

# CSE 5243 INTRO. TO DATA MINING

## Classification (Basic Concepts)

Huan Sun, CSE@The Ohio State University

# Classification: Basic Concepts

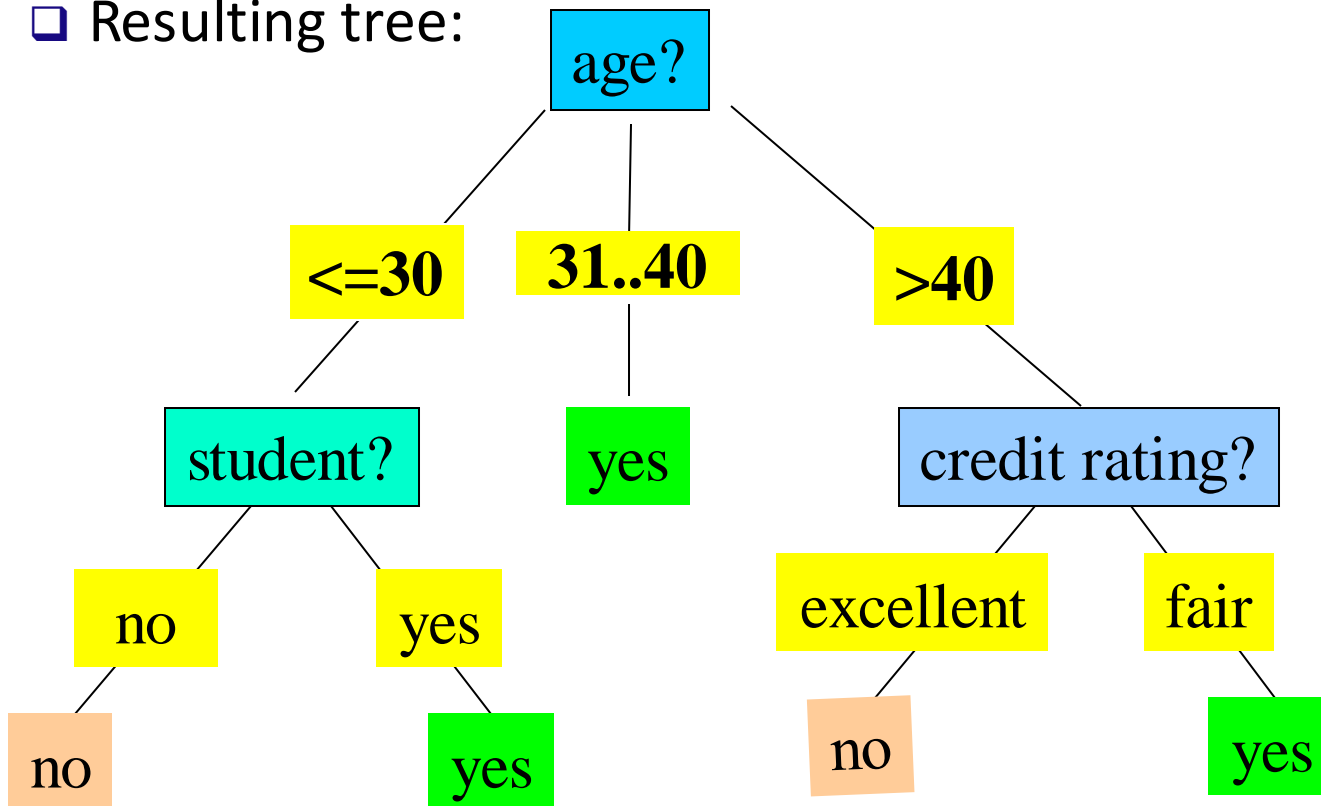
- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Practical Issues of Classification
- Bayes Classification Methods
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary



This class

# 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

# Algorithm Outline

- Split (node, {data tuples})
  - ▣  $A \leq$  the **best attribute** for splitting the {data tuples}
  - ▣ Decision attribute for this node  $\leq A$
  - ▣ For each value of  $A$ , create new child node
  - ▣ For each child node / subset:
    - If one of the stopping conditions is satisfied: STOP
    - Else: Split (child\_node, {subset})

ID3 algorithm: how it works

[https://www.youtube.com/watch?v=\\_XhOdSLIE5c](https://www.youtube.com/watch?v=_XhOdSLIE5c)

# 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"

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

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

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

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

## 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)

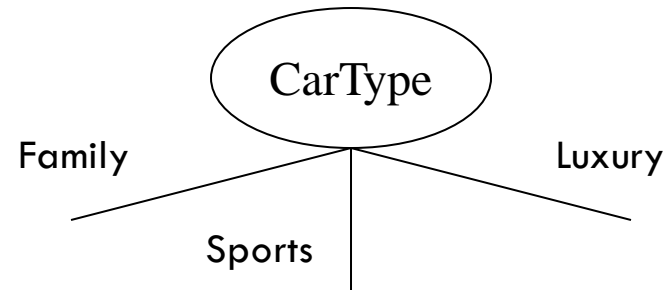
$$\text{SplitInfo}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- The entropy of the partitioning, or the potential information generated by splitting  $D$  into  $v$  partitions.
- **GainRatio(A) = Gain(A)/SplitInfo(A)** (normalizing Information Gain)

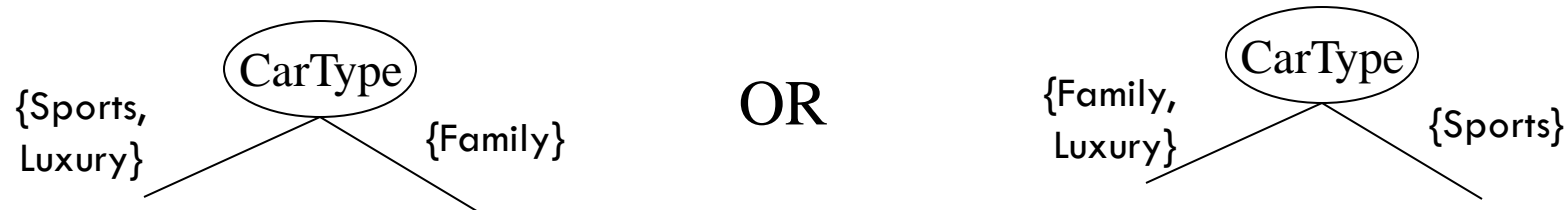


# Splitting Based on Nominal Attributes

- **Multi-way split:** Use as many partitions as distinct values.



