



# **CS 412 Intro. to Data Mining**

## **Chapter 8. Classification: Basic Concepts**

**Jiawei Han, Computer Science, Univ. Illinois at Urbana-Champaign, 2017**





# Chapter 8. Classification: Basic Concepts

---

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



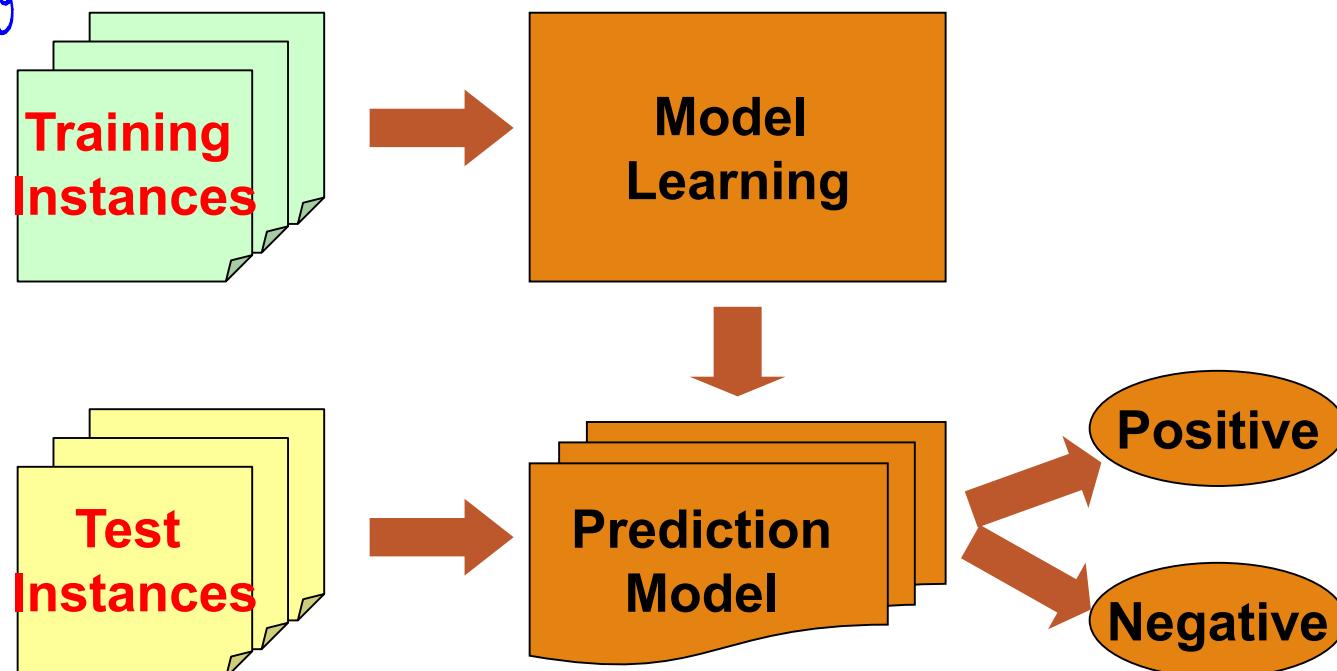
# Supervised vs. Unsupervised Learning (1)

- Supervised learning (classification) → **监督式模型训练**
- Supervision: The training data such as observations or measurements are accompanied by **labels** indicating the classes which they belong to
- New data is classified based on the models built from the training set

