Data Mining: Classification

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Prediction
- Classification accuracy
- Summary

Classification vs. Prediction

- Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction:
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

Classification Process (1): Model Construction

Training Data

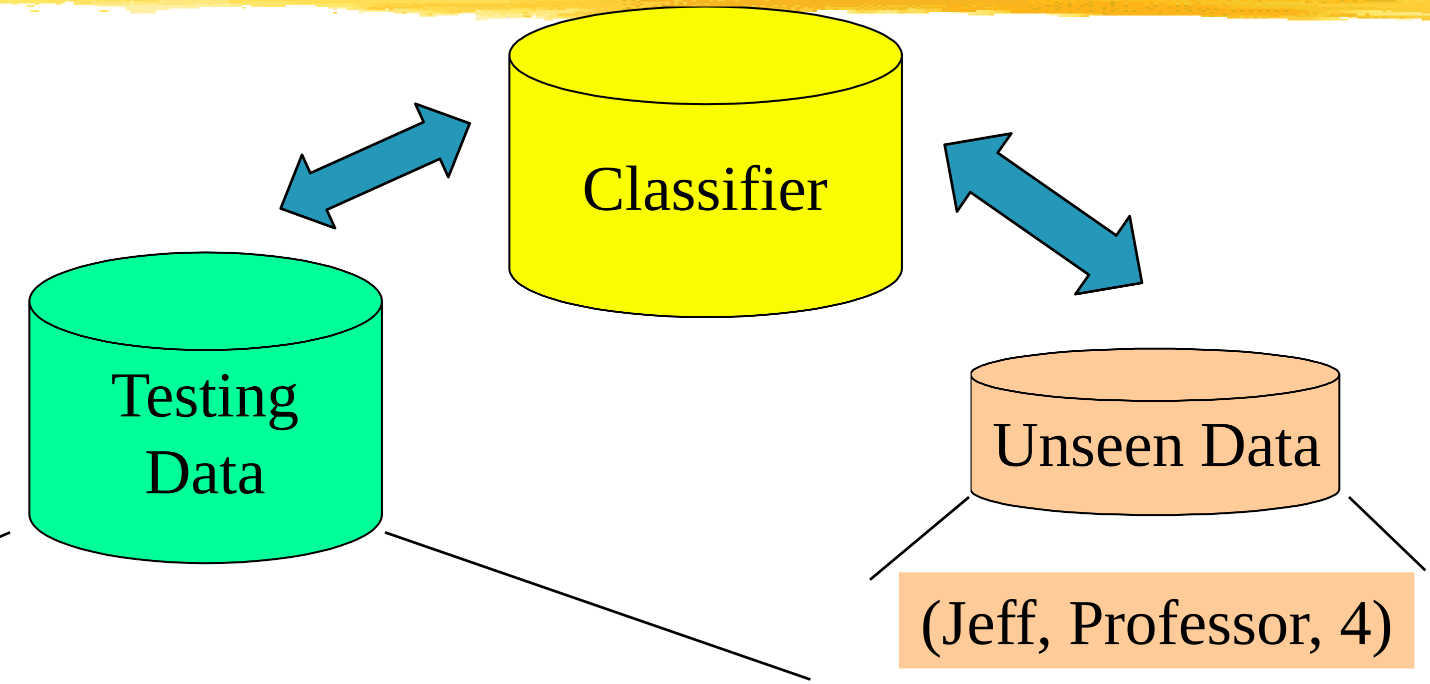
Classification Algorithms

Classifier (Model)

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

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Classification Process (2): Use 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

Tenured?