- **Binary split:** Divides values into two subsets. **Need to find optimal partitioning.**



# Measures of Node Impurity

- Entropy: 
$$H(Y) = - \sum_{i=1}^m p_i \log(p_i) \quad \text{where } p_i = P(Y = y_i)$$
  - ▣ Higher entropy => higher uncertainty, higher node impurity
  - ▣ Why entropy is used in information gain
- Gini Index
- Misclassification error

# Gini Index (CART, IBM IntelligentMiner)

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

$$gini(D) = 1 - \sum_{j=1}^n p_j^2, \text{ where } p_j \text{ is the relative frequency of class } j \text{ in } D$$

- If a data set  $D$  is split on  $A$  into two subsets  $D_1$  and  $D_2$ , the *gini index after the split* 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  $gini_A(D)$  (or, **the largest reduction in impurity**) is chosen to split the node.

# Binary Attributes: Computing Gini Index

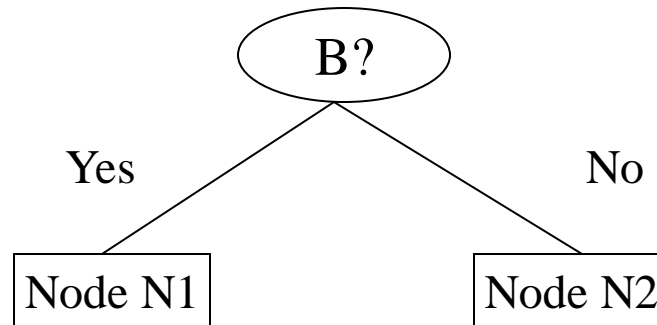
- Splits into two partitions
- Effect of weighing partitions:
  - Prefer Larger and Purer Partitions.

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

$$\begin{aligned} Gini(N1) &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.408 \end{aligned}$$

$$\begin{aligned} Gini(N2) &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.320 \end{aligned}$$

	<b>N1</b>	<b>N2</b>
C1	<b>5</b>	<b>1</b>
C2	<b>2</b>	<b>4</b>
<b>Gini=0.371</b>		



	<b>Parent</b>
C1	<b>6</b>
C2	<b>6</b>
<b>Gini = 0.500</b>	

$$\begin{aligned} Gini(Children) &= 7/12 * 0.408 + \\ &\quad 5/12 * 0.320 \\ &= 0.371 \end{aligned}$$



weighting

# Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	<b>0.393</b>		

Two-way split  
(find best partition of values)

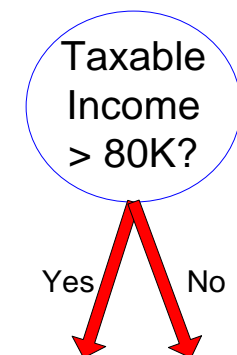
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	<b>0.400</b>	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	<b>0.419</b>	

# Continuous Attributes: Computing Gini Index or Information Gain

- **To discretize the attribute values**
  - Use Binary Decisions based on one splitting value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values - 1
  - 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  $a_{i+1}$
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A < v$  and  $A \geq v$
- Simple method to choose best  $v$ 
  - For each  $v$ , scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



# Continuous Attributes: Computing Gini Index or expected information requirement

**First decide the splitting value to discretize the attribute:**

□ For efficient computation: for each attribute,

Step 1: Sort the attribute on values

Step 2: Linearly scan these values, each time updating the count matrix


Step 3: **Computing Gini index and choose the split position that has the least Gini index**

Cheat		No	No	No	Yes	Yes	Yes	No	No	No	No						
		Taxable Income															
Step 1:	Sorted Values →	60	70	75	85	90	95	100	120	125	220						
	Possible Splitting Values →		65	72	80	87	92	97	110	122	172						
			<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >						
Step 2:	Yes	0	3	0	3	1	2	2	1	3	0	3	0	3	0		
	No	1	6	2	5	3	4	3	4	3	4	4	3	5	2	6	1
Step 3:	Gini		0.400	0.375	0.343	0.417	0.400	<u>0.300</u>	0.343	0.375	0.400						



At each level of the decision tree, for attribute selection, (1) First, discretize a continuous attribute by deciding the splitting value; (2) Then, compare the discretized attribute with other attributes in terms of Gini Index reduction or Information Gain.

# Classification: Basic Concepts

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



# Classifier Evaluation Metrics: Confusion Matrix

**Confusion Matrix:**

Actual class \ Predicted class	$C_1$	$\neg C_1$
$C_1$	<b>True Positives (TP)</b>	<b>False Negatives (FN)</b>
$\neg C_1$	<b>False Positives (FP)</b>	<b>True Negatives (TN)</b>

**Example of Confusion Matrix:**

Actual class \ Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	<b>6954</b>	<b>46</b>	7000
buy_computer = no	<b>412</b>	<b>2588</b>	3000
Total	7366	2634	10000

- Given  $m$  classes, an entry,  $CM_{ij}$  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

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

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

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

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

# Limitation of Accuracy

- Consider a 2-class problem
  - ▣ Number of Class 0 examples = 9990
  - ▣ Number of Class 1 examples = 10
- If a model predicts everything to be class 0,  
Accuracy is  $9990/10000 = 99.9\%$ 
  - ▣ Accuracy is misleading because model does not detect any class 1 example

# Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F-measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

