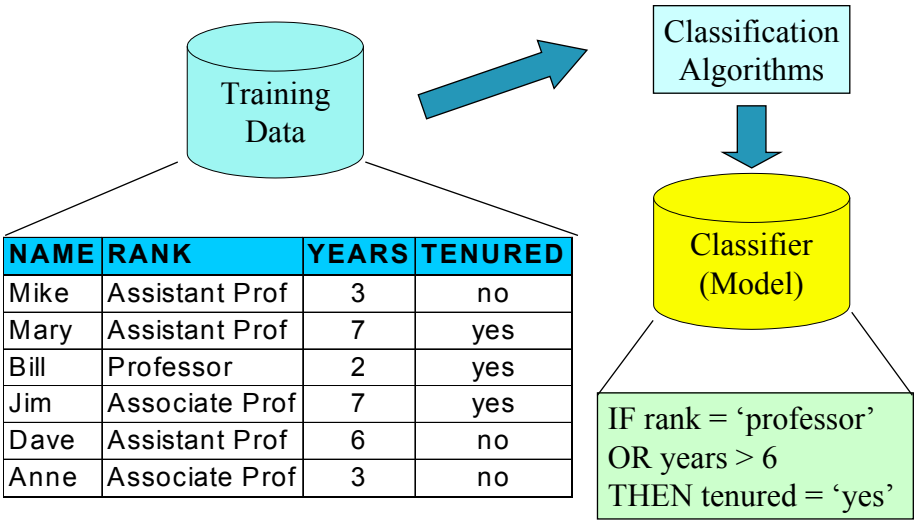
- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is



## Classification—A Two-Step Process

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

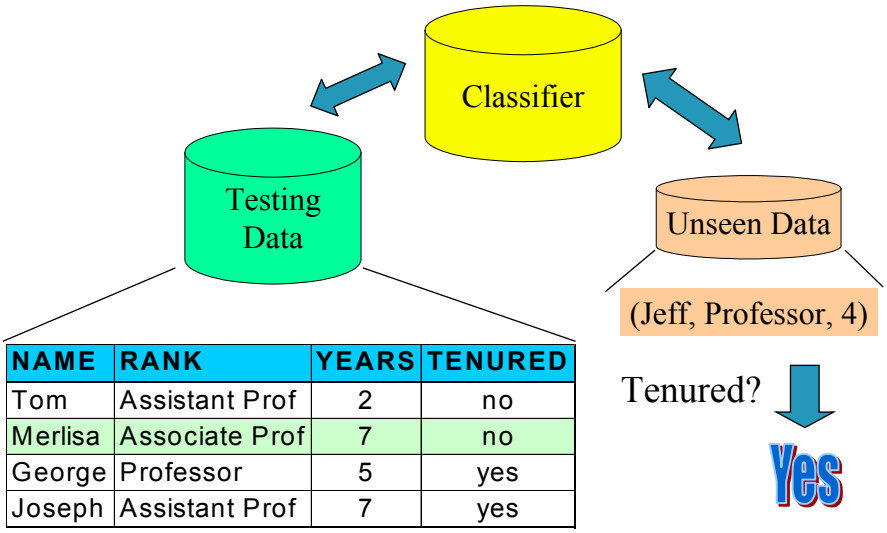
## Process (1): Model Construction




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Mike | Assistant Prof | 3 | no |
| Mary | Assistant Prof | 7 | yes |
| Bill | Professor | 2 | yes |
| Jim | Associate Prof | 7 | yes |
| Dave | Assistant Prof | 6 | no |
| Anne | Associate Prof | 3 | no |



## 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 |

## Classification and Prediction – Sub-Topics

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



## Issues: Data Preparation

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

## Issues: Evaluating Classification Methods

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



## Classification and Prediction – Sub-Topics

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

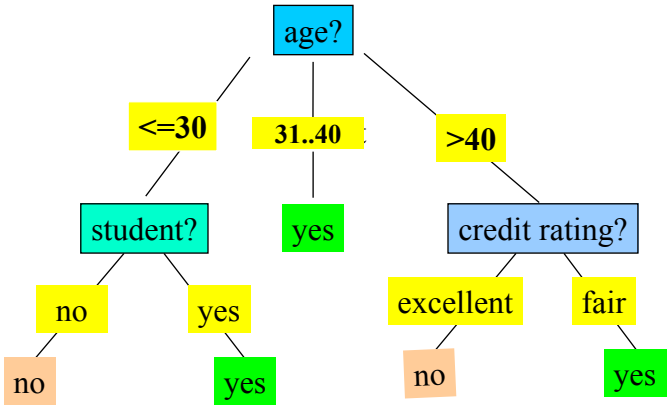
## Decision Tree Induction: Training Dataset

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

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



## Output: A Decision Tree for "*buys_computer*"



## Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - 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



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

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

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

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

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

- Information gained by branching on attribute A

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

## Attribute Selection: Information Gain

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

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

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

| age | $p_i$ | $n_i$ | $I(p_i, n_i)$ |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 31…40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

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

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

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

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$



## Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - » Sort the value A in increasing order
  - » Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of $a_i$ and $a_{i+1}$
  - » The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - » D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

## Gain Ratio for Attribute Selection (C4.5)

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

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

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



## Gini index (CART, IBM IntelligentMiner)

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

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

  where $p_j$ is the relative frequency of class *j* in *D*
- If a data set *D* is split on A into two subsets $D_1$ and $D_2$, the *gini* index *gini*(*D*) is defined as

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

- Reduction in Impurity:

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

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

## Gini index (CART, IBM IntelligentMiner)

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

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

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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$
$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$
$$= 0.450$$
$$= Gini_{income \in \{high\}}(D)$$

but $gini_{\{medium,high\}}$ is 0.30 and thus the best since it is the lowest

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes



## Comparing Attribute Selection Measures

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

## Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to $\chi^2$ distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others



## Overfitting and Tree Pruning

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

## Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - » Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - » Assign the most common value of the attribute
  - » Assign probability to each of the possible values
- Attribute construction
  - » Create new attributes based on existing ones that are sparsely represented
  - » This reduces fragmentation, repetition, and replication



## Classification in Large Databases

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

## Scalable Decision Tree Induction Methods

- SLIQ (EDBT'96 — Mehta et al.)
  - » Builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - » Constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - » Integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - » Builds an AVC-list (attribute, value, class label)
- BOAT (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - » Uses bootstrapping to create several small samples



## Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list**: AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X* )
  - » Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n* )
  - » Set of AVC-sets of all predictor attributes at the node *n*

## Rainforest: Training Set and Its AVC Sets

Training Examples

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

AVC-set on *Age*

| Age | Buy_Computer | |
|---|---|---|
| | yes | no |
| <=30 | 3 | 2 |
| 31..40 | 4 | 0 |
| >40 | 3 | 2 |

AVC-set on *income*

| income | Buy_Computer | |
|---|---|---|
| | yes | no |
| high | 2 | 2 |
| medium | 4 | 2 |
| low | 3 | 1 |

AVC-set on *Student*

| student | Buy_Computer | |
|---|---|---|
| | yes | no |
| yes | 6 | 1 |
| no | 3 | 4 |

AVC-set on *credit_rating*

| Credit rating | Buy_Computer | |
|---|---|---|
| | yes | no |
| fair | 6 | 2 |
| excellent | 3 | 3 |



## Data Cube-Based Decision-Tree Induction

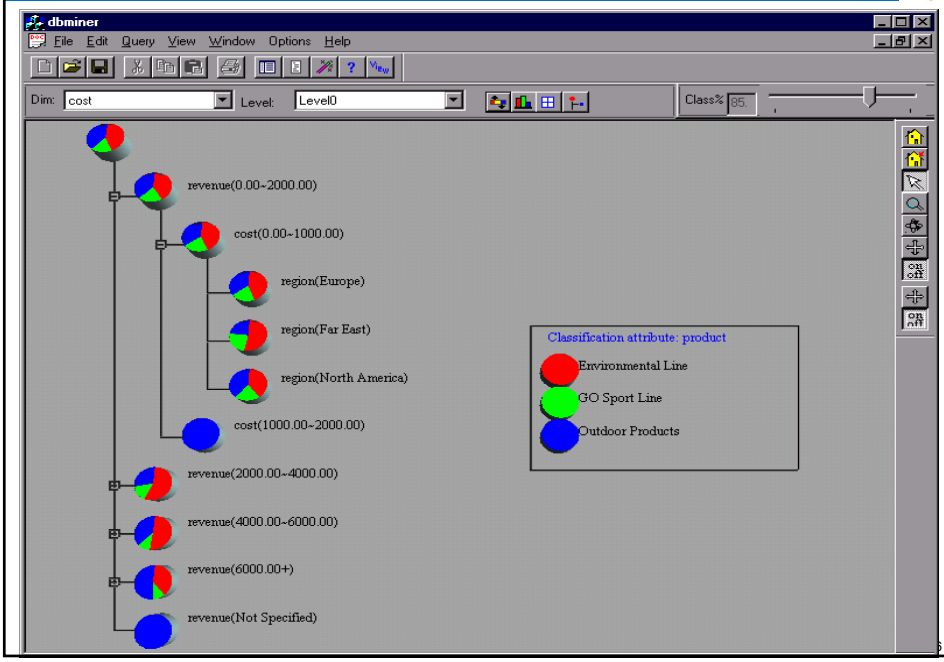
- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level

## BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

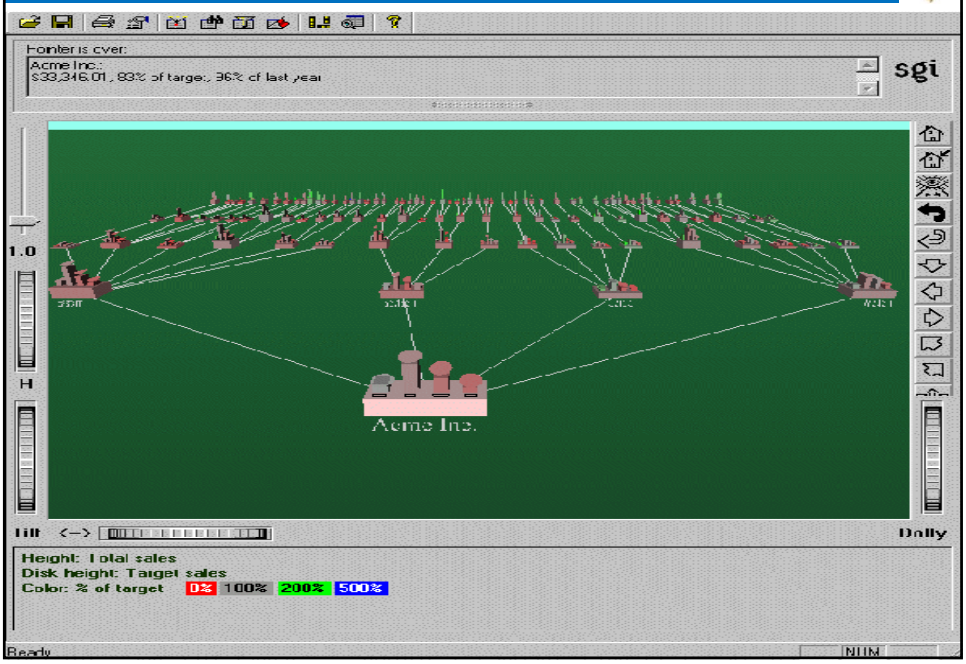
- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - » It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.



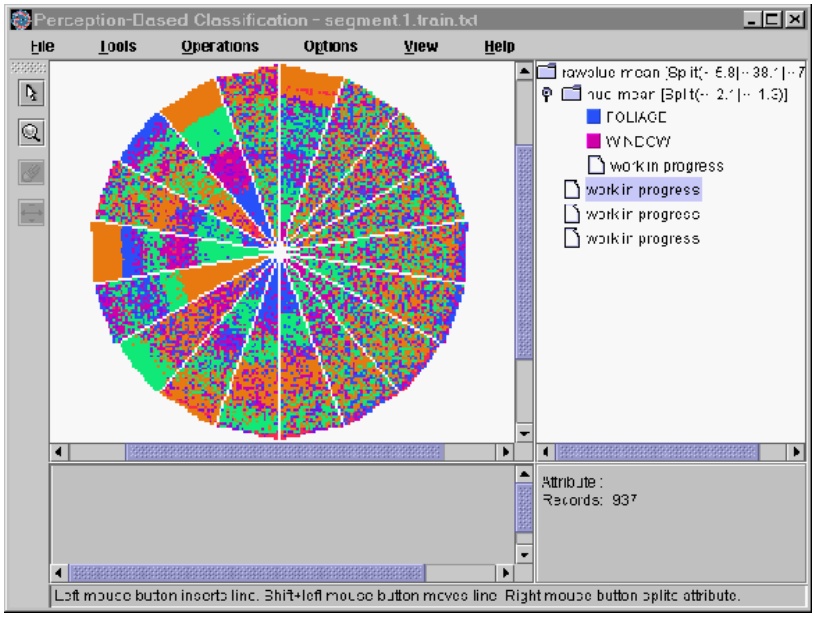
## Presentation of Classification Results



## Visualization of a Decision Tree in SGI/MineSet 3.0




## Interactive Visual Mining by Perception-Based Classification (PBC)



## Classification and Prediction – Sub-Topics

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



## Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

## Bayesian Theorem: Basics

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



## Bayesian Theorem

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

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

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to $C_2$ iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

## Towards Naïve Bayesian Classifier

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

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

- Since P(X) is constant for all classes, only

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

needs to be maximized



## Derivation of Naïve Bayes Classifier

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

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

- This greatly reduces the computation cost: Only counts the class distribution
- If $A_k$ is categorical, $P(x_k|C_i)$ is the # of tuples in $C_i$ having value $x_k$ for $A_k$ divided by $|C_{i,D}|$ (# of tuples of $C_i$ in D)
- If $A_k$ is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

and $P(x_k|C_i)$ is

## Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

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



## Naïve Bayesian Classifier: An Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

**$P(X|C_i)$ :** P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
**$P(X|C_i)*P(C_i)$ :** P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

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

## Avoiding the 0-Probability Problem

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

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

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - » Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - » The "corrected" prob. estimates are close to their "uncorrected" counterparts



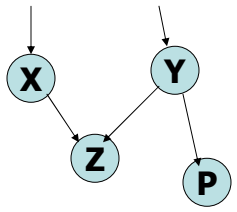
## Naïve Bayesian Classifier: Comments

- Advantages
  - » Easy to implement
  - » Good results obtained in most of the cases
- Disadvantages
  - » Assumption: class conditional independence, therefore loss of accuracy
  - » Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - » Bayesian Belief Networks

## Bayesian Belief Networks

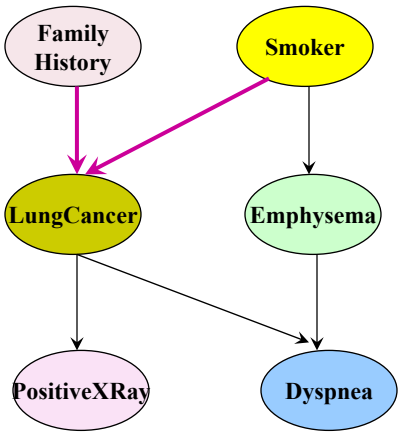
- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - » Represents dependency among the variables
  - » Gives a specification of joint probability distribution




- ❑ 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




**Bayesian Belief Networks**

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:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

## Training Bayesian Networks

- Several scenarios:
  - » Given both the network structure and all variables observable: *learn only the CPTs*
  - » Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - » Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - » Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref. D. Heckerman: Bayesian networks for data mining



## Classification and Prediction – Sub-Topics

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

## 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

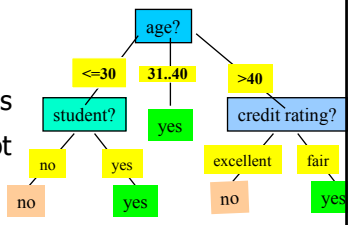
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - $n_{covers}$ = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

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

  accuracy(R) = $n_{correct}$ / $n_{covers}$
- If more than one rule are 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



## 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*

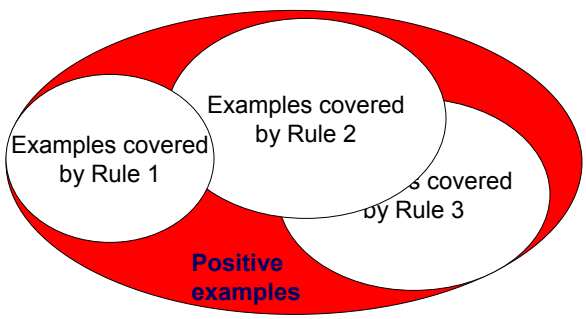
## Rule Induction: Sequential Covering Method

- 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*



## Sequential Covering Algorithm

**while** (enough target tuples left)
generate a rule
remove positive target tuples satisfying this rule



## How to Learn-One-Rule?

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

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

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

- Rule pruning based on an independent set of test tuples

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

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

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



## Rule Generation

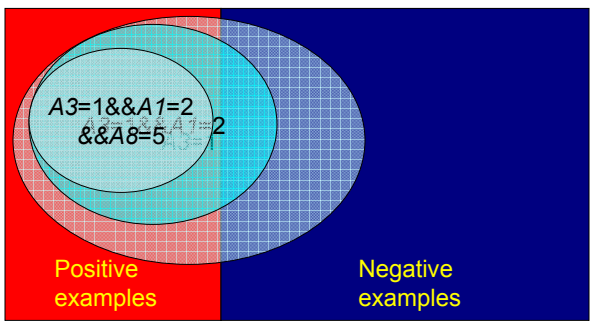
- To generate a rule

**while**(true)

find the best predicate *p*

**if** foil-gain(*p*) > threshold **then** add *p* to current rule

**else** break



## Classification and Prediction – Sub-Topics

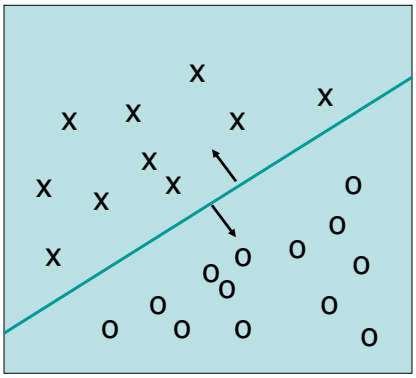
- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your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