Yes

Supervised vs. Unsupervised Learning

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

Issues (1): 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 (2): Evaluating Classification Methods

- Predictive accuracy
- Speed and scalability
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Scalability
  - efficiency in disk-resident databases
- Goodness of rules
  - decision tree size
  - compactness of classification rules

Classification by Decision Tree Induction

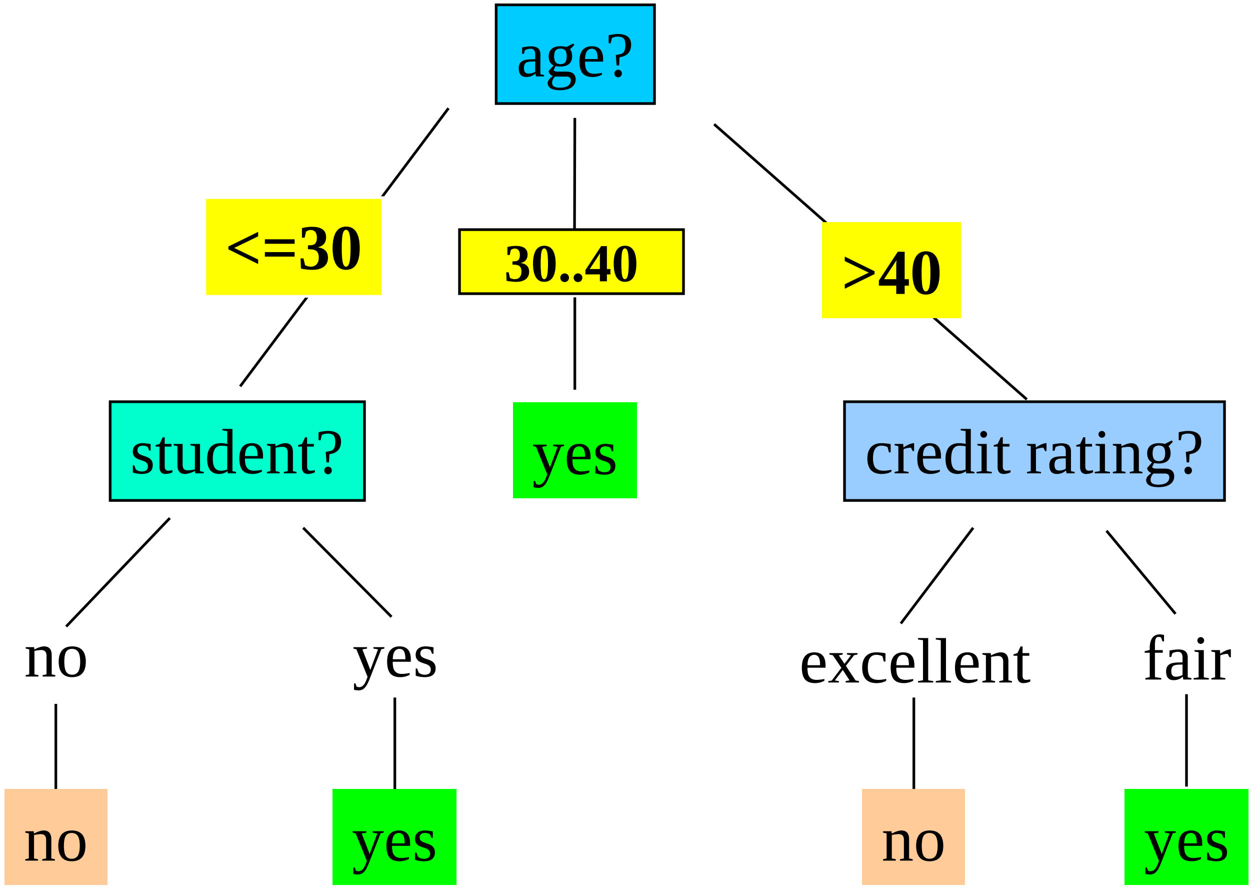
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Training Dataset

This follows an example from Quinlan's ID3

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

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)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
- Gini index (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that using attribute A a set *S* will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of *P* and n_i examples of *N*, the entropy, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on *A*

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for age:

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

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$Gain(age) = I(p,n) - E(age)$$

Similarly

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Gini Index (IBM IntelligentMiner)

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

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

  where p_j is the relative frequency of class *j* in *T*.
- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