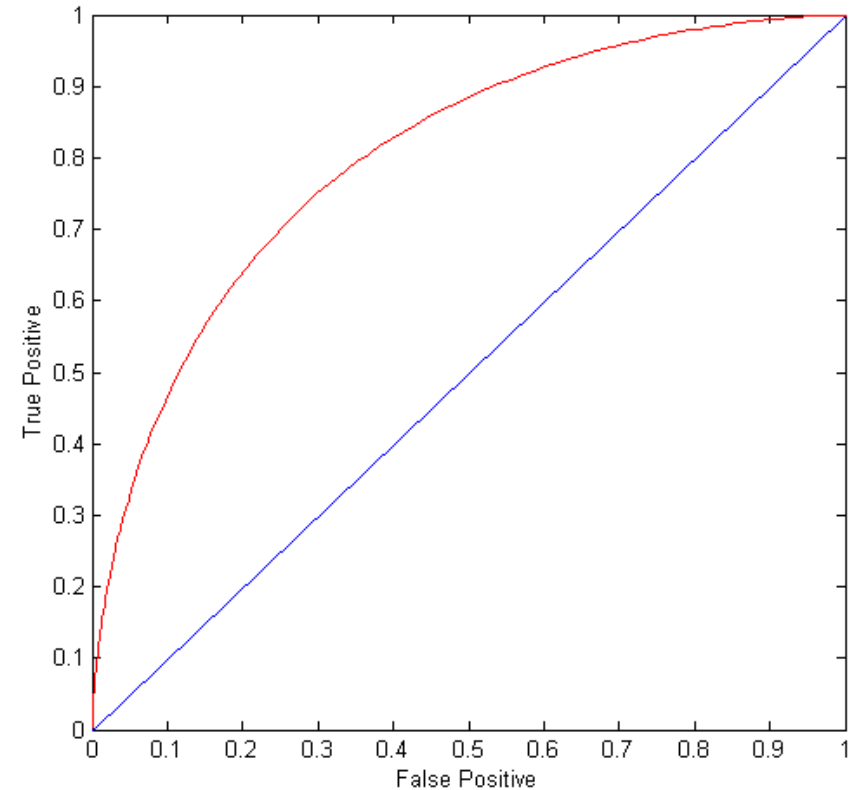
$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

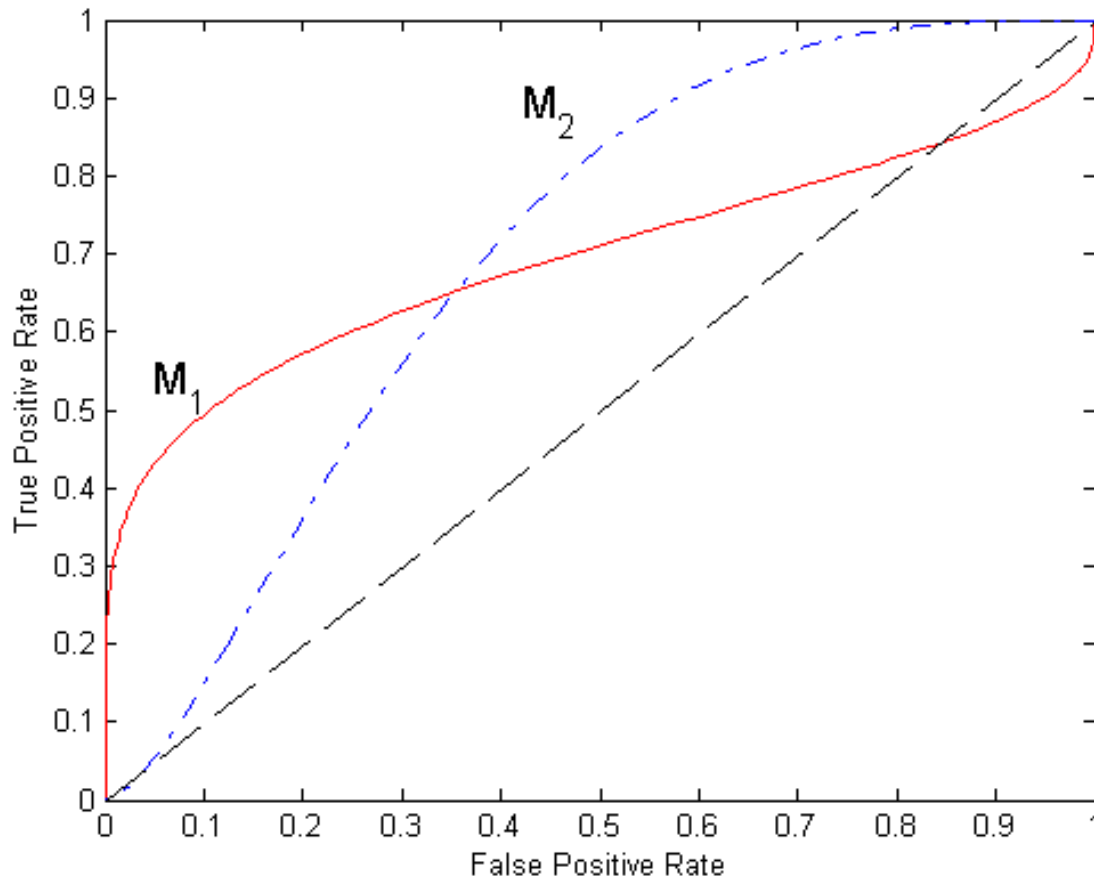
- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout  $k$  times, accuracy = avg. of the accuracies 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

# Using ROC for Classification Model Comparison

- **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




# Using ROC for Classification Model Comparison



- No model consistently outperform the other
  - $M_1$  is better for small FPR
  - $M_2$  is better for large FPR
- Area Under the ROC curve
  - **Ideal:**
    - Area = 1
  - **Random guess** (diagonal line):
    - Area = 0.5

