



CS 412 Intro. to Data Mining

Chapter 8. Classification: Basic Concepts

Jiawei Han, Computer Science, Univ. Illinois at Urbana-Champaign, 2106





Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary



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

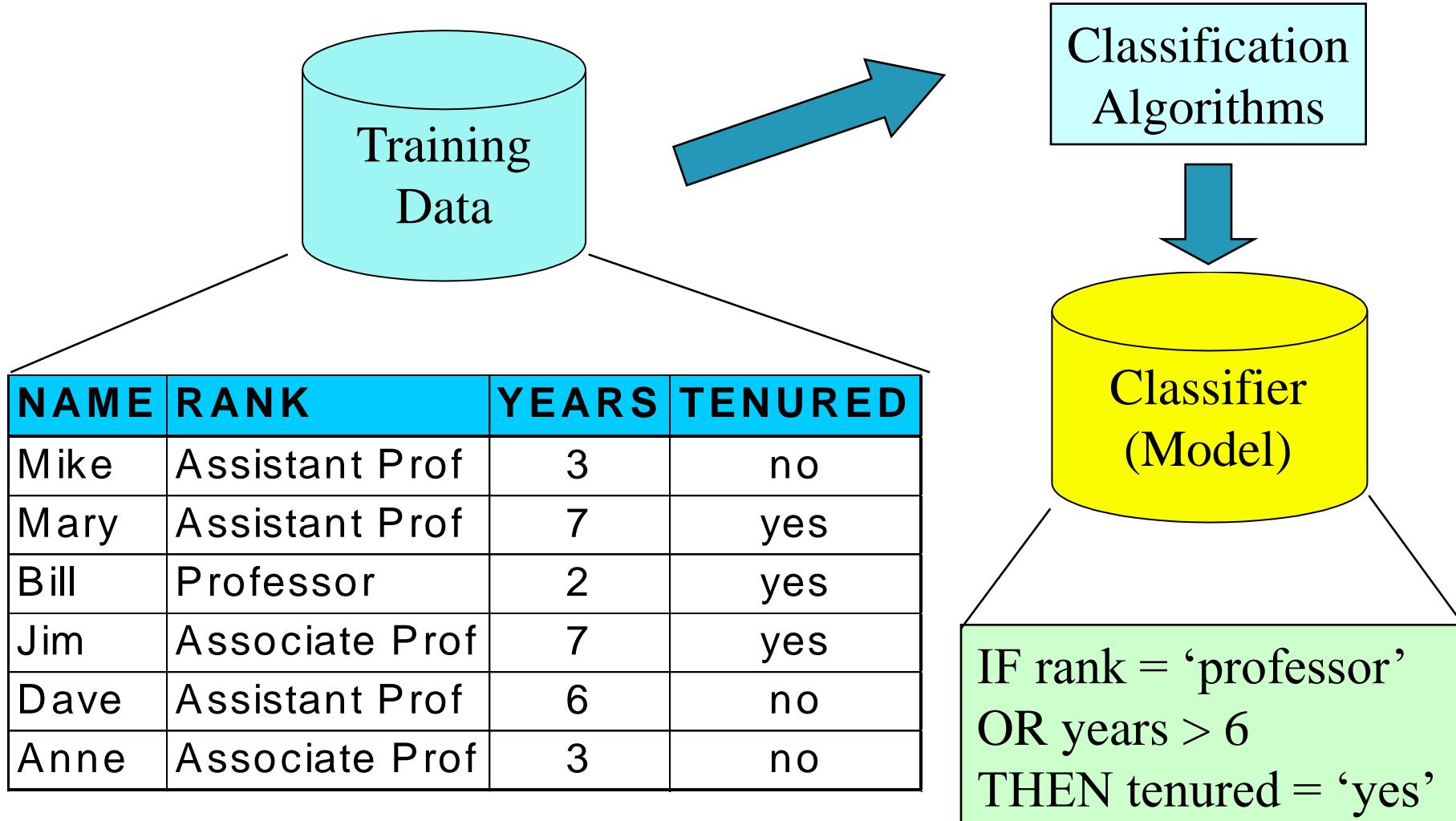
Prediction Problems: Classification vs. Numeric Prediction