Labels  $\Rightarrow$  Training data = Model Training

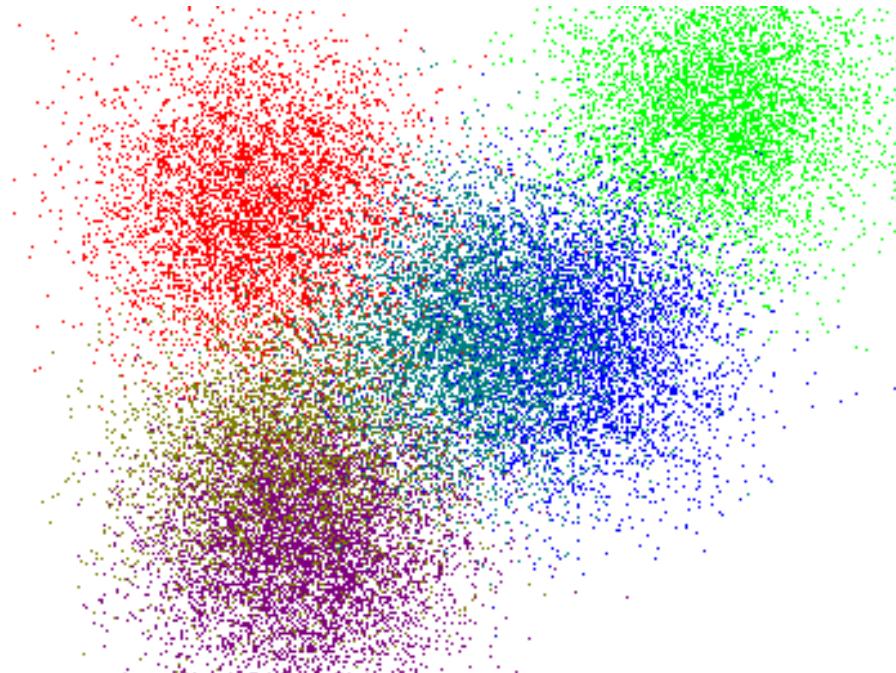
Training Data with class label:

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



# Supervised vs. Unsupervised Learning (2)

- Unsupervised learning (clustering) → គេងរករាយ Model ទូទៅដែលមិនត្រូវ Training
  - The class labels of training data are unknown
  - Given a set of observations or measurements, establish the possible existence of classes or clusters in the data



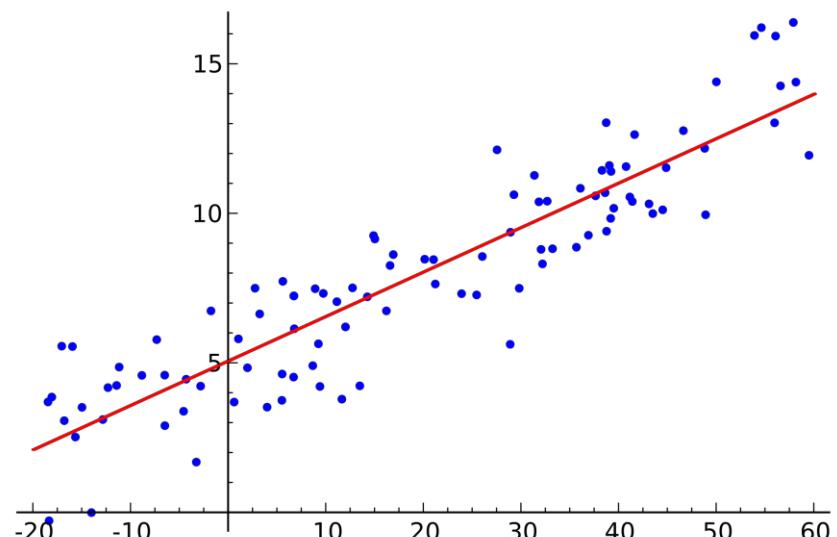
គេងនូវការណ៍ Clustering បែនលក់

# Prediction Problems: Classification vs. Numeric Prediction

- Classification ស្នាត់ដើម្បីគាំពូកថាដែលរបៀបរាយ

  - Predict categorical class labels (discrete or nominal)
  - Construct a model based on the training set and the **class labels** (the values in a classifying attribute) and use it in classifying new data

- Numeric prediction
  - Model continuous-valued functions (i.e., predict unknown or missing values)
  - Typical applications of classification
    - 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—Model Construction, Validation and Testing

---

- **Model construction**
  - Each sample is assumed to belong to a predefined class (shown by the **class label**)
  - The set of samples used for model construction is **training set**
  - Model: Represented as decision trees, rules, mathematical formulas, or other forms
- **Model Validation and Testing:**
  - **Test:** Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - **Accuracy:** % of test set samples that are correctly classified by the model
    - Test set is independent of training set
  - **Validation:** If *the test set* is used to select or refine models, it is called **validation (or development) (test) set**
- **Model Deployment:** If the accuracy is acceptable, use the model to classify new data