# Classification: Basic Concepts

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

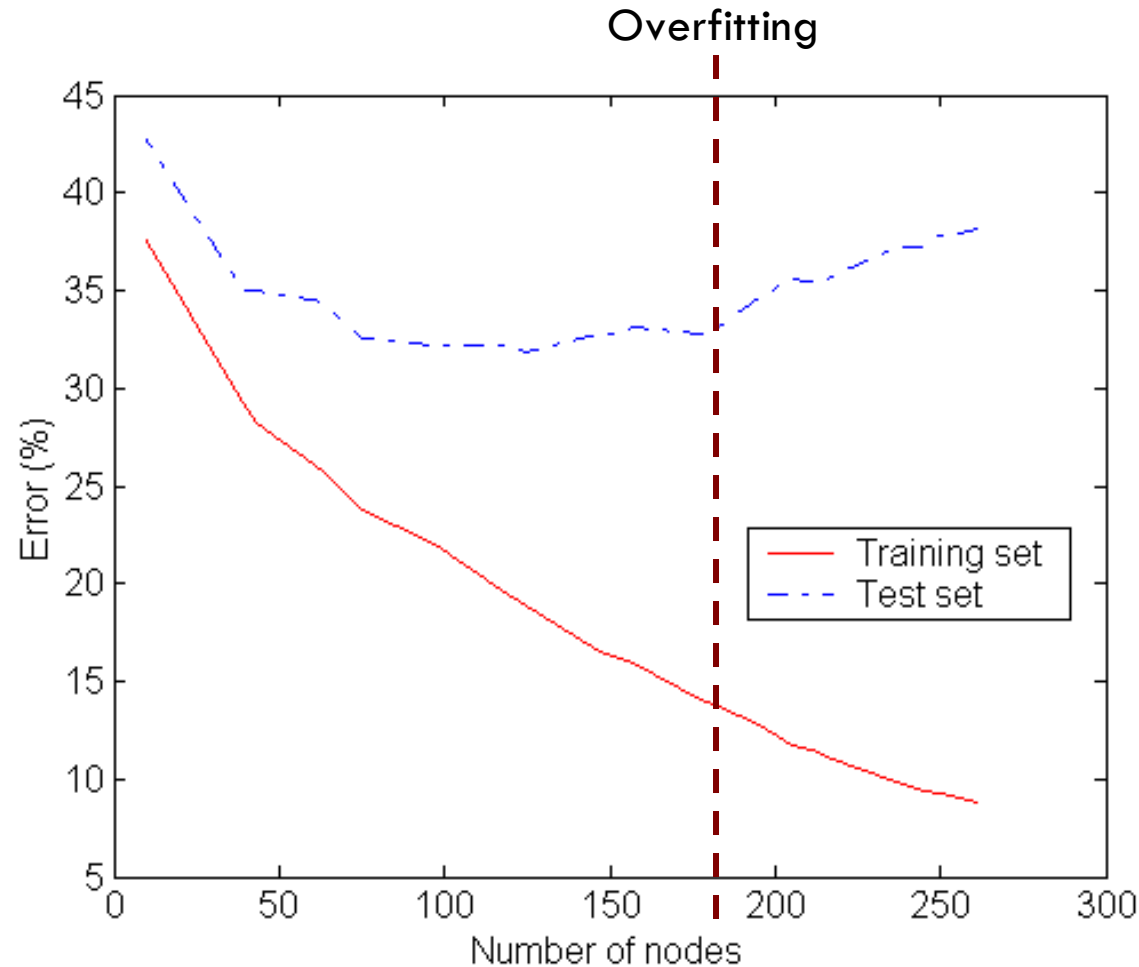


# Practical Issues of Classification

---

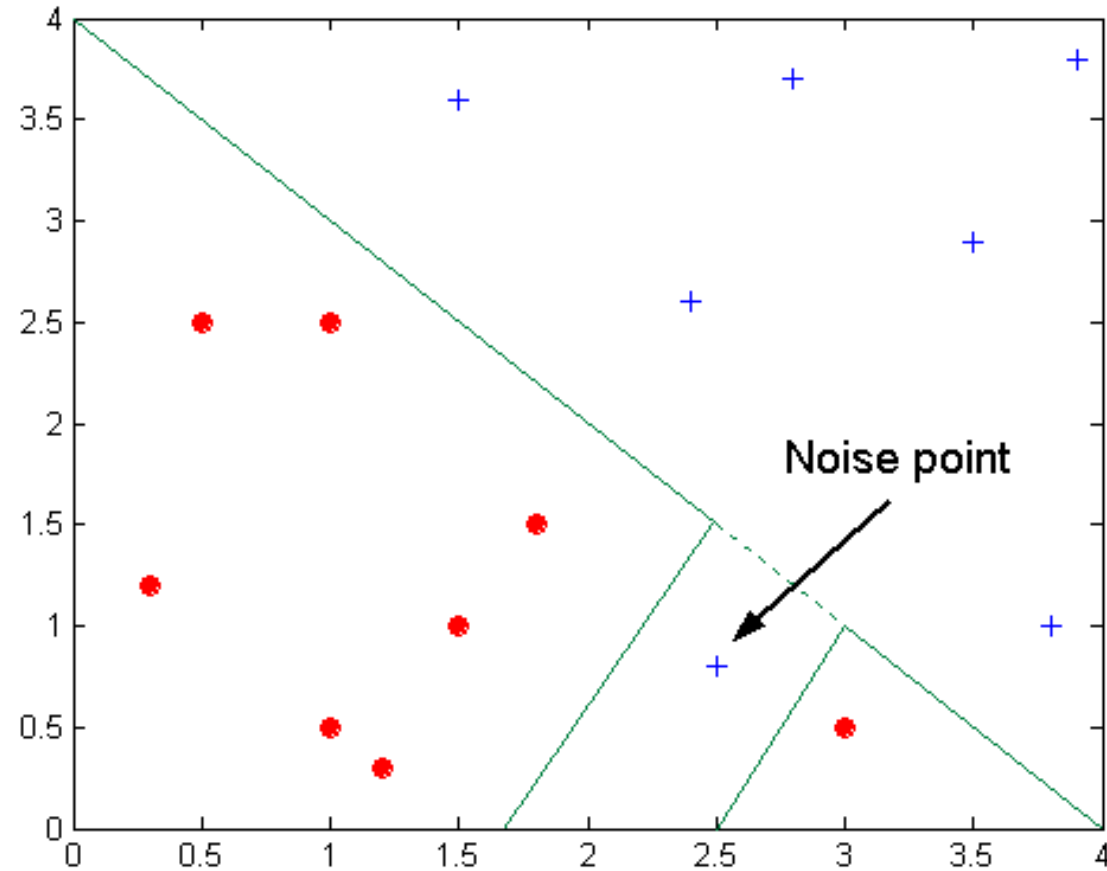
- Underfitting and Overfitting
- Missing Values
- Costs of Classification