## Classification: A Mathematical Mapping

- Classification:
  - predicts categorical class labels
- E.g., Personal homepage classification
  - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or $-1$
  - $x_1$ : # of word "homepage"
  - $x_2$ : # of word "welcome"
- Mathematically
  - $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$
  - We want a function f: $X \rightarrow Y$

## Linear Classification



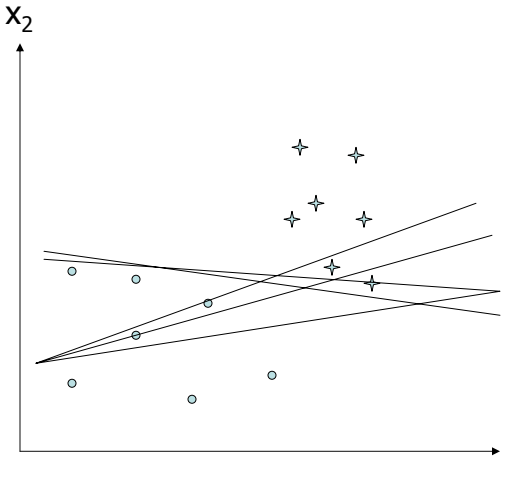

- Binary Classification problem
- The data above the red line belongs to class ‘x’
- The data below red line belongs to class ‘o’
- Examples: SVM, Perceptron, Probabilistic Classifiers



## Discriminative Classifiers

- Advantages
  - » prediction accuracy is generally high
    - As compared to Bayesian methods – in general
  - » robust, works when training examples contain errors
  - » fast evaluation of the learned target function
    - Bayesian networks are normally slow
- Criticism
  - » long training time
  - » difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - » not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

## Perceptron & Winnow




- Vector: x, w
- Scalar: x, y, w

Input: $\{(x_1, y_1), \ldots\}$

Output: classification function f(x)

$f(x_i) > 0$ for $y_i = +1$

$f(x_i) < 0$ for $y_i = -1$

f(x) => $wx + b = 0$

or $w_1x_1+w_2x_2+b = 0$

- Perceptron: update W additively
- Winnow: update W multiplicatively



## Classification by Backpropagation

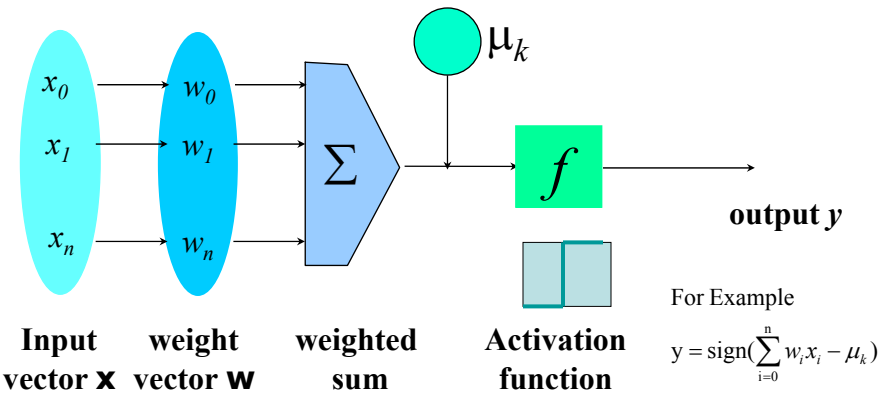
- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, 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

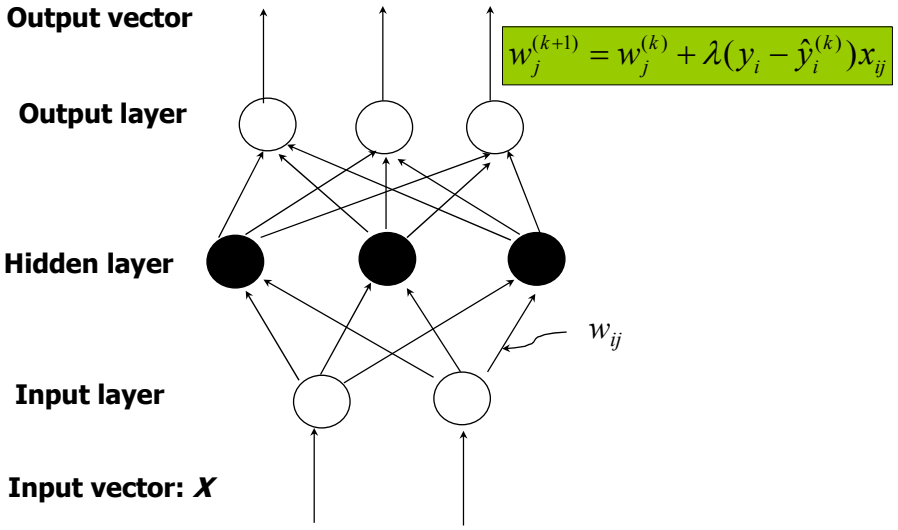


## A Neuron (= a perceptron)




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

## A Multi-Layer Feed-Forward Neural Network






## How A Multi-Layer Neural Network Works?

- 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

## Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*



## 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)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

## Backpropagation and Interpretability

- Efficiency of backpropagation: Each epoch (one interation 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



## Classification and Prediction – Sub-Topics

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

## SVM—Support Vector Machines

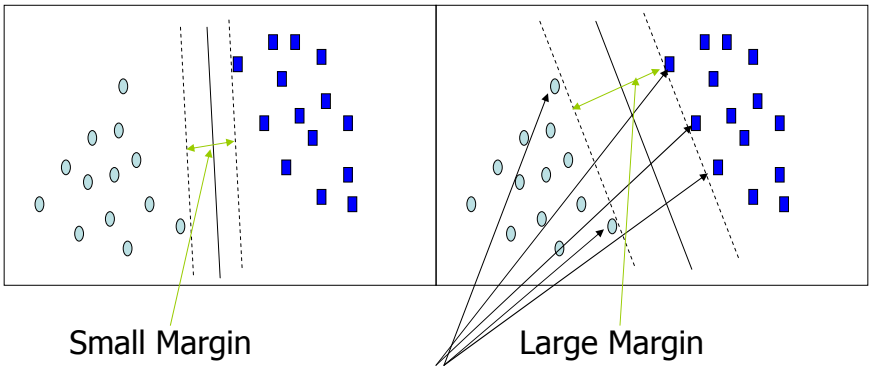
- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)



## SVM—History and Applications

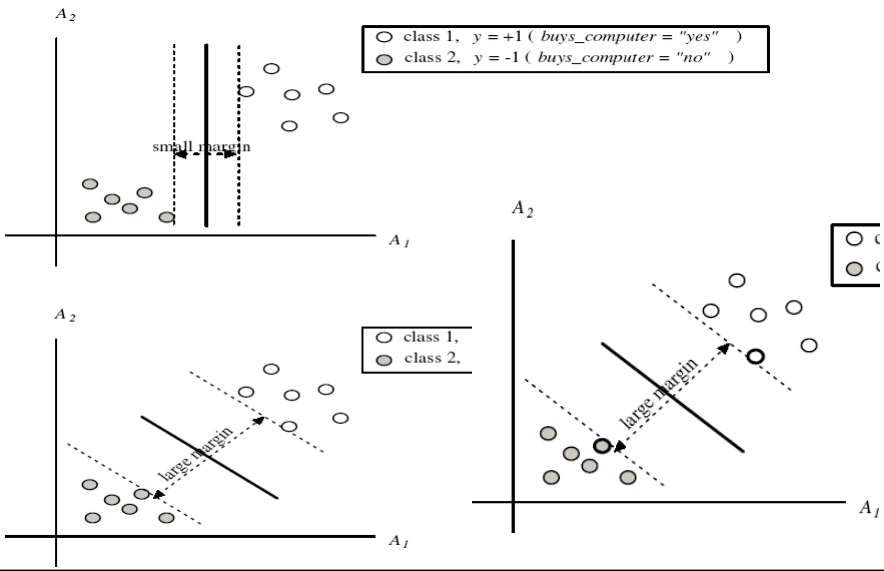
- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
  - » handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