# **Chapter 8. Classification: Basic Concepts**

---

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



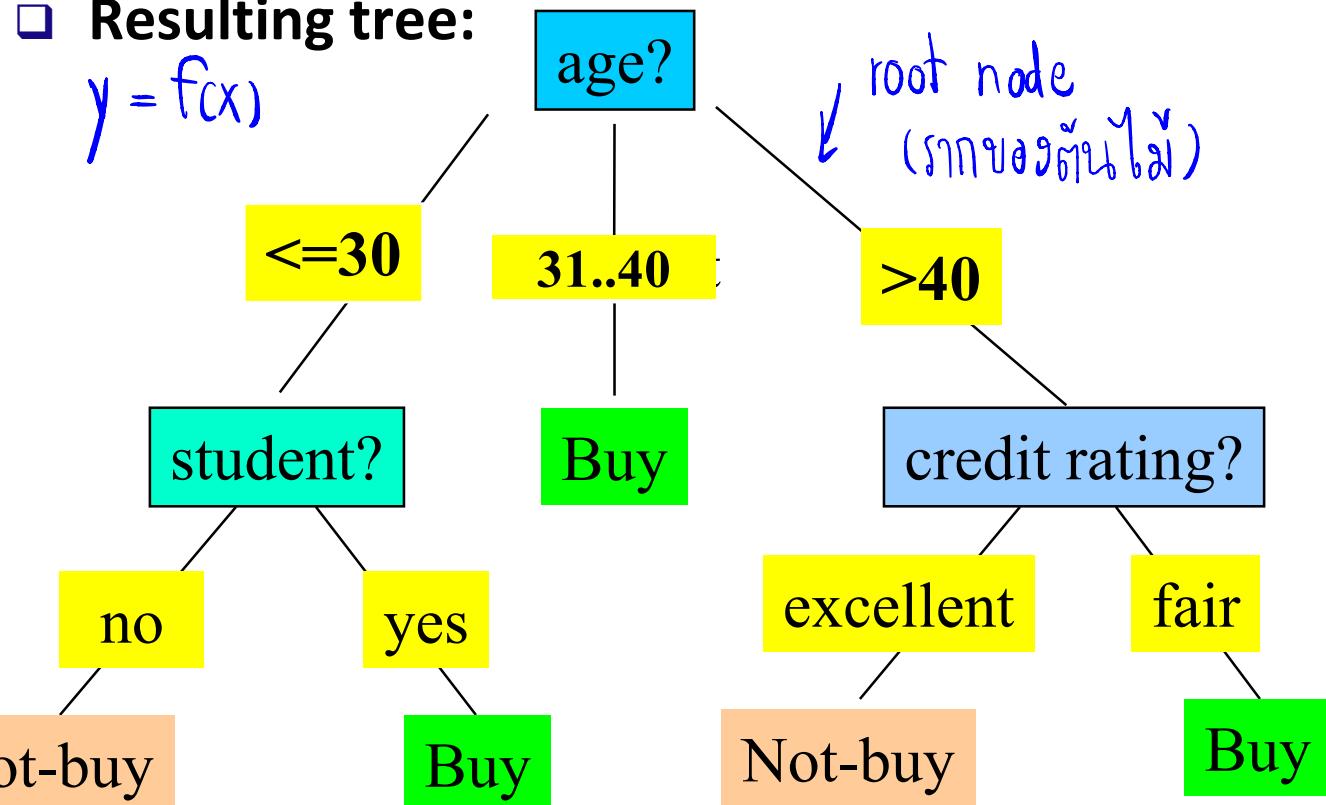
# Decision Tree Induction: An Example

## □ Decision tree construction:

- A top-down, recursive, divide-and-conquer process

## □ Resulting tree:

$$y = f(x)$$



សៀវភៅ  
x (feature)

Training data set: Who buys computer?

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

Note: The data set is adapted from "Playing Tennis" example of R. Quinlan

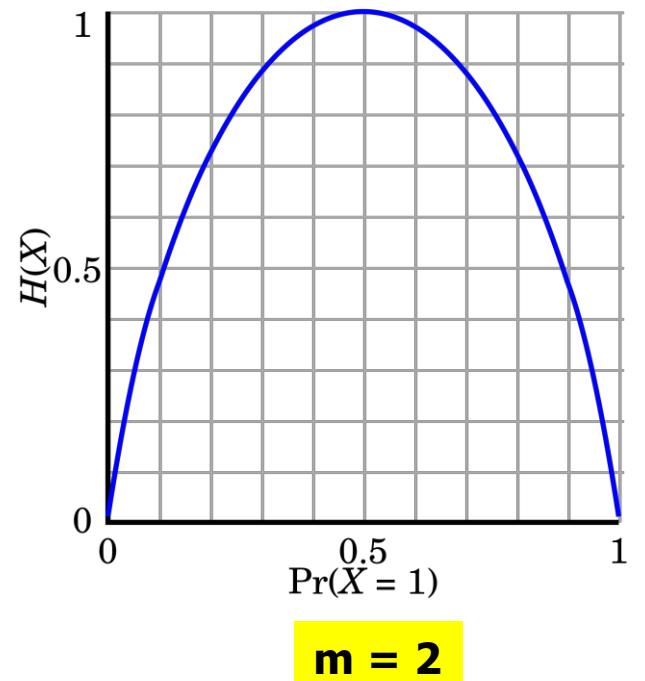
# From Entropy to Info Gain: A Brief Review of Entropy