# Underfitting and Overfitting



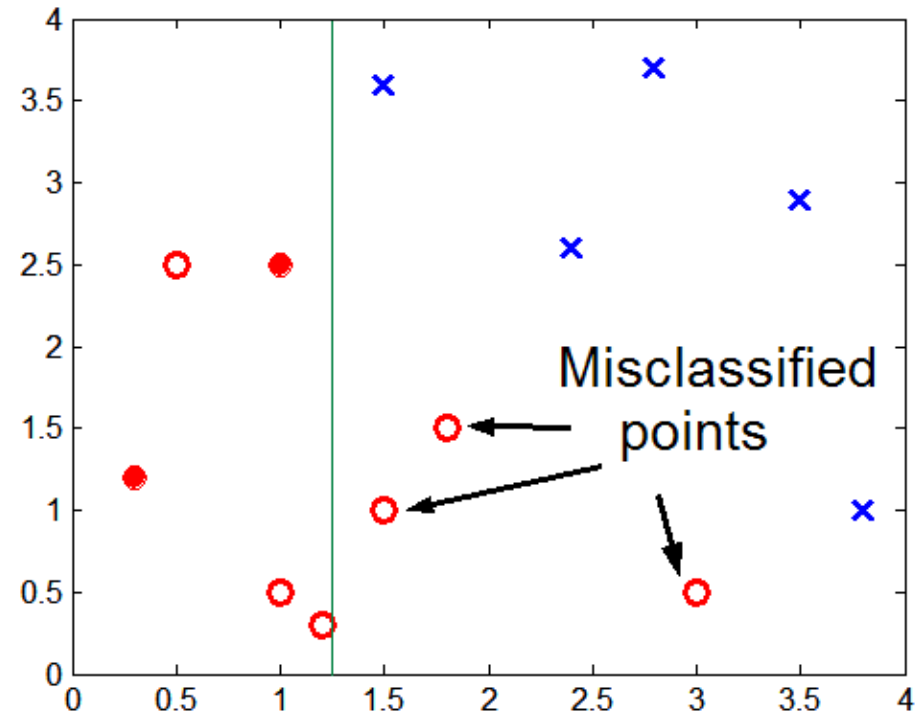
Underfitting: when model is too simple, both training and test errors are large

# Overfitting due to Noise



Decision boundary is distorted by noise point

# Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

# Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

# Estimating Generalization Errors

- **Re-substitution errors:** error on training ( $\sum e(t)$ )
- **Generalization errors:** error on testing ( $\sum e'(t)$ )

# Estimating Generalization Errors

- **Re-substitution errors:** error on training ( $\sum e(t)$ )
- **Generalization errors:** error on testing ( $\sum e'(t)$ )
- Methods for estimating generalization errors:
  - **Optimistic approach:**  $e'(t) = e(t)$

# Estimating Generalization Errors

- **Re-substitution errors:** error on training ( $\sum e(t)$ )
- **Generalization errors:** error on testing ( $\sum e'(t)$ )
- Methods for estimating generalization errors:
  - **Optimistic approach:**  $e'(t) = e(t)$
  - **Pessimistic approach:**
    - For each leaf node:  $e'(t) = (e(t)+0.5)$
    - Total errors:  $e'(T) = e(T) + N \times 0.5$  (N: number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):

$$\text{Training error} = 10/1000 = 1\%$$

$$\text{Generalization error} = (10 + 30 \times 0.5)/1000 = 2.5\%$$



# Estimating Generalization Errors

- **Re-substitution errors:** error on training ( $\sum e(t)$ )
- **Generalization errors:** error on testing ( $\sum e'(t)$ )
- Methods for estimating generalization errors:
  - **Optimistic approach:**  $e'(t) = e(t)$
  - **Pessimistic approach:**
    - For each leaf node:  $e'(t) = (e(t)+0.5)$
    - Total errors:  $e'(T) = e(T) + N \times 0.5$  (**N: number of leaf nodes**)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
      - Training error =  $10/1000 = 1\%$
      - Generalization error =  $(10 + 30 \times 0.5)/1000 = 2.5\%$
  - **Reduced error pruning (REP):**
    - uses validation data set to estimate generalization error

# How to Address Overfitting

## □ Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree

- Typical stopping conditions for a node:

- Stop if all instances belong to the same class
- Stop if all the attribute values are the same

- More restrictive conditions:

- Stop if number of instances is less than some user-specified threshold
- Stop if class distribution of instances are independent of the available features (e.g., using  $\chi^2$  test)
- Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

# How to Address Overfitting...

## □ Post-pruning

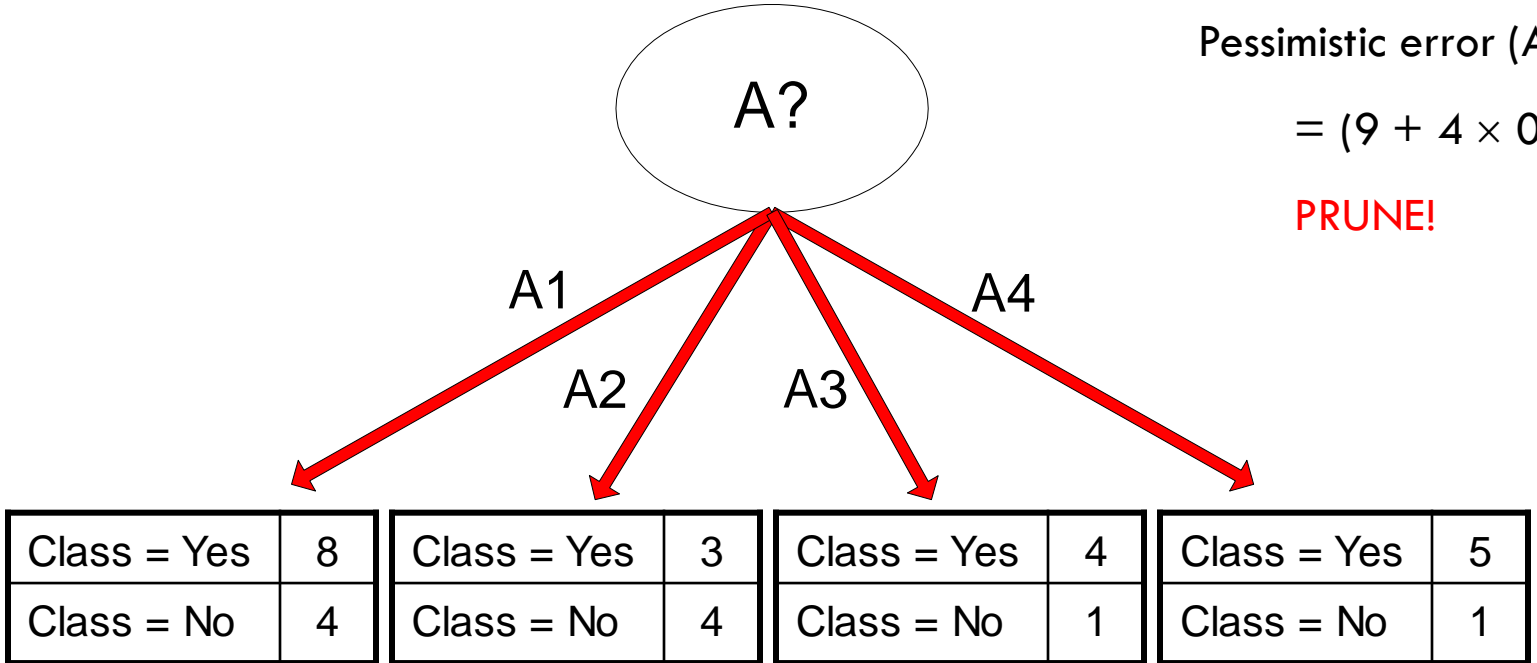
- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the sub-tree
- Can use MDL for post-pruning

# Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30  
Pessimistic error = (10 + 0.5)/30 = 10.5/30  
Training Error (After splitting) = 9/30  
Pessimistic error (After splitting)  
= (9 + 4 × 0.5)/30 = 11/30

**PRUNE!**



# Examples of Post-pruning

- Optimistic error?

Don't prune for both cases

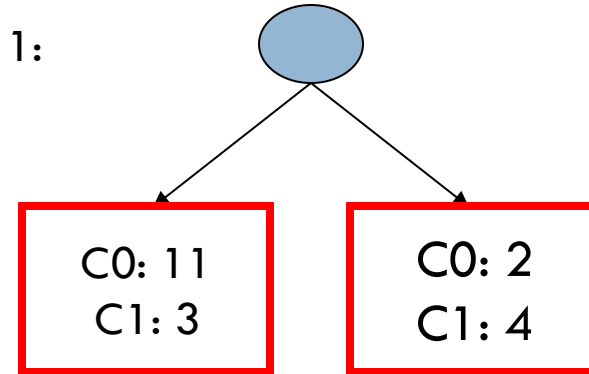
- Pessimistic error?

Don't prune case 1, prune case 2

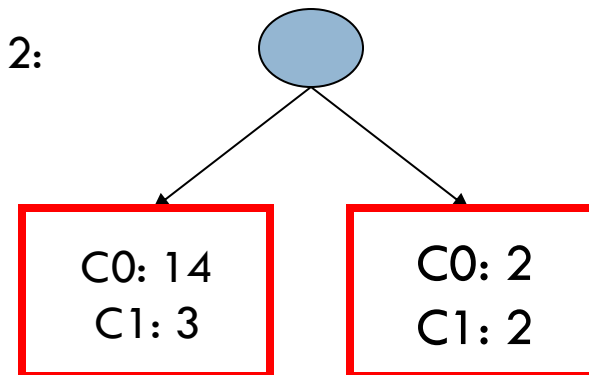
- Reduced error pruning?

Depends on validation set

Case 1:



Case 2:



# Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

# Handling Missing Attribute Values

- **Missing values affect decision tree construction in three different ways:**
  - ▣ Affects how impurity measures are computed
  - ▣ Affects how to distribute instance with missing value to child nodes
  - ▣ Affects how a test instance with missing value is classified
- While the book describes a few ways it can be handled as part of the process – it is often best to handle this using standard statistical methods
  - ▣ **EM-based estimation** (Covered in the second lecture, 08/28/2018)

# 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, naive 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

## □ Bayes' Theorem:

- Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
- Let  $H$  be a *hypothesis* that  $X$  belongs to class  $C$
- Classification is to determine  $P(H | \mathbf{X})$ , (i.e., **posteriori probability**): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$

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

# Bayes' Theorem: Basics

## □ Bayes' Theorem:

- ▣ Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
- ▣ Let  $H$  be a *hypothesis* that  $X$  belongs to class  $C$
- ▣ Classification is to determine  $P(H | \mathbf{X})$ , (i.e., *posteriori probability*): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$

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

- ▣  $P(H)$  (*prior probability*): the initial probability
  - ▣ E.g.,  $\mathbf{X}$  will buy computer, regardless of age, income, ...

# Bayes' Theorem: Basics

## □ Bayes' Theorem:

- Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
- Let  $H$  be a *hypothesis* that  $X$  belongs to class  $C$
- Classification is to determine  $P(H | \mathbf{X})$ , (i.e., *posteriori probability*): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$

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

- $P(H)$  (*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

# Bayes' Theorem: Basics

## □ Bayes' Theorem:

- Let  $\mathbf{X}$  be a data sample ("evidence"): class label is unknown
- Let  $H$  be a *hypothesis* that  $X$  belongs to class  $C$
- Classification is to determine  $P(H | \mathbf{X})$ , (i.e., **posteriori probability**): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$

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

- $P(H)$  (*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  $X$  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  $n$ -dimensional 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  $P(C_i | \mathbf{X})$

# 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  $n$ -dimensional 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  $P(\mathbf{X})$  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



# 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  $D$ )

# Naïve Bayes Classifier

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

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

- If  $A_k$  is **continuous-valued**,  $P(x_k | C_i)$  is usually computed based on Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$

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

and  $P(x_k | C_i)$  is

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

# Naïve Bayes Classifier: Training Dataset

Class:

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

Data to be classified:

X = (age <=30, Income = medium,  
Student = yes, Credit\_rating = Fair)

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

# Naïve Bayes Classifier: An Example

- $P(C_i)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$

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

# 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, where,  
 $X = (\text{age} \leq 30, \text{Income} = \text{medium}, \text{Student} = \text{yes}, \text{Credit\_rating} = \text{Fair})$

$$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

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

# 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, where,

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

$$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

age	income	student	credit_rating	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