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

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

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

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

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

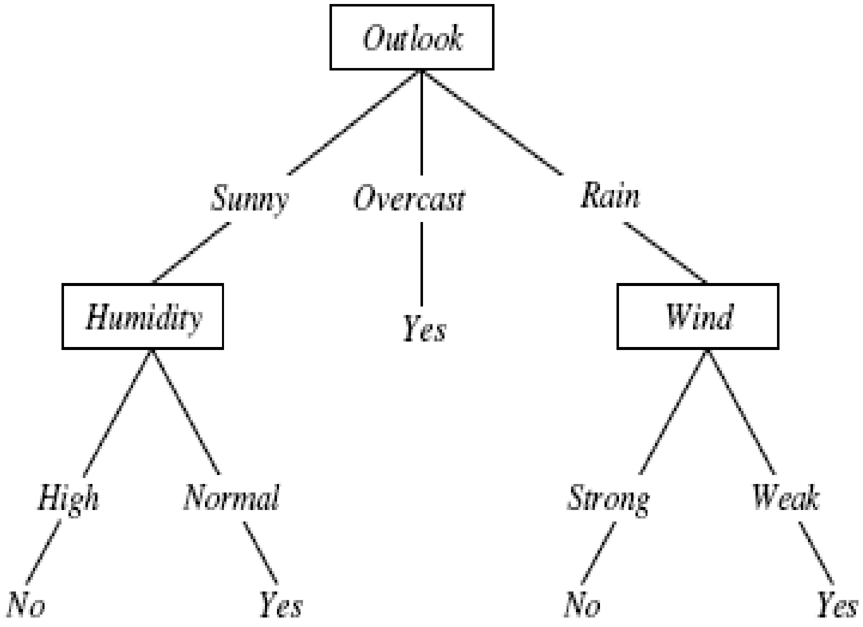
Over fitting in Decision Trees

- Why "over"-fitting?
  - A model can become more complex than the true target function (concept) when it tries to satisfy noisy data as well.

- Consider adding the following training example which is **incorrectly labeled as negative:**

Sky; Temp; Humidity; Wind; PlayTennis

Sunny; Hot; Normal; Strong; PlayTennis = No



- ID3 (the Greedy algorithm that was outlined) will make a new split and will classify future examples following the new path as negative.

- Problem is due to ”overfitting” the training data which may be thought as insufficient generalization of the training data
  - Coincidental regularities in the data
  - Insufficient data
  - Differences between training and test distributions

- Definition of overfitting
  - A hypothesis is said to overfit the training data if there exists some other hypothesis that has **larger** error over the training data but **smaller** error over the entire instances.

Classification Errors

Decision trees may obtain zero error on a set of training examples, but may misclassify examples not contained in the training set.

Training error: probability of error for training examples

True error: probability of error for unrestricted examples

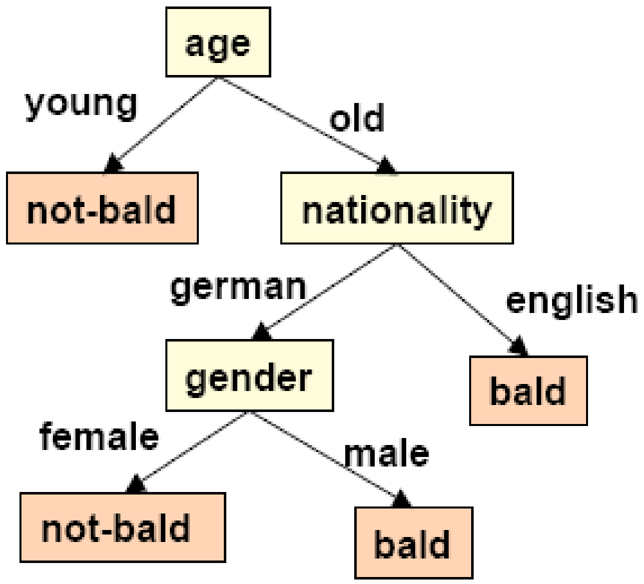
Overfitting of a decision tree to training data is an important source for errors.