- 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

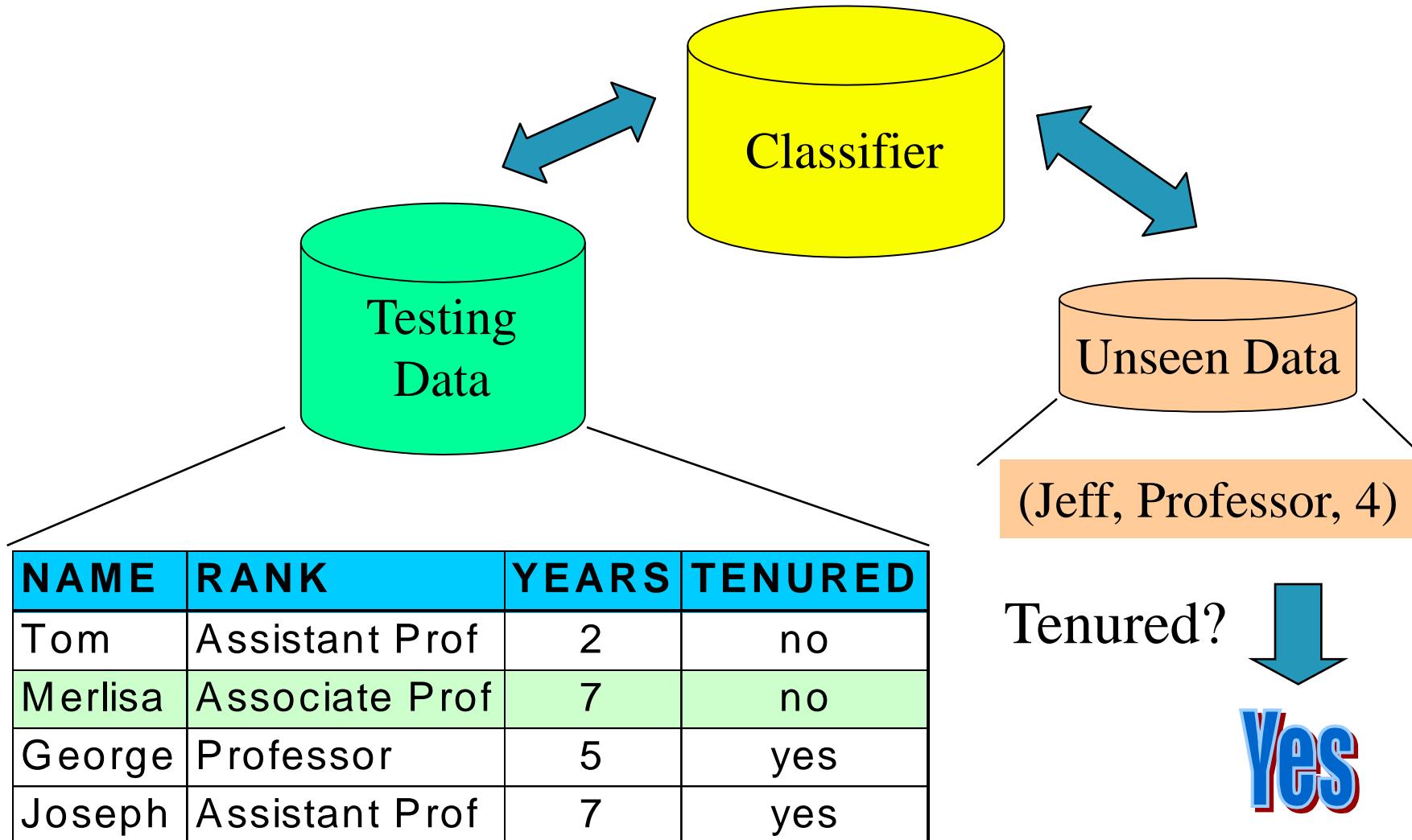
Classification—A Two-Step Process

- **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**
 - Model: 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:** % 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 new data
 - Note: If *the test set* is used to select/refine models, it is called **validation (test) set** or **development test set**