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

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

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

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



# Information Gain: An Attribute Selection Measure

- Select the attribute with the highest information gain (used in typical decision tree induction algorithm: ID3/C4.5)
- 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

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

ณ node ที่  $\frac{1}{2}$  ก่อสร้าง

# Example: Attribute Selection with 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	$p_i$	$n_i$	$I(p_i, n_i)$
$\leq 30$	2	3	0.971
31...40	4	0	0
$>40$	3	2	0.971

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

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

$\frac{5}{14} I(2,3)$  means "age  $\leq 30$ " 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, we can get

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

# Decision Tree Induction: Algorithm

---

- Basic algorithm
  - Tree is constructed in a **top-down, recursive, divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - On each node, attributes are selected based on the training examples on that node, and 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
  - There are no samples left
- Prediction
  - **Majority voting** is employed for classifying the leaf

# How to Handle Continuous-Valued Attributes?

---

- Method 1: Discretize continuous values and treat them as categorical values
  - E.g., age: < 20, 20..30, 30..40, 40..50, > 50
- Method 2: Determine the *best split point* for continuous-valued attribute A
  - Sort the value A in increasing order: e.g. 15, 18, 21, 22, 24, 25, 29, 31, ...
  - *Possible split point*: the midpoint between *each pair of adjacent values*
    - $(a_i + a_{i+1})/2$  is the midpoint between the values of  $a_i$  and  $a_{i+1}$
    - e.g.,  $(15+18)/2 = 16.5, 19.5, 21.5, 23, 24.5, 27, 30, \dots$
  - The point with the *maximum information gain* for A is selected as the **split-point** for A
- Split: Based on split point P
  - The set of tuples in D satisfying  $A \leq P$  vs. those with  $A > P$

# Gain Ratio: A Refined Measure for Attribute Selection

---

- Information gain measure is biased towards attributes with a large number of values
- Gain ratio: Overcomes the problem (as a normalization to information gain)

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

- $\text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}(A)$
- The attribute with the maximum gain ratio is selected as the splitting attribute
- Gain ratio is used in a popular algorithm C4.5 (a successor of ID3) by R. Quinlan
- Example
  - $\text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \log_2 \frac{4}{14} - \frac{6}{14} \log_2 \frac{6}{14} - \frac{4}{14} \log_2 \frac{4}{14} = 1.557$
  - $\text{GainRatio}(\text{income}) = 0.029/1.557 = 0.019$

# Another Measure: Gini Index

---

- Gini index: Used in CART, and also in IBM IntelligentMiner
- If a data set  $D$  contains examples from  $n$  classes, gini index,  $gini(D)$  is defined as
  - $$gini(D) = 1 - \sum_{j=1}^n p_j^2$$
  - $p_j$  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

---

- Example: D has 9 tuples in buys\_computer = “yes” and 5 in “no”

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

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

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

- Gini<sub>{low,high}</sub> is 0.458; Gini<sub>{medium,high}</sub> is 0.450
  - Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index
- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

# Comparing Three Attribute Selection Measures

---

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

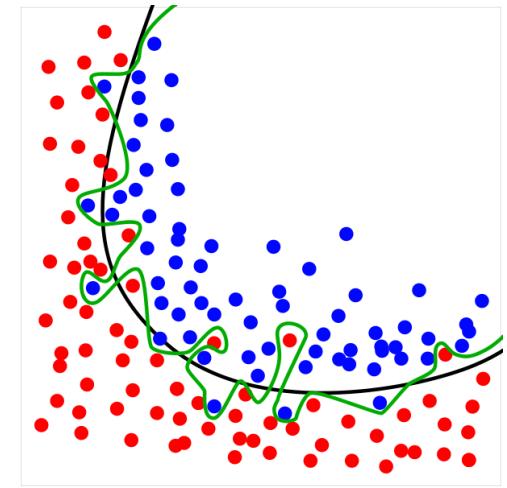
# Other Attribute Selection Measures

---

- Minimal Description Length (MDL) principle
  - Philosophy: 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
- CHAID: a popular decision tree algorithm, measure based on  $\chi^2$  test for independence
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear combination of attributes
- There are many other measures proposed in research and applications
  - E.g., G-statistics, C-SEP
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

