

CS6220: DATA MINING TECHNIQUES


Matrix Data: Classification: Part 1

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

September 16, 2013

Matrix Data: Classification: Part 1

- Classification: Basic Concepts 
- Decision Tree Induction
- Model Evaluation and Selection
- 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
 - 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

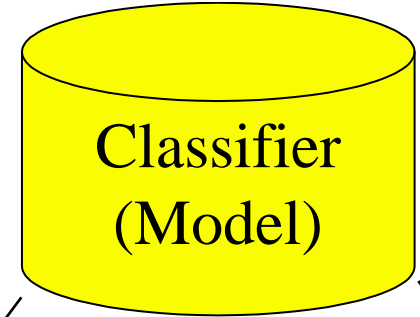
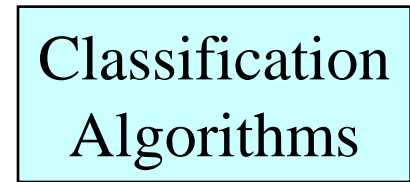
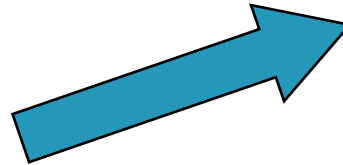
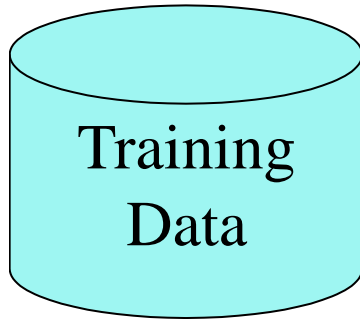
Classification—A Two-Step Process (1)

- **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**
 - For data point i : $\langle \mathbf{x}_i, y_i \rangle$
 - Features: \mathbf{x}_i ; class label: y_i
 - The model is represented as classification rules, decision trees, or mathematical formulae
 - Also called classifier
 - The set of tuples used for model construction is **training set**

Classification—A Two-Step Process (2)

- **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
 - **Test set** is independent of training set (otherwise overfitting)
 - **Accuracy** rate is the percentage of test set samples that are correctly classified by the model
 - Most used for binary classes
 - If the accuracy is acceptable, use the model to **classify new data**
- Note: If *the test set* is used to select models, it is called **validation (test) set**