Process (1): Model Construction



Process (2): Using the Model in Prediction



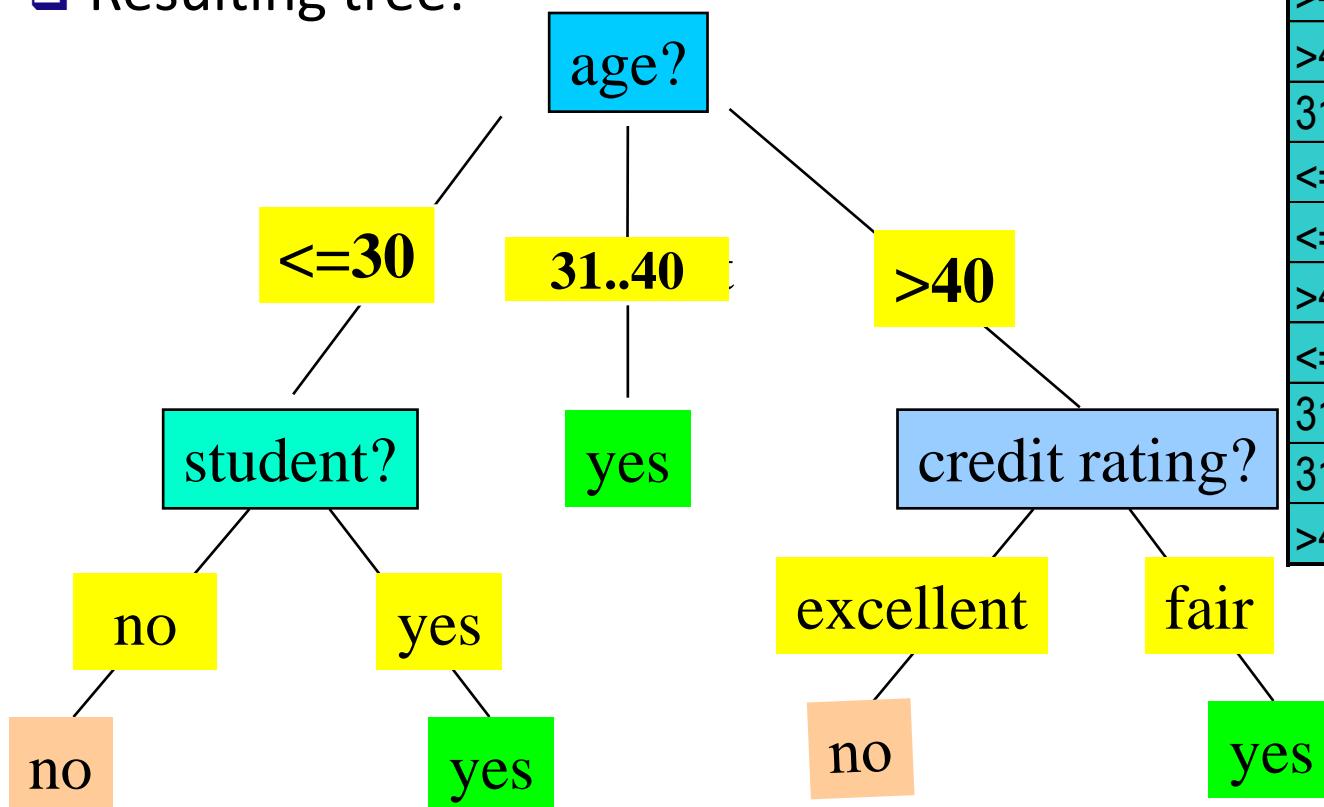
Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary



Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



| 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 |

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

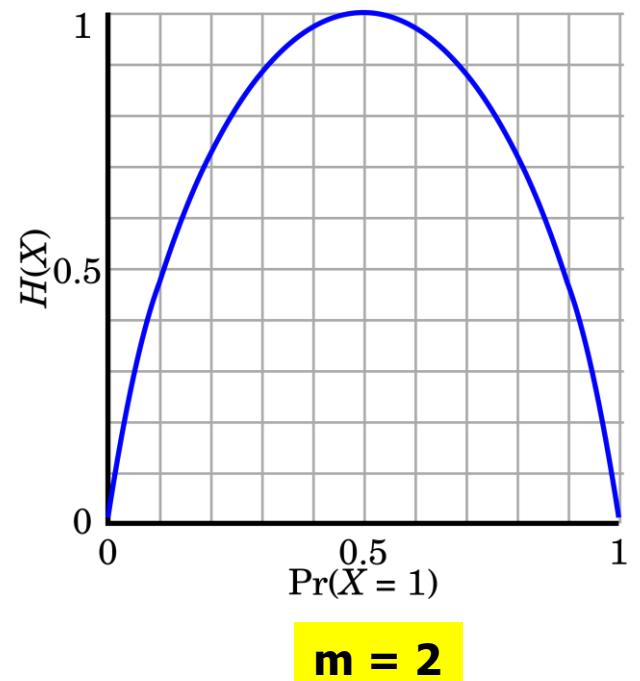
Brief Review of Entropy

- Entropy (Information Theory)
 - A measure of uncertainty associated with a random number
 - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, y_2, \dots, y_m\}$

$$H(Y) = - \sum_{i=1}^m p_i \log(p_i) \text{ where } p_i = P(Y = y_i)$$

- Interpretation
 - Higher entropy \rightarrow higher uncertainty
 - Lower entropy \rightarrow lower uncertainty
- Conditional entropy

$$H(Y|X) = \sum_x p(x) H(Y|X = x)$$



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:

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

- 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 |

$\frac{5}{14} I(2,3)$ means “age ≤ 30 ” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

Hence

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

| age | income | student | credit_rating | buys_computer |
|-----------|--------|---------|---------------|---------------|
| ≤ 30 | high | no | fair | no |
| ≤ 30 | high | no | excellent | no |
| 31...40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31...40 | low | yes | excellent | yes |
| ≤ 30 | medium | no | fair | no |
| ≤ 30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| ≤ 30 | medium | yes | excellent | yes |
| 31...40 | medium | no | excellent | yes |
| 31...40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