# Naïve Bayes Classifier: An Example

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

□  $P(C_i): P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$

$P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$

□ Compute  $P(X_i | C_i)$  for each class

$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$

$P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

□  **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**

**$P(X | C_i): P(X | \text{buys\_computer} = \text{"yes"}) = P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) * P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) * P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) * P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"})$**

# Naïve Bayes Classifier: An Example

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

□  $P(C_i): P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$

$P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$

□ Compute  $P(X_i | C_i)$  for each class

$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$

$P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

□  **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**

**$P(X | C_i): P(X | \text{buys\_computer} = \text{"yes"}) = P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) * P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) * P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) * P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$**



# Naïve Bayes Classifier: An Example

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

□  $P(C_i): P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$

$P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$

□ Compute  $P(X_i | C_i)$  for each class

$P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$

$P(\text{age} = \text{"<= 30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$

$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$

$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$

□  **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**

**$P(X | \text{buys\_computer} = \text{"yes"}) = P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) * P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) * P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) * P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$**

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

# Naïve Bayes Classifier: An Example

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

- $P(C_i)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute  $P(X_i | C_i)$  for each class  
 $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$   
 $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$   
 $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$   
 $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$   
 $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$   
 $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
- **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**   
 $P(X | C_i) : P(X | \text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$   
 $P(X | \text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
- $P(X | C_i) * P(C_i) : P(X | \text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$**   
 $P(X | \text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$

# Naïve Bayes Classifier: An Example

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

- $P(C_i)$ :  $P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$   
 $P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$
  - Compute  $P(X_i | C_i)$  for each class  
 $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$   
 $P(\text{age} = \text{"<=30"} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$   
 $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$   
 $P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$   
 $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$   
 $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$   
 $P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$
  - **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$**   
 $P(X | C_i) : P(X | \text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$   
 $P(X | \text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$   
 $P(X | C_i) * P(C_i) : P(X | \text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$   
 $P(X | \text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$
- Since **Red** > **Blue** here, **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

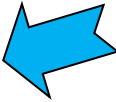
$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- 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

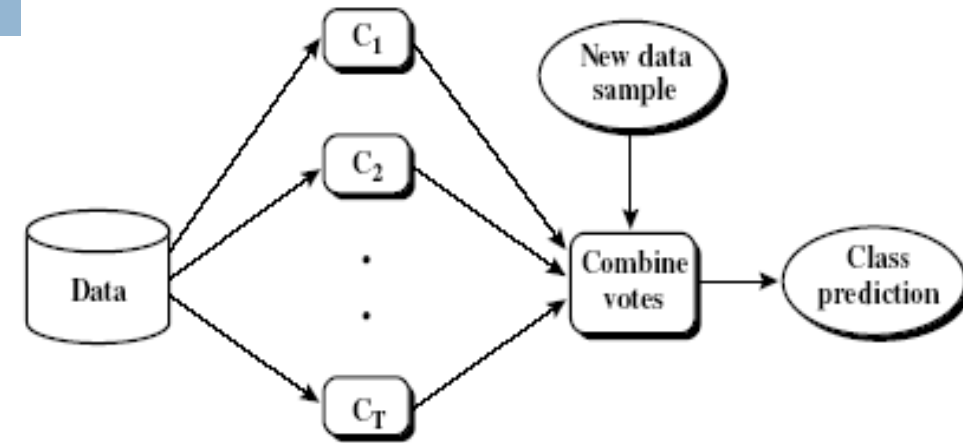
# Naïve Bayes Classifier: Comments

- 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)**

# Classification: Basic Concepts

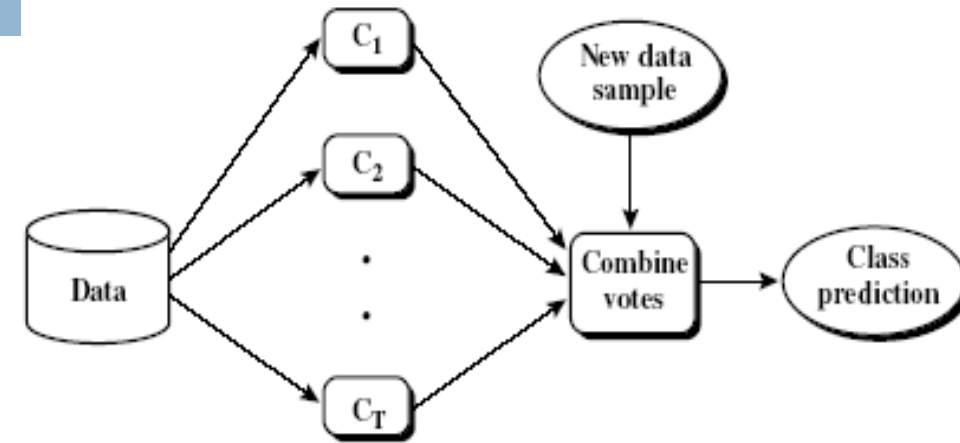
- Classification: Basic Concepts
- Decision Tree Induction
- Model Evaluation and Selection
- Practical Issues of Classification
- Bayes Classification Methods
- Techniques to Improve Classification Accuracy: Ensemble Methods 
- Summary