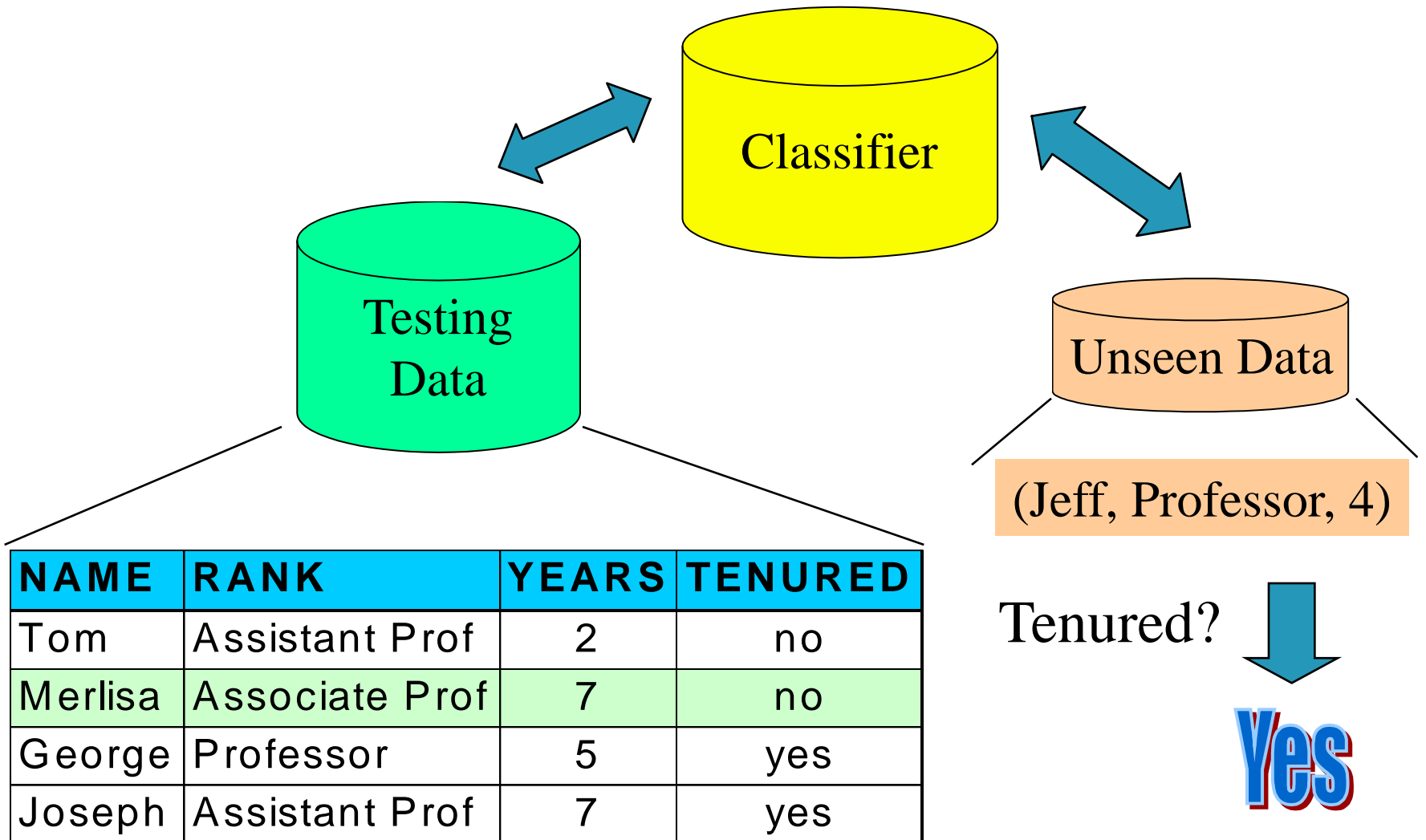
Process (1): Model Construction



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

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


Process (2): Using the Model in Prediction



Classification Methods Overview

- Part 1
 - Decision Tree
 - Model Evaluation
- Part 2
 - Bayesian Learning: Naïve Bayes, Bayesian belief network
 - Logistic Regression
- Part 3
 - SVM
 - kNN
 - Other Topics