Computing Information-Gain for Continuous-Valued 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 $A \leq \text{split-point}$, and D2 is the set of tuples in D satisfying $A > \text{split-point}$

Gain Ratio for Attribute Selection (C4.5)

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

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

- GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$
- $gain_ratio(income) = 0.029/1.557 = 0.019$
- 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*)

Computation of Gini Index

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

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

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

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

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

- 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
 - **Gini 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 χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^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”

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 is decision tree induction popular?
 - 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
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
 - Builds an AVC-list (attribute, value, class label)

RainForest: A Scalable Classification Framework

- The criteria that determine the quality of the tree can be computed separately
 - 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

| age | income | student | credit_rating | buy_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 |

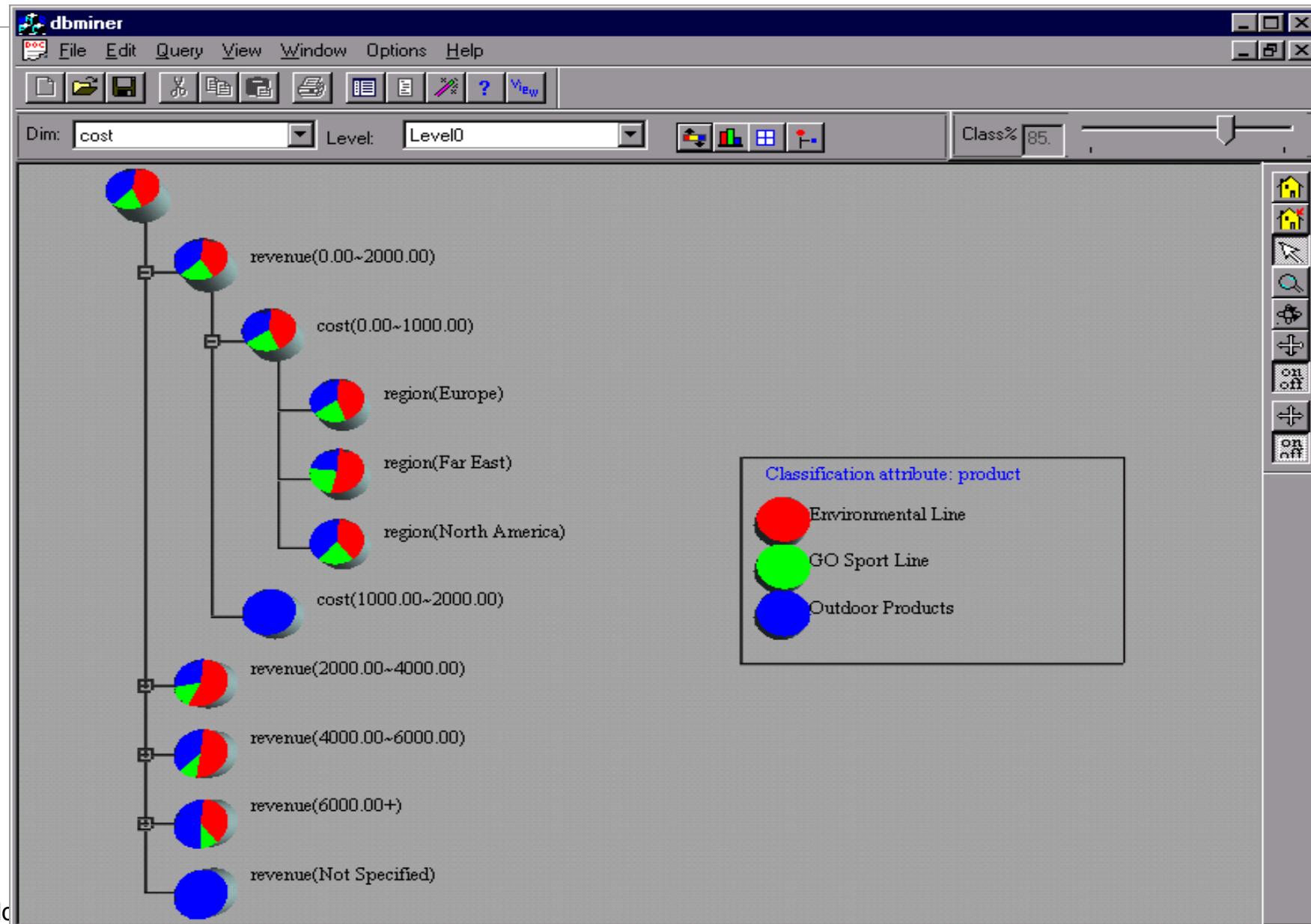
The Training Data

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

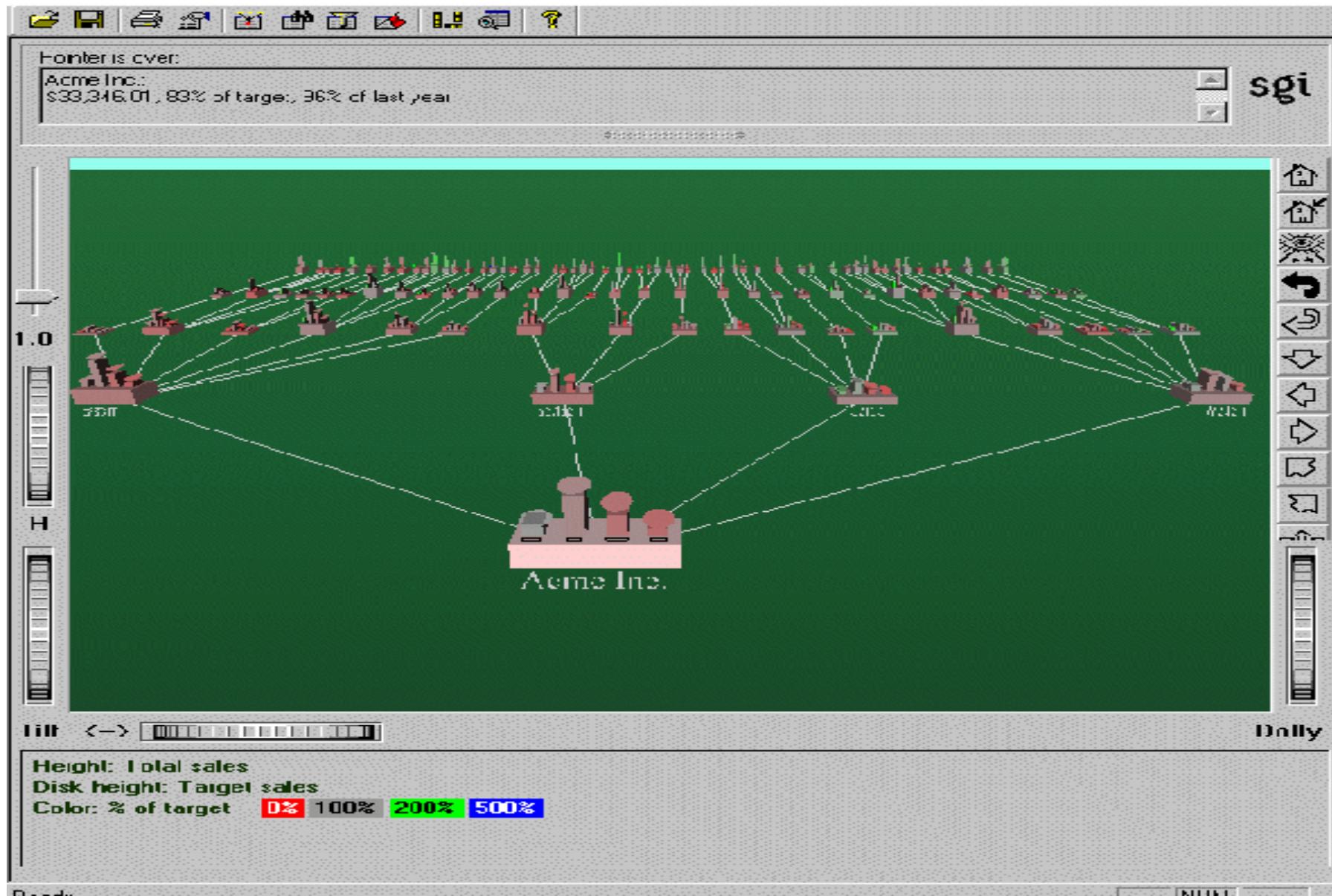
| AVC-set on Student | | AVC-set on Credit_Rating | |
|--------------------|--------------|--------------------------|--------------|
| student | Buy_Computer | | Buy_Computer |
| | yes | no | |
| yes | 6 | 1 | 6 |
| no | 3 | 4 | 2 |