## SVM—General Philosophy

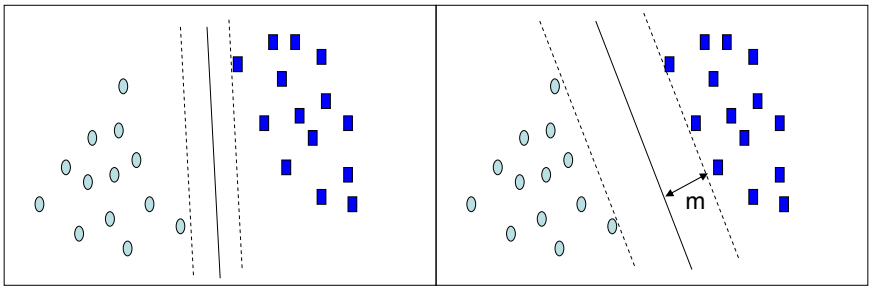





## SVM—Margins and Support Vectors



## SVM—When Data Is Linearly Separable




Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels $y_i$

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)



## SVM—Linearly Separable

- A separating hyperplane can be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ 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 hyperplane 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 hyperplanes $H_1$ 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

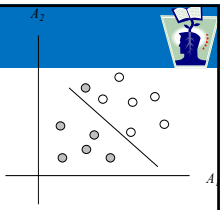
## Why Is SVM Effective on High Dimensional Data?

- 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 hyperplane 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



## SVM—Linearly Inseparable






- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space $Z$ using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and $b$ and then substitute back so that we see that the linear decision hyperplane in the new $(\mathbf{Z})$ space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$ ∎

- Search for a linear separating hyperplane in the new space

## SVM—Kernel functions

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a kernel function $K(\mathbf{X_i}, \mathbf{X_j})$ to the original data, i.e., $K(\mathbf{X_i}, \mathbf{X_j}) = \Phi(\mathbf{X_i})\ \Phi(\mathbf{X_j})$
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters)



## Scaling SVM by Hierarchical Micro-Clustering

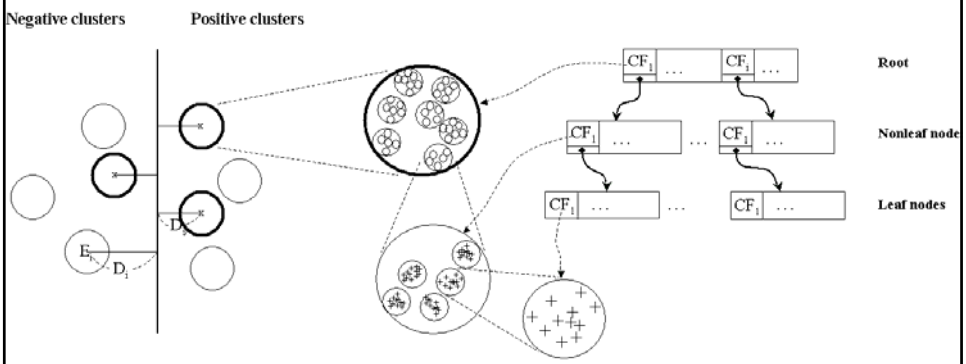
- SVM is not scalable to the number of data objects in terms of training time and memory usage
- "Classifying Large Datasets Using SVMs with Hierarchical Clusters Problem" by Hwanjo Yu, Jiong Yang, Jiawei Han, KDD'03
- CB-SVM (Clustering-Based SVM)
  - Given limited amount of system resources (e.g., memory), maximize the SVM performance in terms of accuracy and the training speed
  - Use micro-clustering to effectively reduce the number of points to be considered
  - At deriving support vectors, de-cluster micro-clusters near "candidate vector" to ensure high classification accuracy

## CB-SVM: Clustering-Based SVM

- Training data sets may not even fit in memory
- Read the data set once (minimizing disk access)
  - » Construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory
  - » The statistical summary maximizes the benefit of learning SVM
- The summary plays a role in indexing SVMs
- Essence of Micro-clustering (Hierarchical indexing structure)
  - » Use micro-cluster hierarchical indexing structure
    - provide finer samples closer to the boundary and coarser samples farther from the boundary
  - » Selective de-clustering to ensure high accuracy



## CF-Tree: Hierarchical Micro-cluster



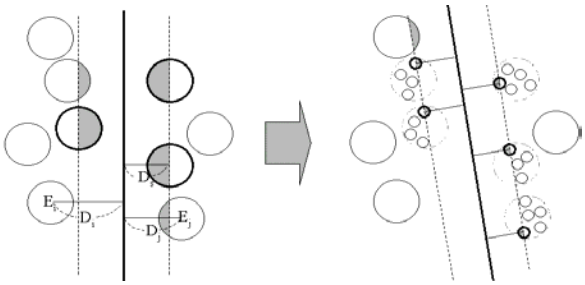
## CB-SVM Algorithm: Outline

- Construct two CF-trees from positive and negative data sets independently
  - » Need one scan of the data set
- Train an SVM from the centroids of the root entries
- De-cluster the entries near the boundary into the next level
  - » The children entries de-clustered from the parent entries are accumulated into the training set with the non-declustered parent entries
- Train an SVM again from the centroids of the entries in the training set
- Repeat until nothing is accumulated

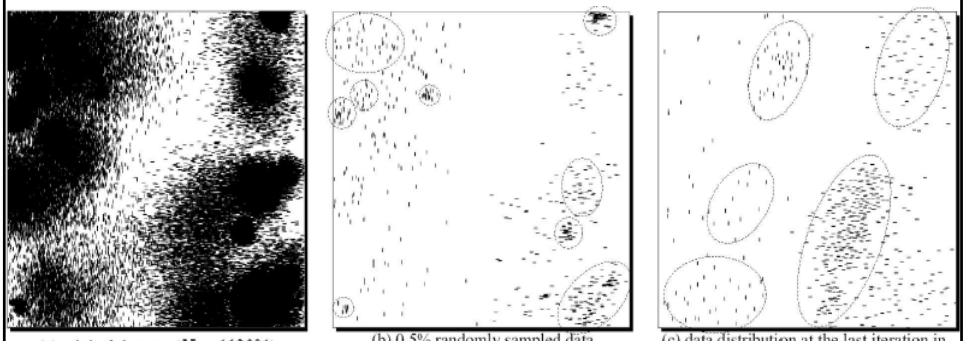


## Selective Declustering

- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster $E_i$ such that
  - » $D_i - R_i < D_s$, where $D_i$ is the distance from the boundary to the center point of $E_i$ and $R_i$ is the radius of $E_i$
  - » Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
    - "Support cluster": The cluster whose centroid is a support vector



## Experiment on Synthetic Dataset



(a) original data set ($N = 113601$)

(b) 0.5% randomly sampled data ($N = 603$)

(c) data distribution at the last iteration in CB-SVM ($N = 597$)

**Figure 6: Synthetic data set in a two-dimensional space.** '|': positive data; '−': negative data



## Experiment on a Large Data Set

| S-Rate | # of data | # of errors | T-Time | S-Time |
|---|---|---|---|---|
| 0.0001% | 23 | 6425 | 0.000114 | 822.97 |
| 0.001% | 226 | 2413 | 0.000972 | 825.40 |
| 0.01% | 2333 | 1132 | 0.03 | 828.61 |
| 0.1% | 23273 | 1012 | 6.287 | 835.87 |
| 1% | 230380 | 1015 | 1192.793 | 838.92 |
| 5% | 1151714 | 1020 | 20705.4 | 842.92 |
| ASVM | 307 | 865 | 54872.213 | |
| CB-SVM | 2893 | 876 | 1.639 | 2528.213 |

**Table 4: Performance results on the very large data set (# of training data = 23066169, # of testing data = 233890).** S-Rate: sampling rate; T-Time: training time; S-Time: sampling time; ASVM: selective sampling

## SVM vs. Neural Network

- SVM
  - » Relatively new concept
  - » Deterministic algorithm
  - » Nice Generalization properties
  - » Hard to learn – learned in batch mode using quadratic programming techniques
  - » Using kernels can learn very complex functions
- Neural Network
  - » Relatively old
  - » Nondeterministic algorithm
  - » Generalizes well but doesn't have strong mathematical foundation
  - » Can easily be learned in incremental fashion
  - » To learn complex functions—use multilayer perceptron (not that trivial)



## SVM Related Links

- SVM Website
  - » http://www.kernel-machines.org/
- Representative implementations
  - » LIBSVM: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - » SVM-light: simpler but performance is not better than LIBSVM, support only binary classification and only C language
  - » SVM-torch: another recent implementation also written in C.