# 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  $M^*$

# Ensemble Methods: Increasing the Accuracy

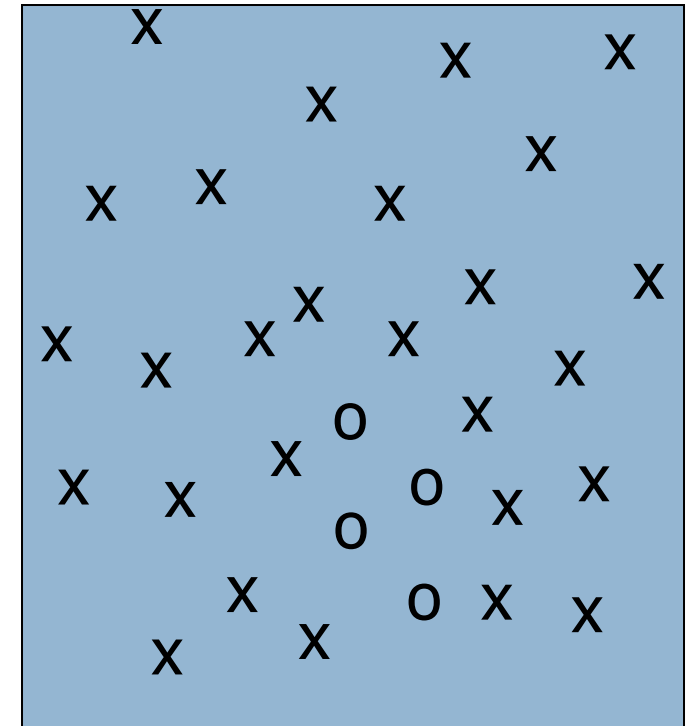


- Ensemble methods
  - ▣ Use **a combination of models** to increase accuracy
  - ▣ Combine a series of  $k$  learned models,  $M_1, M_2, \dots, M_k$ , with the aim of creating an improved model  $M^*$
  
- Popular ensemble methods
  - ▣ **Bagging**: averaging the prediction over a collection of classifiers
  - ▣ **Boosting**: weighted vote with a collection of classifiers
  - ▣ Random forests: Imagine that each of the classifiers in the ensemble is a decision tree classifier so that the collection of classifiers is a “forest”



# 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
- Still difficult for class imbalance problem on multiclass tasks



66

# Backup Slides

# Other Issues

---

- Data Fragmentation
- Search Strategy
- Expressiveness

# Data Fragmentation

---

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

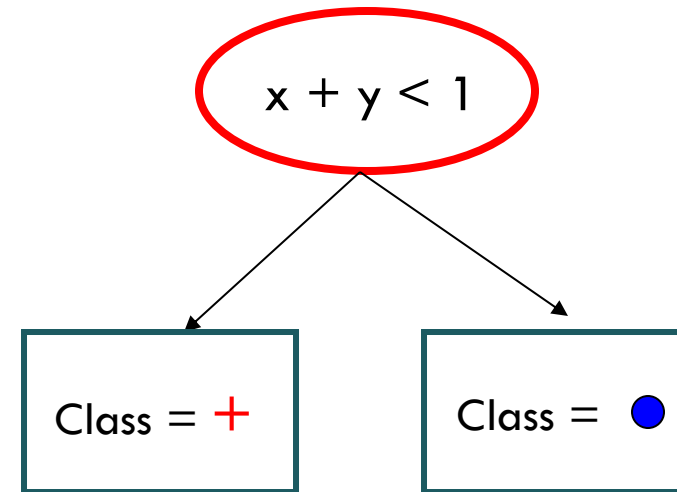
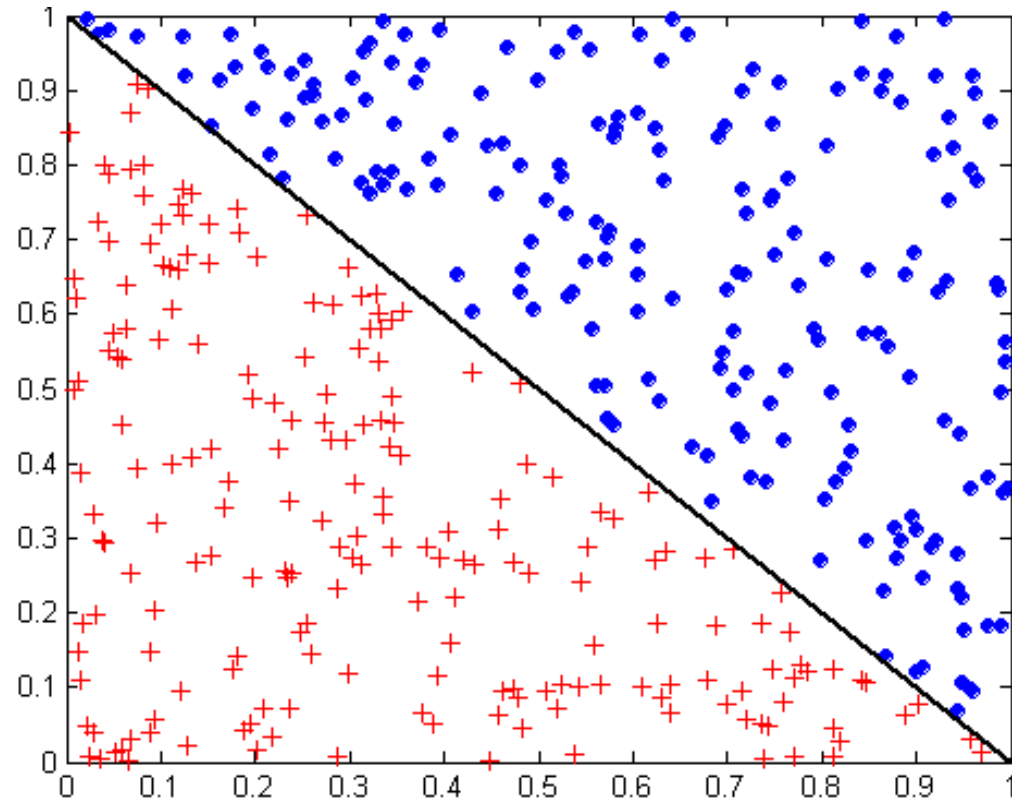
# Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - ▣ Bottom-up
  - ▣ Bi-directional

# Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - ▣ But they do not generalize well to certain types of Boolean functions
    - Example: XOR or Parity functions (example in book)
- Not expressive enough for modeling continuous variables
  - ▣ Particularly when test condition involves only a single attribute at-a-time

# Expressiveness: Oblique Decision Trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive
- Needs multi-dimensional discretization

# 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:  $err(\mathbf{X}_j)$  is the misclassification error of tuple  $\mathbf{X}_j$ . Classifier  $M_i$  error rate is the sum of the weights of the misclassified tuples:
$$error(M_i) = \sum_j w_j \times err(\mathbf{X}_j)$$
- The weight of classifier  $M_i$ 's vote is  $\log \frac{1 - error(M_i)}{error(M_i)}$

# 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