Its AVC Sets

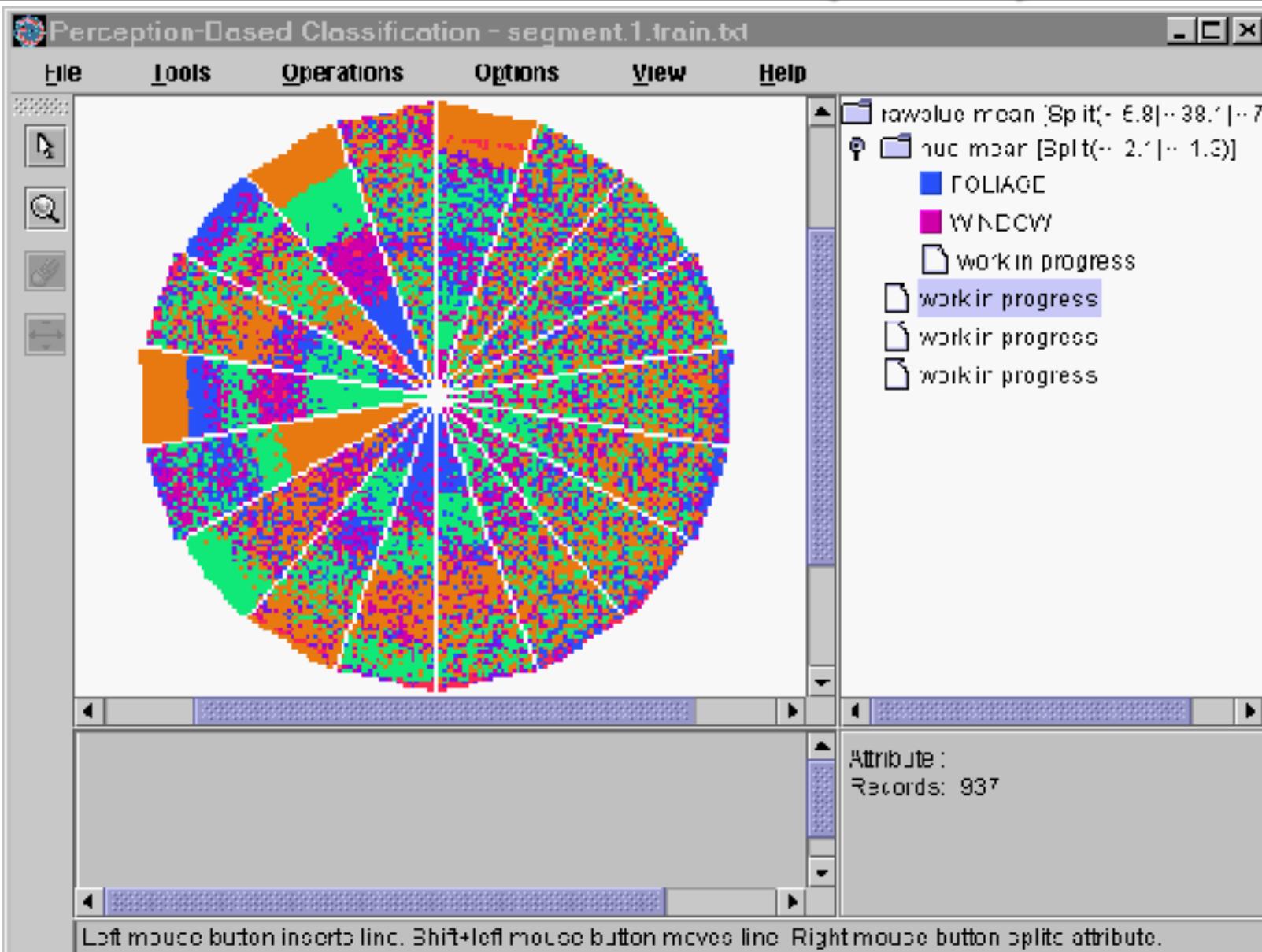
Presentation of Classification Results



Visualization of a Decision Tree in SGI/MineSet 3.0



Interactive Visual Mining by Perception-Based Classification (PBC)



Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods 
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Bayesian Classification: Why?

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

Bayes' Theorem: Basics

- Total probability Theorem:
$$P(B) = \sum_{i=1}^M P(B|A_i)P(A_i)$$
- Bayes' Theorem:
$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$
 - Let \mathbf{X} be a data sample ("evidence"): class label is unknown
 - Let H be a *hypothesis* that \mathbf{X} belongs to class C
 - Classification is to determine $P(H|\mathbf{X})$, (i.e., *posteriori probability*): the probability that the hypothesis holds given the observed data sample \mathbf{X}
 - $P(H)$ (*prior probability*): the initial probability
 - E.g., \mathbf{X} will buy computer, regardless of age, income, ...
 - $P(\mathbf{X})$: probability that sample data is observed
 - $P(\mathbf{X}|H)$ (likelihood): the probability of observing the sample \mathbf{X} , given that the hypothesis holds
 - E.g., Given that \mathbf{X} will buy computer, the prob. that \mathbf{X} is 31..40, medium income