Example:

male	young	german	not-bald
female	old	german	not-bald
male	old	english	bald
male	old	german	bald

Unfortunately, old english females will be classified as bald.

Overfitting is due to insufficient generalization of examples.



Inconsistent Example Sets

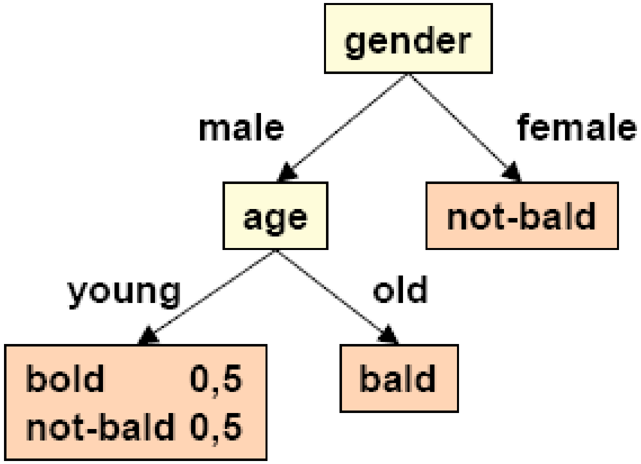
An example set is inconsistent if it contains examples which differ only by the value of the goal attribute.

Reasons for inconsistency:

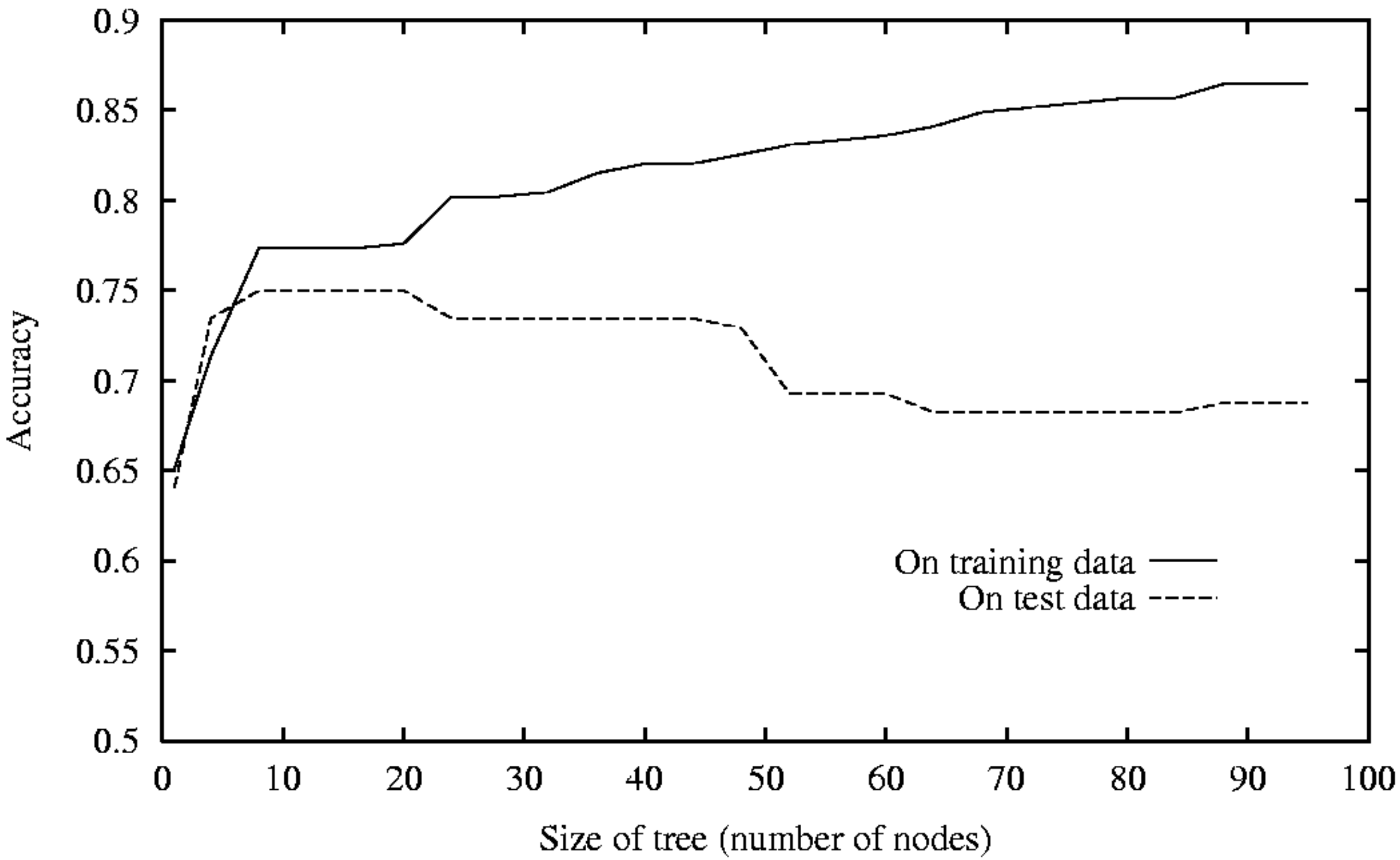
- too few or irrelevant attributes
- noisy data