## Notes about SVM - Introductory Literature

- "Statistical Learning Theory" by **Vapnik**: difficult to understand, containing many errors.
- C. J. C. **Burges**. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.
  - » Easier than Vapnik's book, but still not introductory level; the examples are not so intuitive
- The book An Introduction to Support Vector Machines by **Cristianini and Shawe-Taylor**
  - » Not introductory level, but the explanation about Mercer's Theorem is better than above literatures
- Neural Networks and Learning Machines by **Haykin**
  - » Contains a nice chapter on SVM introduction



## Classification and Prediction – Sub-Topics

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

## Lazy vs. Eager Learning

- Lazy vs. eager learning
  - » Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - » Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - » Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - » Eager: must commit to a single hypothesis that covers the entire instance space

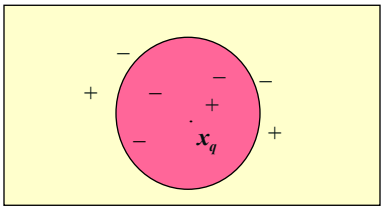


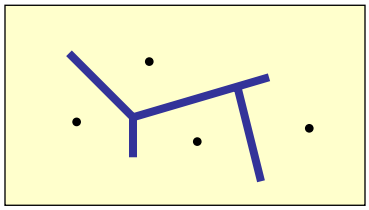
## Lazy Learner: Instance-Based Methods

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

## The *k*-Nearest Neighbor Algorithm

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





## Discussion on the *k*-NN Algorithm

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

## Case-Based Reasoning (CBR)

- CBR: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - » Instances represented by rich symbolic descriptions (e.g., function graphs)
  - » Search for similar cases, multiple retrieved cases may be combined
  - » Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - » Find a good similarity metric
  - » Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases



## Classification and Prediction – Sub-Topics

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

## 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 C4.5



## 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

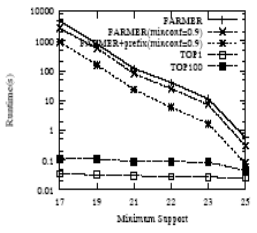
## A Closer Look at CMAR

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
- Efficiency: Uses an enhanced FP-tree that maintains the distribution of class labels among tuples satisfying each frequent itemset
- Rule pruning whenever a rule is inserted into the tree
  - » Given two rules, $R_1$ and $R_2$, if the antecedent of $R_1$ is more general than that of $R_2$ and $conf(R_1) \geq conf(R_2)$, then $R_2$ is pruned
  - » Prunes rules for which the rule antecedent and class are not positively correlated, based on a $\chi^2$ test of statistical significance
- Classification based on generated/pruned rules
  - » If only one rule satisfies tuple X, assign the class label of the rule
  - » If a rule set S satisfies X, CMAR
    - divides S into groups according to class labels
    - uses a weighted $\chi^2$ measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - assigns X the class label of the strongest group

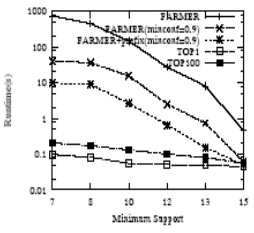


## Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

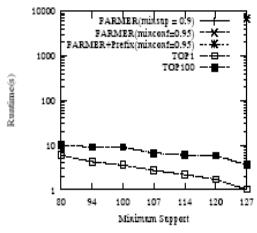
| Dataset | RCBT | CBA | IRG Classifier | C4.5 family | | | SVM |
|---|---|---|---|---|---|---|---|
| | | | | single tree | bagging | boosting | |
| AML/ALL (ALL) | 91.18% | 91.18% | 64.71% | 91.18% | 91.18% | 91.18% | 97.06% |
| Lung Cancer(LC) | 97.99% | 81.88% | 89.93% | 81.88% | 96.64% | 81.88% | 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% |




(a) ALL-AML leukemia




(b) Lung Cancer




(c) Ovarian Cancer

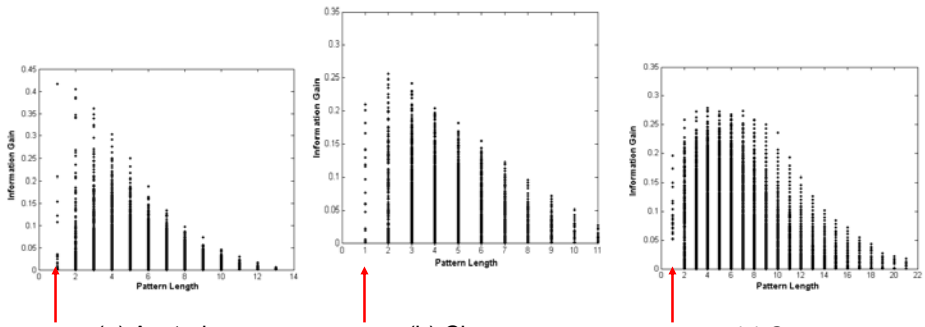
## Frequent Pattern-Based Classification

- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", ICDE'07.
- Accuracy issue
  - » Increase the discriminative power
  - » Increase the expressive power of the feature space
- Scalability issue
  - » It is computationally infeasible to generate all feature combinations and filter them with an information gain threshold
  - » Efficient method (DDPMine: FPtree pruning): H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct Discriminative Pattern Mining for Effective Classification", ICDE'08.



## Frequent Pattern vs. Single Feature

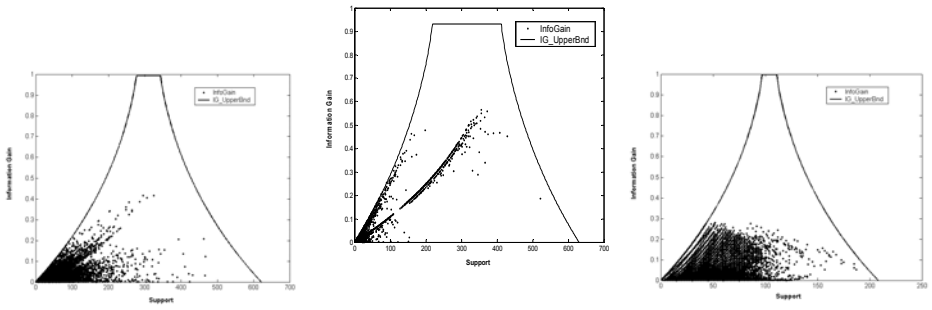
The discriminative power of some frequent patterns is higher than that of single features.




(a) Austral (b) Cleve (c) Sonar

Fig. 1. Information Gain vs. Pattern Length

## Empirical Results




(a) Austral (b) Breast (c) Sonar

Fig. 2. Information Gain vs. Pattern Frequency



## Feature Selection

- Given a set of frequent patterns, both non-discriminative and redundant patterns exist, which can cause overfitting
- We want to single out the discriminative patterns and remove redundant ones
- The notion of Maximal Marginal Relevance (MMR) is borrowed
  - » A document has high marginal relevance if it is both relevant to the query and contains minimal marginal similarity to previously selected documents

## Experimental Results

**Table 1. Accuracy by SVM on Frequent Combined Features vs. Single Features**

| Data | Single Feature | | | Freq. Pattern | |
|---|---|---|---|---|---|
| | *Item_All* | *Item_FS* | *Item_RBF* | *Pat_All* | *Pat_FS* |
| anneal | **99.78** | **99.78** | 99.11 | 99.33 | 99.67 |
| austral | 85.01 | 85.50 | 85.01 | 81.79 | **91.14** |
| auto | 83.25 | 84.21 | 78.80 | 74.97 | **90.79** |
| breast | 97.46 | 97.46 | 96.98 | 96.83 | **97.78** |
| cleve | 84.81 | 84.81 | 85.80 | 78.55 | **95.04** |
| diabetes | 74.41 | 74.41 | 74.55 | 77.73 | **78.31** |
| glass | 75.19 | 75.19 | 74.78 | 79.91 | **81.32** |
| heart | 84.81 | 84.81 | 84.07 | 82.22 | **88.15** |
| hepatic | 84.50 | 89.04 | 85.83 | 81.29 | **96.83** |
| horse | 83.70 | 84.79 | 82.36 | 82.35 | **92.39** |
| iono | 93.15 | 94.30 | 92.61 | 89.17 | **95.44** |
| iris | 94.00 | **96.00** | 94.00 | 95.33 | **96.00** |
| labor | 89.99 | 91.67 | 91.67 | 94.99 | **95.00** |
| lymph | 81.00 | 81.62 | 84.29 | 83.67 | **96.67** |
| pima | 74.56 | 74.56 | 76.15 | 76.43 | **77.16** |
| sonar | 82.71 | 86.55 | 82.71 | 84.60 | **90.86** |
| vehicle | 70.43 | 72.93 | 72.14 | 73.33 | **76.34** |
| wine | 98.33 | 99.44 | 98.33 | 98.30 | **100** |
| zoo | 97.09 | 97.09 | 95.09 | 94.18 | **99.00** |