Prediction Based on Bayes' Theorem

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

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

- Informally, this can be viewed as
posteriori = likelihood x prior/evidence
- Predicts \mathbf{X} belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|\mathbf{X})$ for all the k classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification Is to Derive the Maximum Posteriori

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

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

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

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

needs to be maximized

Naïve Bayes Classifier

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

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

- ❑ This greatly reduces the computation cost: Only counts the class distribution
- ❑ 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 continuous-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

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

Naïve Bayes Classifier: Training Dataset

Class:

C1:`buys_computer = 'yes'`

C2:`buys_computer = 'no'`

Data to be classified:

$X = (\text{age} \leq 30, \text{Income} = \text{medium},$
 $\text{Student} = \text{yes}, \text{Credit_rating} = \text{Fair})$

| age | income | student | credit_rating | com |
|-----------|--------|---------|---------------|-----|
| ≤ 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 Bayes Classifier: An Example

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

| age | income | student | credit_rating | com |
|---------|--------|---------|---------------|-----|
| <=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 |

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

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

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

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

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

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

Avoiding the Zero-Probability Problem

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

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

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

Naïve Bayes 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 Bayes Classifier
 - How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection 
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Model Evaluation and Selection

- ❑ Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- ❑ Use **validation 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: Confusion Matrix

Confusion Matrix:

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

Example of Confusion Matrix:

| Actual class\Predicted class | buy_computer = yes | buy_computer = no | Total |
|------------------------------|--------------------|-------------------|-------|
| buy_computer = yes | 6954 | 46 | 7000 |
| buy_computer = no | 412 | 2588 | 3000 |
| Total | 7366 | 2634 | 10000 |

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

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

| A\P | C | $\neg C$ | |
|----------|----|----------|-----|
| C | TP | FN | P |
| $\neg C$ | FP | TN | N |
| | P' | N' | All |

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

$$\text{Accuracy} = (TP + TN)/\text{All}$$

- **Error rate**: $1 - \text{accuracy}$, or

$$\text{Error rate} = (FP + FN)/\text{All}$$

- **Class Imbalance Problem:**
 - One class may be *rare*, e.g. fraud, or HIV-positive
 - Significant *majority of the negative class* and minority of the positive class
- **Sensitivity**: True Positive recognition rate
 - **Sensitivity** = TP/P
- **Specificity**: True Negative recognition rate
 - **Specificity** = TN/N

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **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?
- Comment:
 - Perfect score is 1.0
 - Inverse relationship between precision & recall
- **F measure (or F-score):** harmonic mean of precision and recall
 - In general, it is the weighted measure of precision & recall

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

$$F = \frac{1}{\alpha \cdot \frac{1}{P} + (1 - \alpha) \cdot \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Assigning β times as much weight to recall as to precision)

- **F1-measure (balanced F-measure)**

- That is, when $\beta = 1$,

$$F_1 = \frac{2PR}{P + R}$$

Classifier Evaluation Metrics: Example

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

Precision = $90/230 = 39.13\%$

Recall = $90/300 = 30.00\%$

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 Classifier Accuracy: Bootstrap

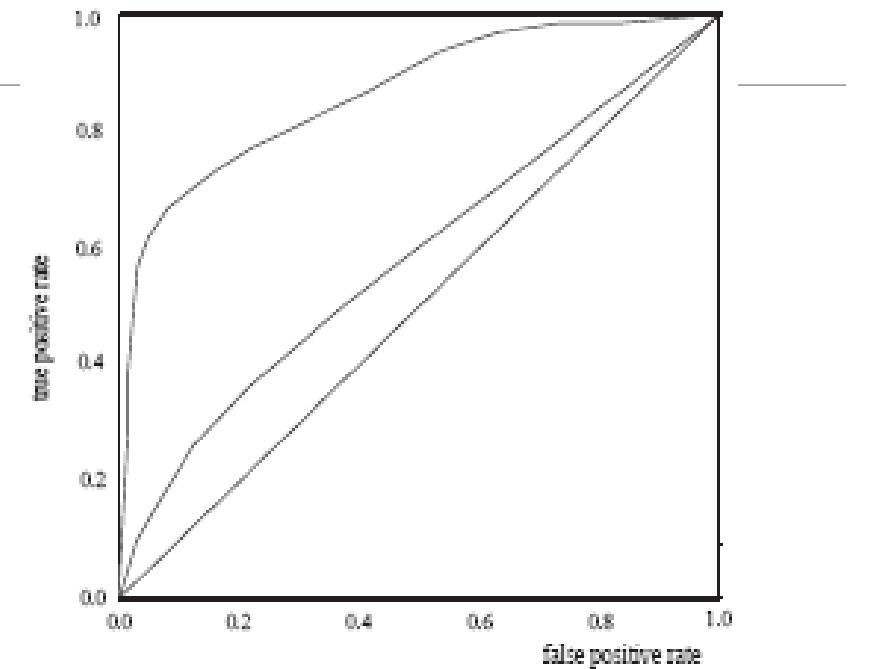
❑ Bootstrap

- ❑ Works well with small data sets
- ❑ Samples the given training tuples uniformly *with replacement*
 - ❑ 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 bootstrap**
 - ❑ 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:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$