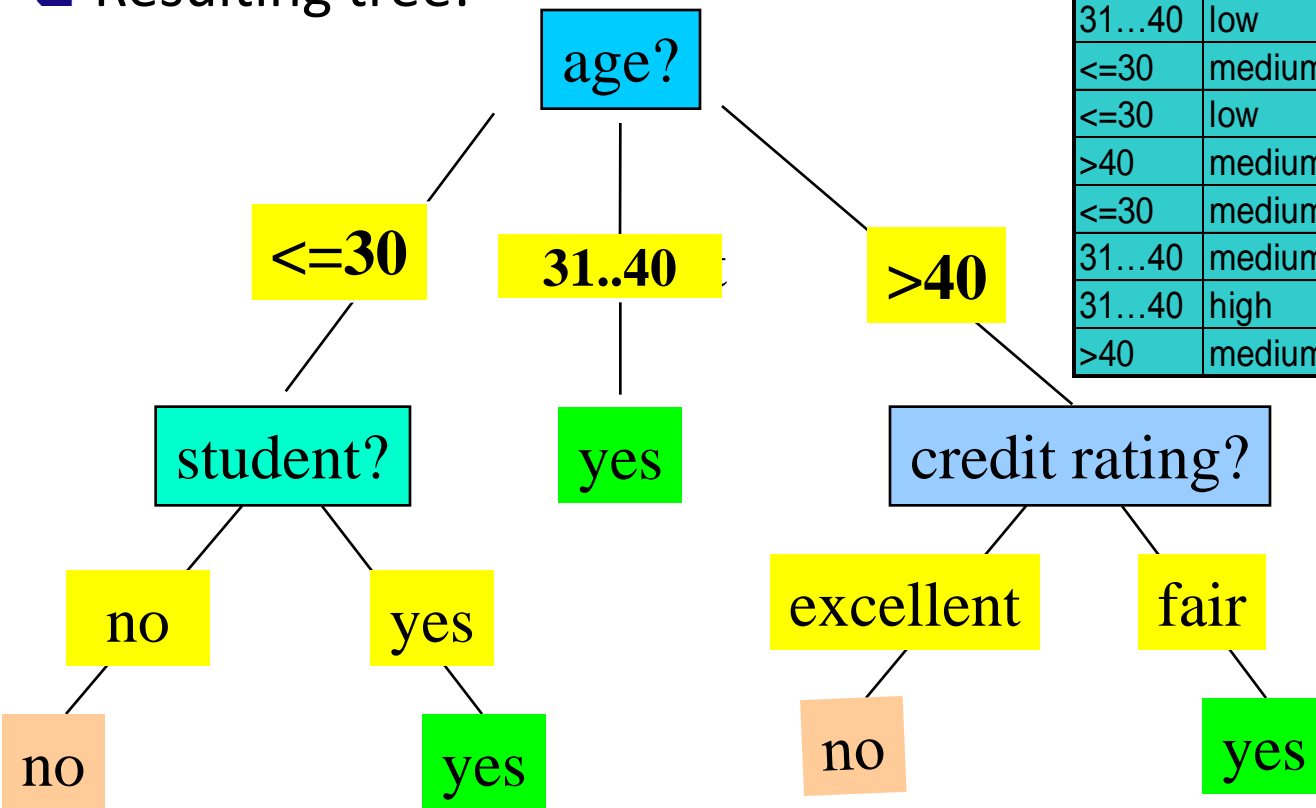
Matrix Data: Classification: Part 1

- Classification: Basic Concepts
- Decision Tree Induction 
- Model Evaluation and Selection
- 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

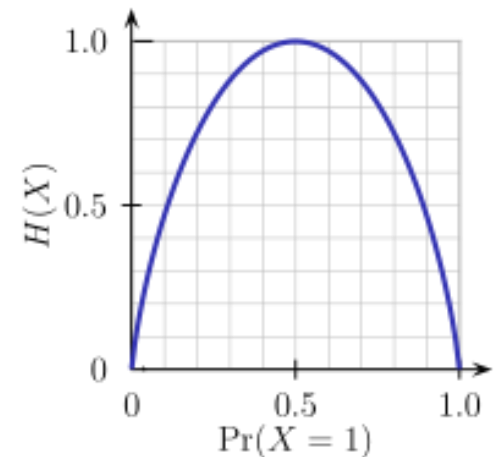


Algorithm for Decision Tree Induction

- 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 (impurity) associated with a random variable
 - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \dots, y_m\}$,
 - $H(Y) = -\sum_{i=1}^m p_i \log(p_i)$, where $p_i = P(Y = y_i)$
 - Interpretation:
 - Higher entropy => higher uncertainty
 - Lower entropy => lower uncertainty
- Conditional Entropy
 - $H(Y|X) = \sum_x p(x)H(Y|X = x)$



$m = 2$

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”

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

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
31...40	4	0	0
> 40	3	2	0.971

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

$\frac{5}{14} I(2,3)$ 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,

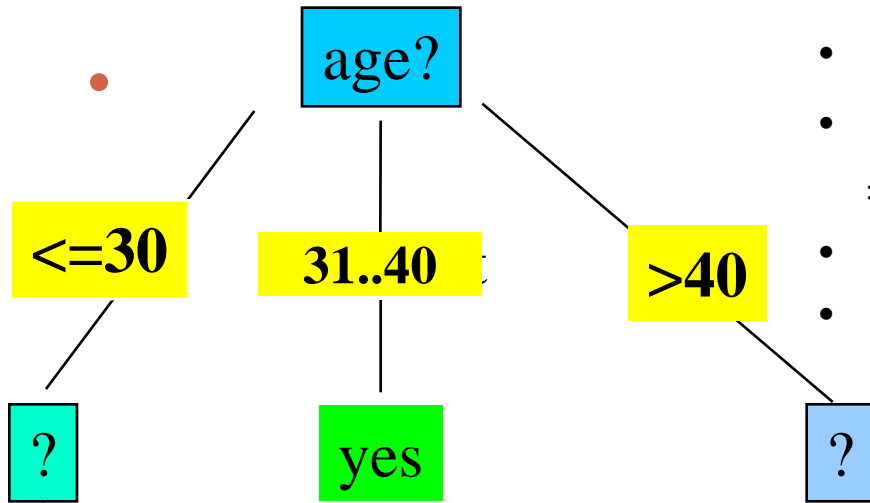
$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