**Table 2. Accuracy by C4.5 on Frequent Combined Features vs. Single Features**

| Dataset | Single Features | | Frequent Patterns | |
|---|---|---|---|---|
| | *Item_All* | *Item_FS* | *Pat_All* | *Pat_FS* |
| anneal | 98.33 | 98.33 | 97.22 | **98.44** |
| austral | 84.53 | 84.53 | 84.21 | **88.24** |
| auto | 71.70 | 77.63 | 71.14 | **78.77** |
| breast | 95.56 | 95.56 | 95.40 | **96.35** |
| cleve | 80.87 | 80.87 | 80.84 | **91.42** |
| diabetes | **77.02** | **77.02** | 76.00 | 76.58 |
| glass | 75.24 | 75.24 | 76.62 | **79.89** |
| heart | 81.85 | 81.85 | 80.00 | **86.30** |
| hepatic | 78.79 | 85.21 | 80.71 | **93.04** |
| horse | 83.71 | 83.71 | 84.50 | **87.77** |
| iono | 92.30 | 92.30 | 92.89 | **94.87** |
| iris | **94.00** | **94.00** | 93.33 | 93.33 |
| labor | 86.67 | 86.67 | **95.00** | 91.67 |
| lymph | 76.95 | 77.62 | 74.90 | **83.67** |
| pima | 75.86 | 75.86 | 76.28 | **76.72** |
| sonar | 80.83 | 81.19 | **83.67** | **83.67** |
| vehicle | 70.70 | 71.49 | **74.24** | 73.06 |
| wine | 95.52 | 93.82 | 96.63 | **99.44** |
| zoo | 91.18 | 91.18 | 95.09 | **97.09** |



## Scalability Tests

**Table 3. Accuracy & Time on Chess Data**

| *min_sup* | #Patterns | Time (s) | SVM (%) | C4.5 (%) |
|---|---|---|---|---|
| 1 | N/A | N/A | N/A | N/A |
| 2000 | 68,967 | 44.703 | 92.52 | 97.59 |
| 2200 | 28,358 | 19.938 | 91.68 | 97.84 |
| 2500 | 6,837 | 2.906 | 91.68 | 97.62 |
| 2800 | 1,031 | 0.469 | 91.84 | 97.37 |
| 3000 | 136 | 0.063 | 91.90 | 97.06 |

**Table 4. Accuracy & Time on Waveform Data**

| *min_sup* | #Patterns | Time (s) | SVM (%) | C4.5 (%) |
|---|---|---|---|---|
| 1 | 9,468,109 | N/A | N/A | N/A |
| 80 | 26,576 | 176.485 | 92.40 | 88.35 |
| 100 | 15,316 | 90.406 | 92.19 | 87.29 |
| 150 | 5,408 | 23.610 | 91.53 | 88.80 |
| 200 | 2,481 | 8.234 | 91.22 | 87.32 |

## DDPMine: Branch-and-Bound Search

$$\sup(child) \le \sup(parent)$$

$$\sup(b) \le \sup(a)$$

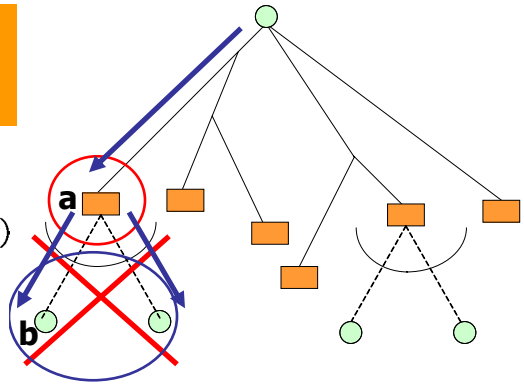
maximize $IG(C|b)$

subject to

$min\_sup \le sup(b) \le sup(a)$

$0 \le sup_+(b) \le sup_+(a)$

$0 \le sup_-(b) \le sup_-(a)$



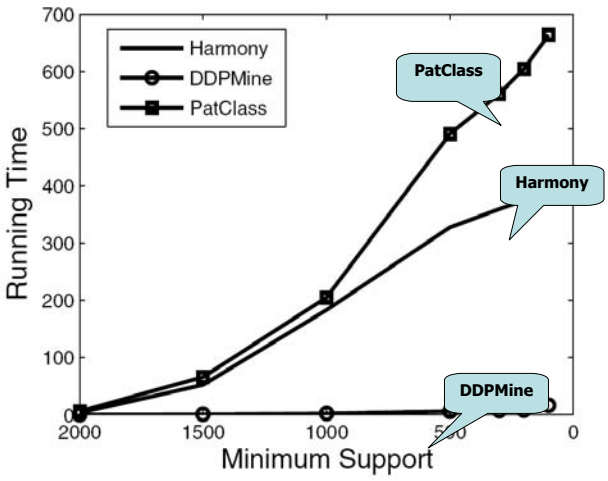

**a: constant, a parent node**

**b: variable, a descendent**

**Association between information gain and frequency**



## DDPMine Efficiency: Runtime




PatClass: ICDE'07 Pattern Classification Alg.

## Classification and Prediction – Sub-Topics

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

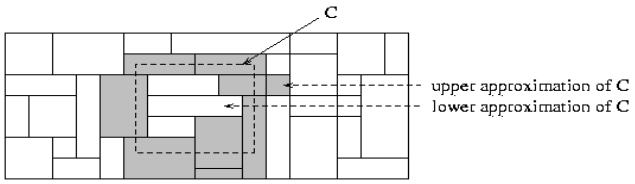


## Genetic Algorithms (GA)

- 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 offsprings
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offsprings 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

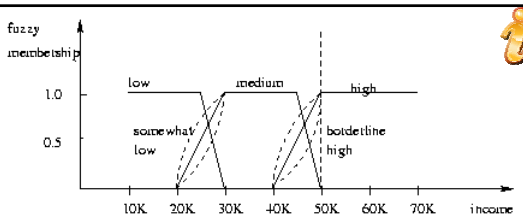
## Rough Set Approach

- 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






## 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

## Classification and Prediction – Sub-Topics

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



## What Is 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

## 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

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1)$, $(\mathbf{X_2}, y_2)$,…, $(\mathbf{X_{|D|}}, y_{|D|})$
  - 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



## 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

## Other Regression-Based Models

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of y is a function of the mean value of y, not a constant
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels



## Regression Trees and Model Trees

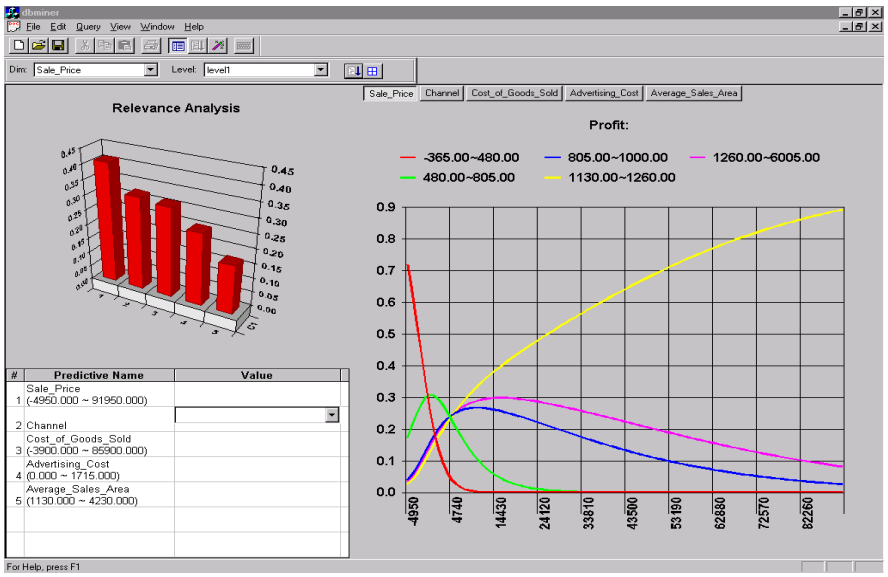
- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a *continuous-valued prediction*
  - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

## Predictive Modeling in Multidimensional Databases

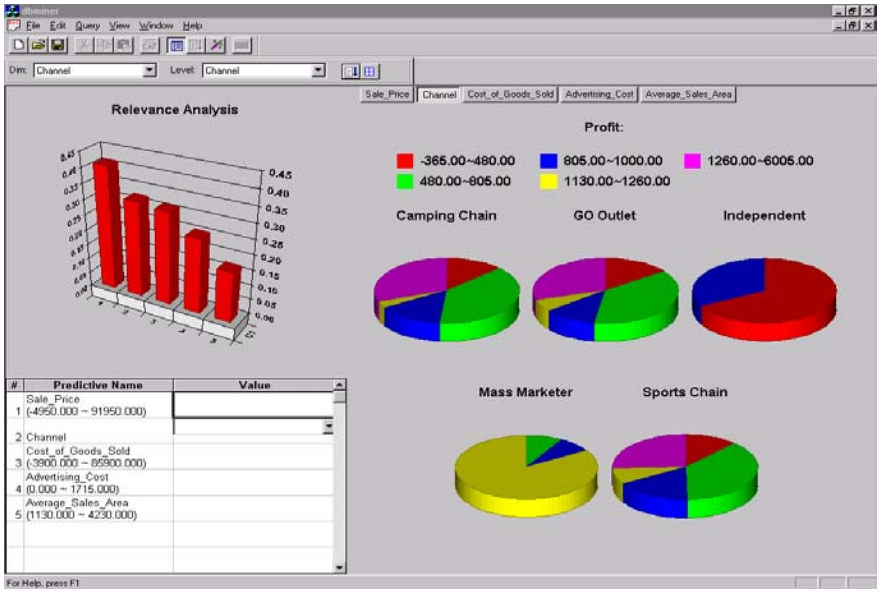
- Predictive modeling: Predict data values or construct generalized linear models based on the database data
- One can only predict value ranges or category distributions
- Method outline:
  - » Minimal generalization
  - » Attribute relevance analysis
  - » Generalized linear model construction
  - » Prediction
- Determine the major factors which influence the prediction
  - » Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis



## Prediction: Numerical Data



## Prediction: Categorical Data






## Classification and Prediction – Sub-Topics

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

## Classifier Accuracy Measures

| Real class\Predicted class | $C_1$ | $\sim C_1$ |
|---|---|---|
| $C_1$ | True positive | False negative |
| $\sim C_1$ | False positive | True negative |

| Real class\Predicted class | buy_computer = yes | buy_computer = no | total | recognition(%) |
|---|---|---|---|---|
| buy_computer = yes | 6954 | 46 | 7000 | 99.34 |
| buy_computer = no | 412 | 2588 | 3000 | 86.27 |
| total | 7366 | 2634 | 10000 | 95.52 |

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given *m* classes, $CM_{i,j}$, an entry in a **confusion matrix**, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
- Alternative accuracy measures (e.g., for cancer diagnosis)

  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
  - This model can also be used for cost-benefit analysis



## Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error betw. $y_i$ and the predicted value $y_i'$
  - Absolute error: $| y_i – y_i' |$
  - Squared error: $(y_i – y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{d}$ Mean squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$
  - Relative absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{\sum_{i=1}^{d} | y_i - \bar{y} |}$ Relative squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{\sum_{i=1}^{d} (y_i - \bar{y})^2}$

  The mean squared-error exaggerates the presence of outliers

  Popularly use (square) root mean-square error, similarly, root relative squared error

## Evaluating the Accuracy of a Classifier or Predictor (I)

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- Cross-validation (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use $D_i$ as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data



## Evaluating the Accuracy of a Classifier or Predictor (II)

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several boostrap methods, and a common one is **.632 boostrap**
  - Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedue k times, overall accuracy of the model:

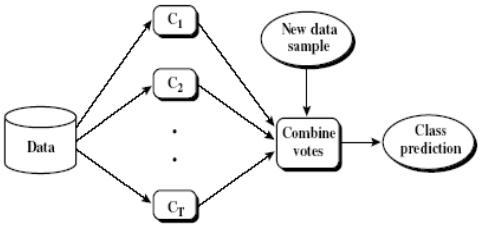
$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

## Classification and Prediction – Sub-Topics

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



## Ensemble Methods: Increasing the Accuracy




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

## Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set $D_i$ of *d* tuples is sampled with replacement from D (i.e., boostrap)
  - A classifier model $M_i$ is learned for each training set $D_i$
- Classification: classify an unknown sample **X**
  - Each classifier $M_i$ returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction



## Boosting

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

## Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, $y_1$), …, ($\mathbf{X_d}$, $y_d$)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set $D_i$ of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model $M_i$ is derived from $D_i$
  - Its error rate is calculated using $D_i$ as a test set
  - If a tuple is misclssified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier $M_i$'s vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$



## Classification and Prediction – Sub-Topics

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

## Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - » Holdout method, random subsampling
  - » Cross-validation
  - » Bootstrap
- Comparing classifiers:
  - » Confidence intervals
  - » Cost-benefit analysis and ROC Curves



## Classifier Evaluation Metrics: Accuracy & Error Rate

**Confusion Matrix:**

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

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

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

**Error rate:** *1 – accuracy*, or

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

## Classifier Evaluation Metrics: Example - Confusion Matrix

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

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



## Classifier Evaluation Metrics: Sensitivity and Specificity

- **Class Imbalance Problem**:
  - one class may be *rare*, e.g. fraud detection data, medical data
  - significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate,

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

**Specificity**: True Negative recognition rate,

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

## Classifier Evaluation Metrics: Precision and Recall

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

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

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

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

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



## Classifier Evaluation Metrics: Example

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

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

## Classifier Evaluation Metrics: *F* and $F_ß$ Measures

- ***F* measure (*$F_1$* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***$F_ß$*:**
  - » weighted measure of precision and recall
  - » assigns ß times as much weight to recall as to precision,

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$



## Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - » Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - » Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - » Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - » At *i*-th iteration, use $D_i$ as test set and others as training set
  - » Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - » ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

## Evaluating the Classifier Accuracy: Bootstrap

- **Bootstrap**
  - » Works well with small data sets
  - » Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - » A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - » Repeat the sampling procedure *k* times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$



## Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

- Suppose we have 2 classifiers, $M_1$ and $M_2$. Which is best?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the 2 error rates is just attributed to *chance*?
  - » Use a **test of statistical significance**
  - » Obtain **confidence limits** for our error estimates

## Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: $M_1$ & $M_2$ are the same, i.e. $|\overline{err}(M_1) - \overline{err}(M_2)$ = 0
- If we can **reject** null hypothesis, then
  - » conclude that the difference between $M_1$ & $M_2$ is **statistically significant**
  - » Chose model with lower error rate



## Estimating Confidence Intervals: t-test

- If only 1 test set available:**pairwise comparison**
  - » For i[th] round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - » Average over 10 rounds to get $\overline{err}(M_1)$ and $\overline{err}(M_2)$
  - » **t-test** computes **t-statistic** with *k-1* **degrees of freedom:**

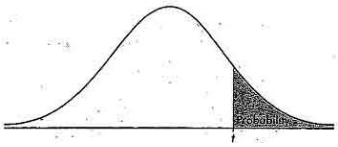
$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}, \quad \text{where}$$

- $$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))\right]^2$$

where $$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

where $k_1$ & $k_2$ are # of cross-validation samples used for $M_1$ & $M_2$, resp.

## Estimating Confidence Intervals: Table for t-distribution




- **Significance level**, e.g, *sig = 0.05* or *5%* means $M_1$ & $M_2$ are *significantly different* for 95% of population
- **Confidence limit**, *z = sig/2*