# Overfitting and Tree Pruning

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



# Classification in Large Databases

---

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - Relatively fast learning speed
  - Convertible to simple and easy to understand classification rules
  - Easy to be adapted to database system implementations (e.g., using SQL)
  - Comparable classification accuracy with other methods
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

# RainForest: A Scalable Classification Framework

- The criteria that determine the quality of the tree can be computed separately
  - Builds an AVC-list: **AVC (Attribute, Value, Class\_label)**
- **AVC-set** (of an attribute  $X$ )
  - Projection of training dataset onto the attribute  $X$  and class label where counts of individual class label are aggregated
- **AVC-group** (of a node  $n$ )
  - Set of AVC-sets of all predictor attributes at the node  $n$

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

The Training Data

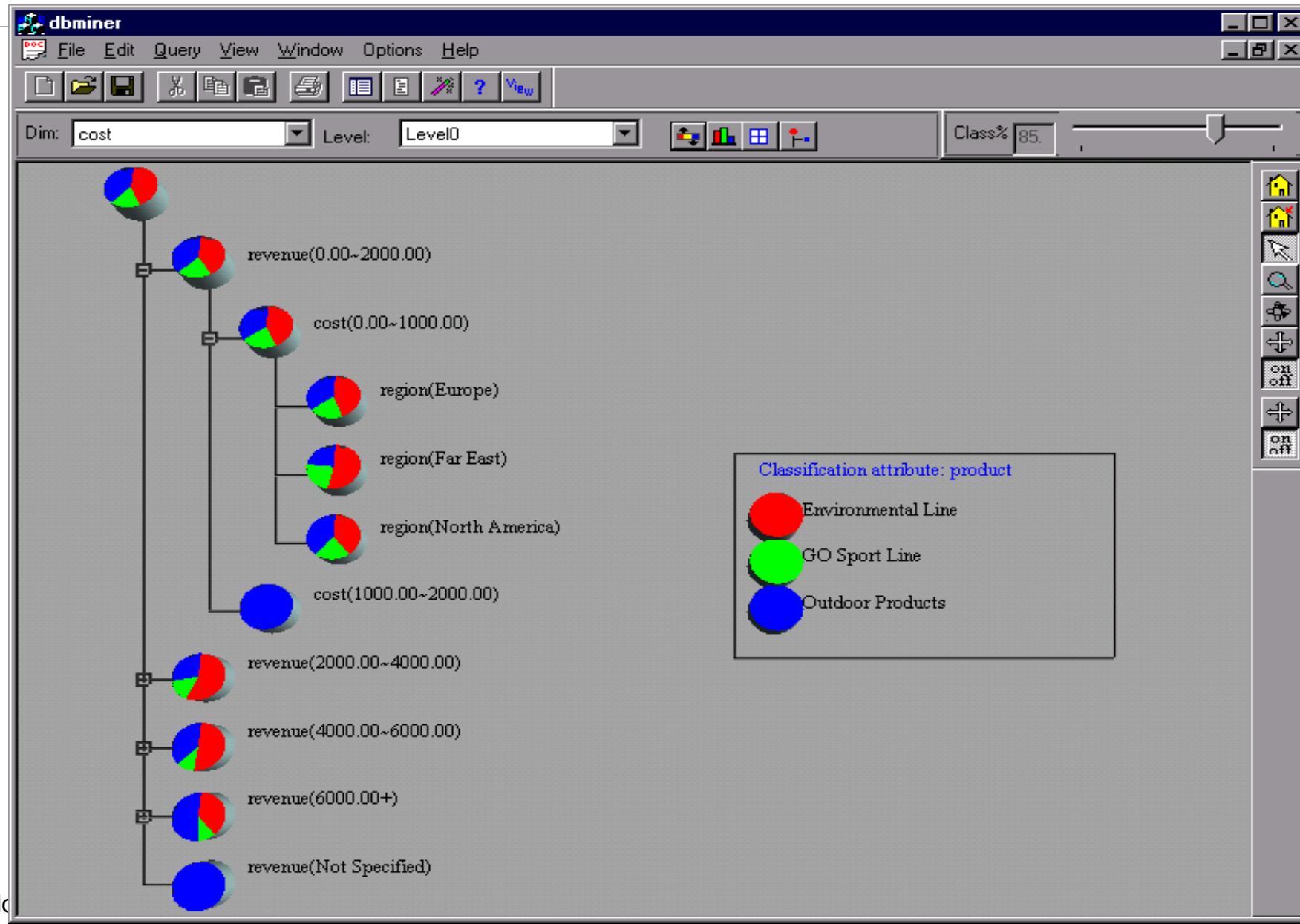
AVC-set on Age		
Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on Income		
income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