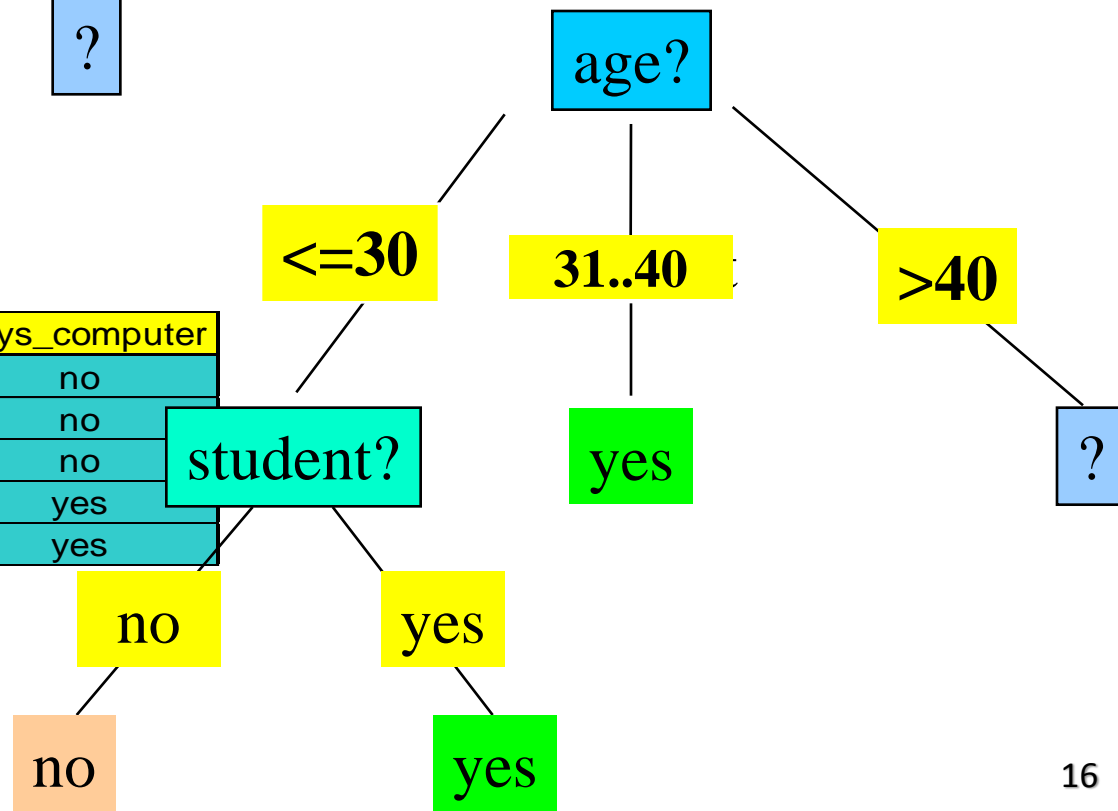
age	income	student	credit_rating	buys_computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
31...40	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
31...40	low	yes	excellent	yes
≤ 30	medium	no	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium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

Attribute Selection for a Branch



- $Info(D_{age \leq 30}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$
- $Gain_{age \leq 30}(income) = Info(D_{age \leq 30}) - Info_{income}(D_{age \leq 30}) = 0.571$
- $Gain_{age \leq 30}(student) = 0.971$
- $Gain_{age \leq 30}(credit_rating) = 0.02$

Which attribute next?



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

$D_{age \leq 30}$

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:
 - D_1 is the set of tuples in D satisfying $A \leq \text{split-point}$, and D_2 is the set of tuples in D satisfying $A > \text{split-point}$

Gain Ratio for Attribute Selection (C4.5)

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

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

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

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

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

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 (why?)
 - **Gini index:**
 - biased to multivalued attributes

*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”

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 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)

*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	2	3
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

*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.

Matrix Data: Classification: Part 1

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection 
- 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
- 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, $\mathbf{CM}_{i,j}$ 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	-C	
C	TP	FN	P
-C	FP	TN	N
	P'	N'	All

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

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

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

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

- **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

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

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

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

- Perfect score is 1.0
- Inverse relationship between precision & recall
- **F measure (F_1 or F-score):** harmonic mean of precision and recall,

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

- F_β : weighted measure of precision and recall
 - assigns β times as much weight to recall as to precision