Decision trees may reflect inconsistent examples by assigning probability values to conflicting outcomes at leaf nodes.

e1	male	young	not-bald
e2	female	old	not-bald
e3	male	old	bald
e4	female	young	not-bald
e5	male	young	bald



Over fitting in Decision Trees



Avoiding over-fitting the data

- How can we avoid overfitting? There are 2 approaches:
  1. **Early stopping**: stop growing the tree before it perfectly classifies the training data
  2. **Pruning:** grow full tree, then prune
     - Reduced error pruning
     - Rule post-pruning

- Pruning approach is found more useful in practice.

Avoid Overfitting in Classification

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

Approaches to Determine the Final Tree Size

- Separate training (2/3) and testing (1/3) sets
- Use cross validation, e.g., 10-fold cross validation
- Use all the data for training
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle:
  - halting growth of the tree when the encoding is minimized

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 in Data Mining Studies

- 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)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)

Bayesian Classification: Why?

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

Classification Accuracy: Estimating Error Rates

- Partition: Training-and-testing
  - use two independent data sets, e.g., training set (2/3), test set(1/3)
  - used for data set with large number of samples
- Cross-validation
  - divide the data set into *k* subsamples
  - use *k-1* subsamples as training data and one sub-sample as test data --- *k*-fold cross-validation
  - for data set with moderate size
- Bootstrapping (leave-one-out)
  - for small size data

Boosting and Bagging

- Boosting increases classification accuracy
  - Applicable to decision trees or Bayesian classifier
- Learn a series of classifiers, where each classifier in the series pays more attention to the examples misclassified by its predecessor
- Boosting requires only linear time and constant space

Boosting Technique (II) — Algorithm

- Assign every example an equal weight *1/N*
- *For t = 1, 2, ..., T Do*
  - Obtain a hypothesis (classifier) $h^{(t)}$ under $w^{(t)}$
  - Calculate the error of *h(t)* and re-weight the examples based on the error
  - Normalize $w^{(t+1)}$ to sum to 1
- Output a weighted sum of all the hypothesis, with each hypothesis weighted according to its accuracy on the training set

Summary

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Scalability is still an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of non-relational data, e.g., text, spatial, multimedia, etc..

References (I)

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95), pages 39-44, Montreal, Canada, August 1995.
- U. M. Fayyad. Branching on attribute values in decision tree generation. In Proc. 1994 AAAI Conf., pages 601-606, AAAI Press, 1994.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97), pages 111-120, Birmingham, England, April 1997.

References (II)

- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, pages 118-159. Blackwell Business, Cambridge Massechusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and c4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifer that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- 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.