**TABLE B: *t*-DISTRIBUTION CRITICAL VALUES**

| | Tail probability *p* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| df | .25 | .20 | .15 | .10 | .05 | .025 | .02 | .01 | .005 | .0025 | .001 | .0005 |
| 1 | 1.000 | 1.376 | 1.963 | 3.078 | 6.314 | 12.71 | 15.89 | 31.82 | 63.66 | 127.3 | 318.3 | 636.6 |
| 2 | .816 | 1.061 | 1.386 | 1.886 | 2.920 | 4.303 | 4.849 | 6.965 | 9.925 | 14.09 | 22.33 | 31.60 |
| 3 | .765 | .978 | 1.250 | 1.638 | 2.353 | 3.182 | 3.482 | 4.541 | 5.841 | 7.453 | 10.21 | 12.92 |
| 4 | .741 | .941 | 1.190 | 1.533 | 2.132 | 2.776 | 2.999 | 3.747 | 4.604 | 5.598 | 7.173 | 8.610 |
| 5 | .727 | .920 | 1.156 | 1.476 | 2.015 | 2.571 | 2.757 | 3.365 | 4.032 | 4.773 | 5.893 | 6.869 |
| 6 | .718 | .906 | 1.134 | 1.440 | 1.943 | 2.447 | 2.612 | 3.143 | 3.707 | 4.317 | 5.208 | 5.959 |
| 7 | .711 | .896 | 1.119 | 1.415 | 1.895 | 2.365 | 2.517 | 2.998 | 3.499 | 4.029 | 4.785 | 5.408 |
| 8 | .706 | .889 | 1.108 | 1.397 | 1.860 | 2.306 | 2.449 | 2.896 | 3.355 | 3.833 | 4.501 | 5.041 |
| 9 | .703 | .883 | 1.100 | 1.383 | 1.833 | 2.262 | 2.398 | 2.821 | 3.250 | 3.690 | 4.297 | 4.781 |
| 10 | .700 | .879 | 1.093 | 1.372 | 1.812 | 2.228 | 2.359 | 2.764 | 3.169 | 3.581 | 4.144 | 4.587 |
| 11 | .697 | .876 | 1.088 | 1.363 | 1.796 | 2.201 | 2.328 | 2.718 | 3.106 | 3.497 | 4.025 | 4.437 |
| 12 | .695 | .873 | 1.083 | 1.356 | 1.782 | 2.179 | 2.303 | 2.681 | 3.055 | 3.428 | 3.930 | 4.318 |
| 13 | .694 | .870 | 1.079 | 1.350 | 1.771 | 2.160 | 2.282 | 2.650 | 3.012 | 3.372 | 3.852 | 4.221 |
| 14 | .692 | .868 | 1.076 | 1.345 | 1.761 | 2.145 | 2.264 | 2.624 | 2.977 | 3.326 | 3.787 | 4.140 |
| 15 | .691 | .866 | 1.074 | 1.341 | 1.753 | 2.131 | 2.249 | 2.602 | 2.947 | 3.286 | 3.733 | 4.073 |
| 16 | .690 | .865 | 1.071 | 1.337 | 1.746 | 2.120 | 2.235 | 2.583 | 2.921 | 3.252 | 3.686 | 4.015 |
| 17 | .689 | .863 | 1.069 | 1.333 | 1.740 | 2.110 | 2.224 | 2.567 | 2.898 | 3.222 | 3.646 | 3.965 |
| 18 | .688 | .862 | 1.067 | 1.330 | 1.734 | 2.101 | 2.214 | 2.552 | 2.878 | 3.197 | 3.611 | 3.922 |
| 19 | .688 | .861 | 1.066 | 1.328 | 1.729 | 2.093 | 2.205 | 2.539 | 2.861 | 3.174 | 3.579 | 3.883 |
| 20 | .687 | .860 | 1.064 | 1.325 | 1.725 | 2.086 | 2.197 | 2.528 | 2.845 | 3.153 | 3.552 | 3.850 |
| 21 | .686 | .859 | 1.063 | 1.323 | 1.721 | 2.080 | 2.189 | 2.518 | 2.831 | 3.135 | 3.527 | 3.819 |
| 22 | .686 | .858 | 1.061 | 1.321 | 1.717 | 2.074 | 2.183 | 2.508 | 2.819 | 3.119 | 3.505 | 3.792 |
| 23 | .685 | .858 | 1.060 | 1.319 | 1.714 | 2.069 | 2.177 | 2.500 | 2.807 | 3.104 | 3.485 | 3.768 |
| 24 | .685 | .857 | 1.059 | 1.318 | 1.711 | 2.064 | 2.172 | 2.492 | 2.797 | 3.091 | 3.467 | 3.745 |
| 25 | .684 | .856 | 1.058 | 1.316 | 1.708 | 2.060 | 2.167 | 2.485 | 2.787 | 3.078 | 3.450 | 3.725 |
| 26 | .684 | .856 | 1.058 | 1.315 | 1.706 | 2.056 | 2.162 | 2.479 | 2.779 | 3.067 | 3.435 | 3.707 |
| 27 | .684 | .855 | 1.057 | 1.314 | 1.703 | 2.052 | 2.158 | 2.473 | 2.771 | 3.057 | 3.421 | 3.690 |
| 28 | .683 | .855 | 1.056 | 1.313 | 1.701 | 2.048 | 2.154 | 2.467 | 2.763 | 3.047 | 3.408 | 3.674 |
| 29 | .683 | .854 | 1.055 | 1.311 | 1.699 | 2.045 | 2.150 | 2.462 | 2.756 | 3.038 | 3.396 | 3.659 |
| 30 | .683 | .854 | 1.055 | 1.310 | 1.697 | 2.042 | 2.147 | 2.457 | 2.750 | 3.030 | 3.385 | 3.646 |
| 40 | .681 | .851 | 1.050 | 1.303 | 1.684 | 2.021 | 2.123 | 2.423 | 2.704 | 2.971 | 3.307 | 3.551 |
| 50 | .679 | .849 | 1.047 | 1.299 | 1.676 | 2.009 | 2.109 | 2.403 | 2.678 | 2.937 | 3.261 | 3.496 |
| 60 | .679 | .848 | 1.045 | 1.296 | 1.671 | 2.000 | 2.099 | 2.390 | 2.660 | 2.915 | 3.232 | 3.460 |
| 80 | .678 | .846 | 1.043 | 1.292 | 1.664 | 1.990 | 2.088 | 2.374 | 2.639 | 2.887 | 3.195 | 3.416 |
| 100 | .677 | .845 | 1.042 | 1.290 | 1.660 | 1.984 | 2.081 | 2.364 | 2.626 | 2.871 | 3.174 | 3.390 |
| 1000 | .675 | .842 | 1.037 | 1.282 | 1.646 | 1.962 | 2.056 | 2.330 | 2.581 | 2.813 | 3.098 | 3.300 |
| ∞ | .674 | .841 | 1.036 | 1.282 | 1.645 | 1.960 | 2.054 | 2.326 | 2.576 | 2.807 | 3.091 | 3.291 |
| | 50% | 60% | 70% | 80% | 90% | 95% | 96% | 98% | 99% | 99.5% | 99.8% | 99.9% |
| | Confidence level *C* | | | | | | | | | | | |

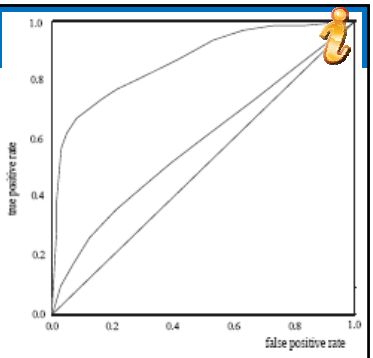


## Estimating Confidence Intervals: Statistical Significance

- Are $M_1$ & $M_2$ **significantly different**?
  - » Compute *t*. Select *significance level* (e.g. *sig = 5%)*
  - » Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - » t-distribution is symmetric – typically upper % points of distribution shown → look up value for **confidence limit** *z=sig/2* (here, 0.025)
  - » **If t > z or t < -z**, then t value lies in rejection region:
    - **Reject null hypothesis** that mean error rates of $M_1$ & $M_2$ are same
    - Conclude: statistically significant difference between $M_1$ & $M_2$
  - » **Otherwise**, conclude that any difference is **chance**.

## Model Selection: ROC Curves

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



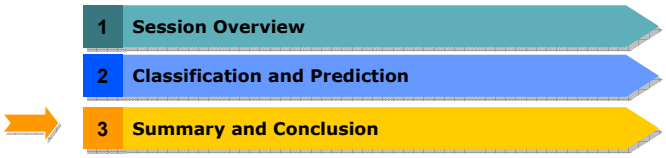

- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0



## Issues Affecting Model Selection

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

## Agenda






## Handling Different Kinds of Cases in Classification (under construction!)

- **Class Imbalance Problem:** Consider sensitivity and specificity measures.
- **Multiclass problem**: Instead of assigning a class label, assign a probability class distribution
- **Active learning**
- **Transfer learning**

## Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes
- Effective and scalable methods have been developed for Bayesian belief network, backpropagation, Support Vector Machine (SVM), pattern-based classification, nearest neighbor classifiers, and case-based reasoning, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches
- Unlike naive Bayesian classification (which assumes class conditional independence), **Bayesian belief networks** allow class conditional independencies to be defined between subsets of variables



## Summary (II)

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

## Summary (III)

- **Backpropagation:**
  - Neural network algorithm that uses gradient descent
  - Searches for a set of *weights* that model the data so as to minimize the error between the network's class prediction and the actual class label of data tuples
  - Rules can be extracted for improved interpretability
- **Support Vector Machine (SVM):**
  - For classification of both linear and nonlinear data
  - Transforms original data into a higher dimension, from where it finds a hyperplane for separation of the data using essential training tuples called **support vectors**.
- **Pattern-Based Classification:**
  - Uses association mining techniques that search for frequently occurring patterns in large databases.
  - The patterns may generate rules, which can be analyzed for use in classification.



## Summary (IV)

- **Lazy Learners:**
  - store all training tuples and wait until presented with a test tuple before performing generalization.
  - *k*-nearest neighbor and case-based reasoning
- **Genetic Algorithms**: populations of rules “evolve” via operations of crossover and mutation until all rules within a population satisfy specified threshold.
- **Rough Set Approach:** approximately define classes that are not distinguishable based on the available attributes
- **Fuzzy Set Approaches:** replace ``brittle'' threshold cutoffs for continuous-valued attributes with degree of membership functions.

## References (1)


- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997.
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984.
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998.
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95.
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, **Discriminative Frequent Pattern Analysis for Effective Classification**, ICDE'07.
- H. Cheng, X. Yan, J. Han, and P. S. Yu, **Direct Discriminative Pattern Mining for Effective Classification**, ICDE'08.
- W. Cohen. **Fast effective rule induction**. ICML'95.
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05.




## References (2)


- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

## References (3)


- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96.
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report**. ECML'93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96.




## References (4)


- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules**. SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters**. KDD'03.

## Assignments & Readings

- Readings

  - Chapter 6
- Individual Project #1
  - Due 03/25/10
- Group Project Proposal
  - Due 03/25/10



## Next Session: Cluster Analysis