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

Classifier Evaluation Metrics: Example

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

- $Precision = 90/230 = 39.13\%$

$$Recall = 90/300 = 30.00\%$$

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

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

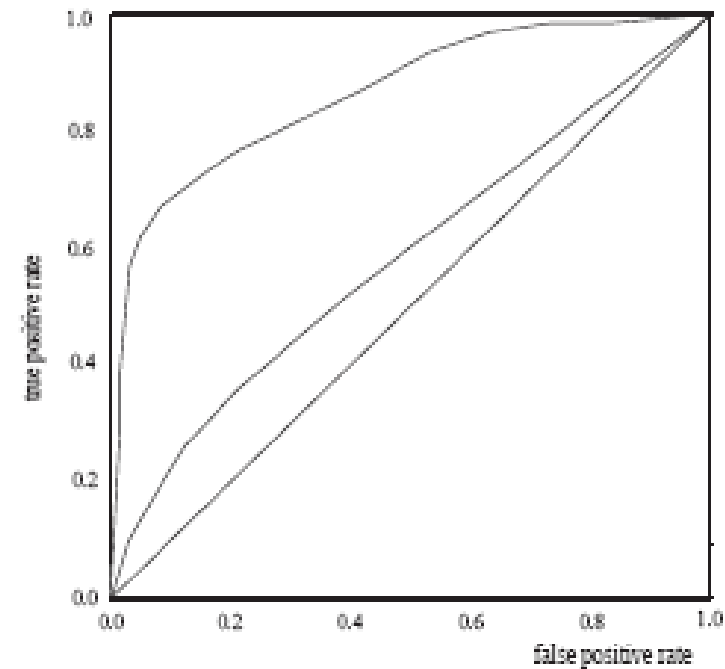
- Suppose we have 2 classifiers, M_1 and M_2 , which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *point estimates* of error on the true population of *future* data cases
- What if the difference between the 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 of two models: M_1 & M_2
- Assume samples follow normal distribution
- Use two sample **t-test** (or **Student's t-test**)
- **Null Hypothesis:** M_1 & M_2 are the same (means are equal)
- If we can **reject** null hypothesis, then
 - we conclude that the difference between M_1 & M_2 is **statistically significant**
 - Chose model with lower error rate

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
- Area under the curve: the closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



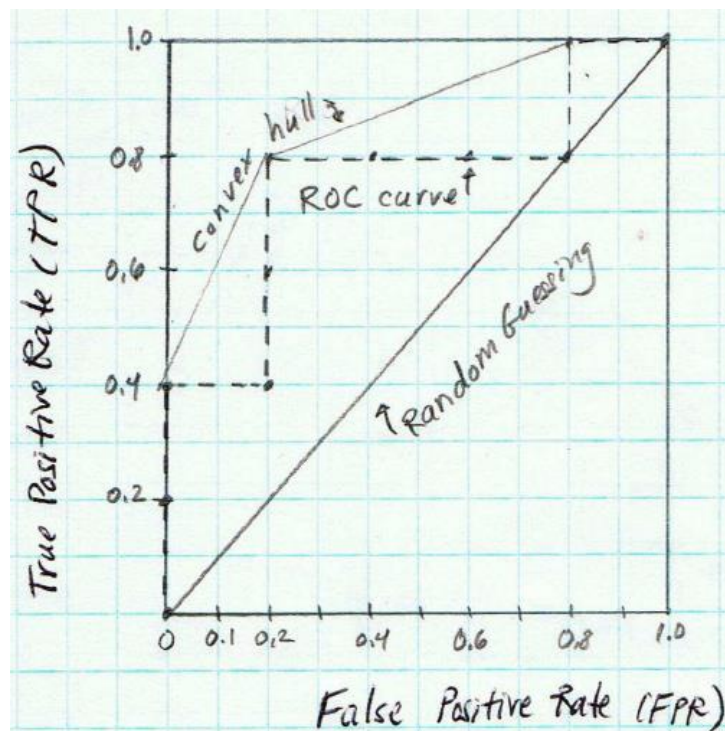
- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Plotting an ROC Curve

- True positive rate: $TPR = TP/P$ (sensitivity)
- False positive rate: $FPR = FP/N$ (1-specificity)
- Rank tuples according to how likely they will be a positive tuple
 - Idea: when we include more tuples in, we are more likely to make mistakes, that is the **trade-off!**
 - Nice property: not threshold (cut-off) need to be specified, only rank matters

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	p	0.9	1	0	5	4	0.2	0
2	p	0.8	2	0	5	3	0.4	0
3	n	0.7	2	1	4	3	0.4	0.2
4	p	0.6	3	1	4	2	0.6	0.2
5	p	0.55	4	1	4	1	0.8	0.2
6	n	0.54	4	2	3	1	0.8	0.4
7	n	0.53	4	3	2	1	0.8	0.6
8	n	0.51	4	4	1	1	0.8	0.8
9	p	0.50	5	4	0	1	1.0	0.8
10	n	0.4	5	5	0	0	1.0	1.0


Example



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

Matrix Data: Classification: Part 1

- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Summary 

Summary

- **Classification** is a form of data analysis that extracts **models** describing important data classes.
- **decision tree induction**
- **Evaluation**
 - **Evaluation metrics** include: accuracy, sensitivity, specificity, precision, recall, F measure, and F_β measure.
 - **Stratified k-fold cross-validation** is recommended for accuracy estimation.
 - **Significance tests** and **ROC curves** are useful for model selection.

-
- Course project sign-up will be due next Monday
 - Homework0: test our course system
 - Write a “hello world” program and submit it to our system

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. Indurkha. **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.