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

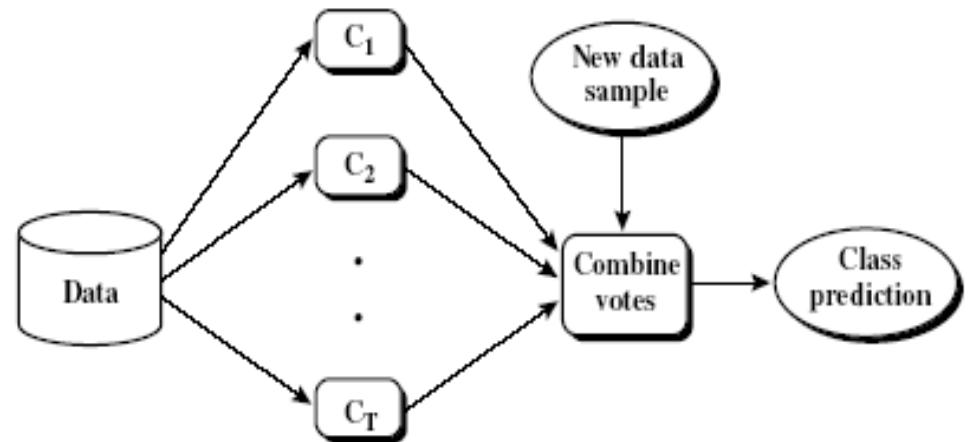
Issues Affecting Model Selection

- **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

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods 
- Summary

Ensemble Methods: Increasing the Accuracy



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

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
 - Given a set D of d tuples, at each iteration i , a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)
 - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample 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: Proved improved accuracy in prediction
 - Often significantly better than a single classifier derived from D
 - For noise data: not considerably worse, more robust

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
- ❑ Boosting algorithm can be extended for numeric prediction
- ❑ Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

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

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

- The weight of classifier M_i 's vote is

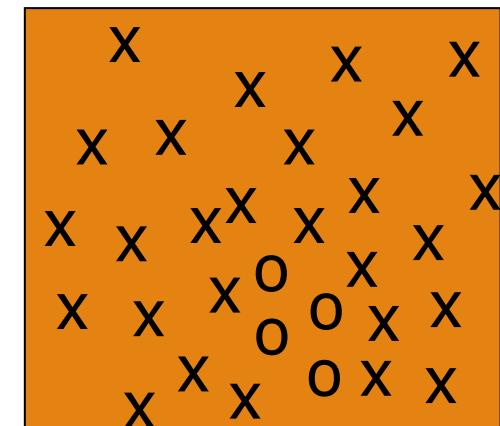
$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

Random Forest (Breiman 2001)

- ❑ Random Forest:
 - ❑ Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
 - ❑ During classification, each tree votes and the most popular class is returned
- ❑ Two Methods to construct Random Forest:
 - ❑ Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
 - ❑ Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- ❑ Comparable in accuracy to Adaboost, but more robust to errors and outliers
- ❑ Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods in two-class classification:
 - **Oversampling:** re-sampling of data from positive class
 - **Under-sampling:** randomly eliminate tuples from negative class
 - **Threshold-moving:** move the decision threshold, t , so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
 - **Ensemble techniques:** Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks



Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
 - Decision Tree Induction
 - Bayes Classification Methods
 - Model Evaluation and Selection
 - Techniques to Improve Classification Accuracy: Ensemble Methods
 - Summary
- 

Summary

- Classification: Extracting models describing important data classes
- Effective and scalable methods
 - Decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods
- Evaluation metrics:
 - Accuracy, sensitivity, specificity, precision, recall, F measure, and F_β measure
 - Stratified k-fold cross-validation is recommended for accuracy estimation
- Ensemble: Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models
- Model selection: **Significance tests** and **ROC curves**
- No single method has been found to be superior over all others for all data sets

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules.** Future Generation Computer Systems, 13, 1997
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning.** KDD'95
- A. J. Dobson. **An Introduction to Generalized Linear Models.** Chapman & Hall, 1990.
- 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.

References (2)

- ❑ 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. **C4.5: Programs for Machine Learning.** Morgan Kaufmann, 1993.
- ❑ J. R. Quinlan. **Bagging, boosting, and c4.5.** AAAI'96.

References (3)

- 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