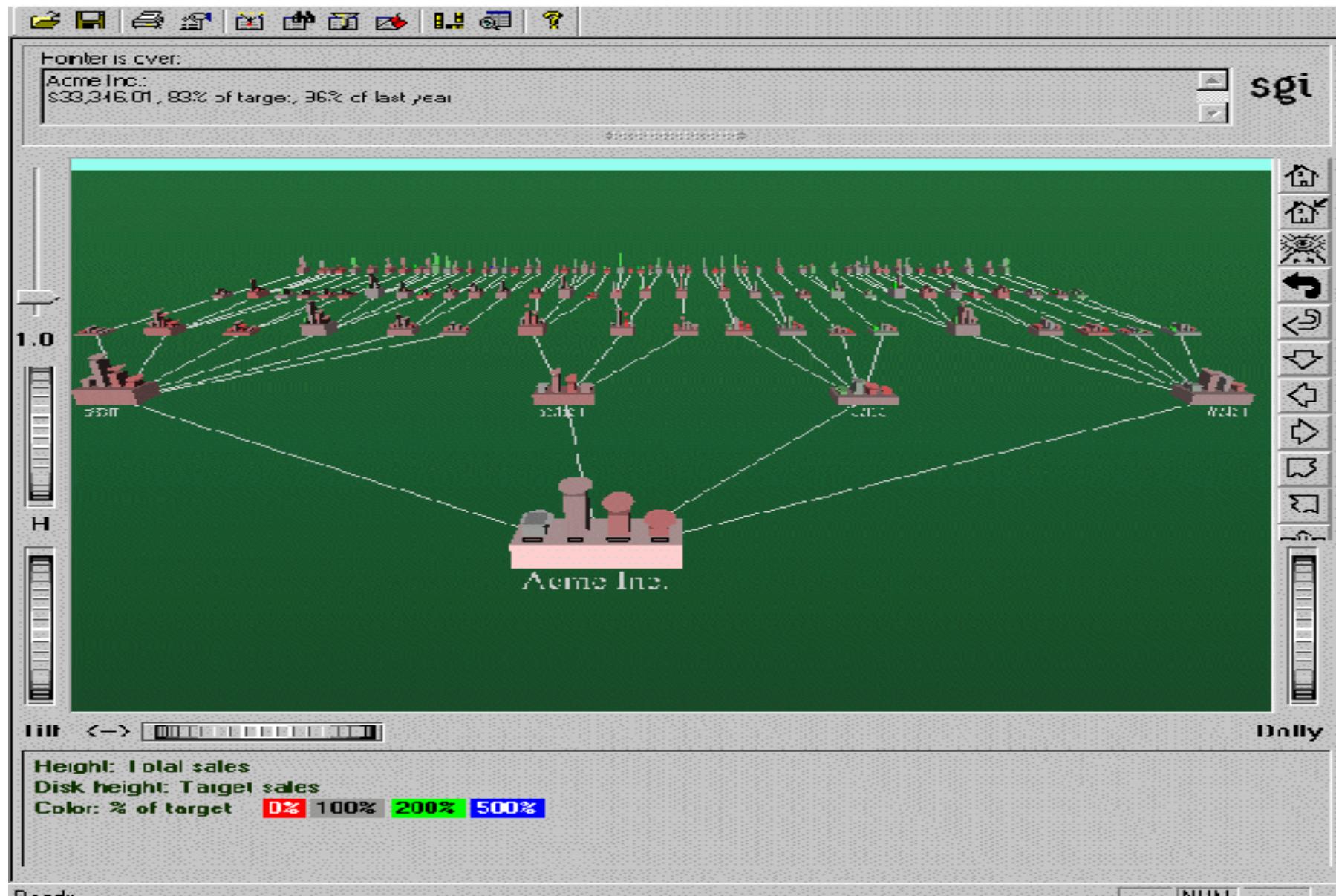
AVC-set on Student		AVC-set on Credit_Rating			
student	Buy_Computer		Credit rating	Buy_Computer	
	yes	no		yes	no
yes	6	1	fair	6	2
	3	4		3	3

Its AVC Sets

# Presentation of Classification Results

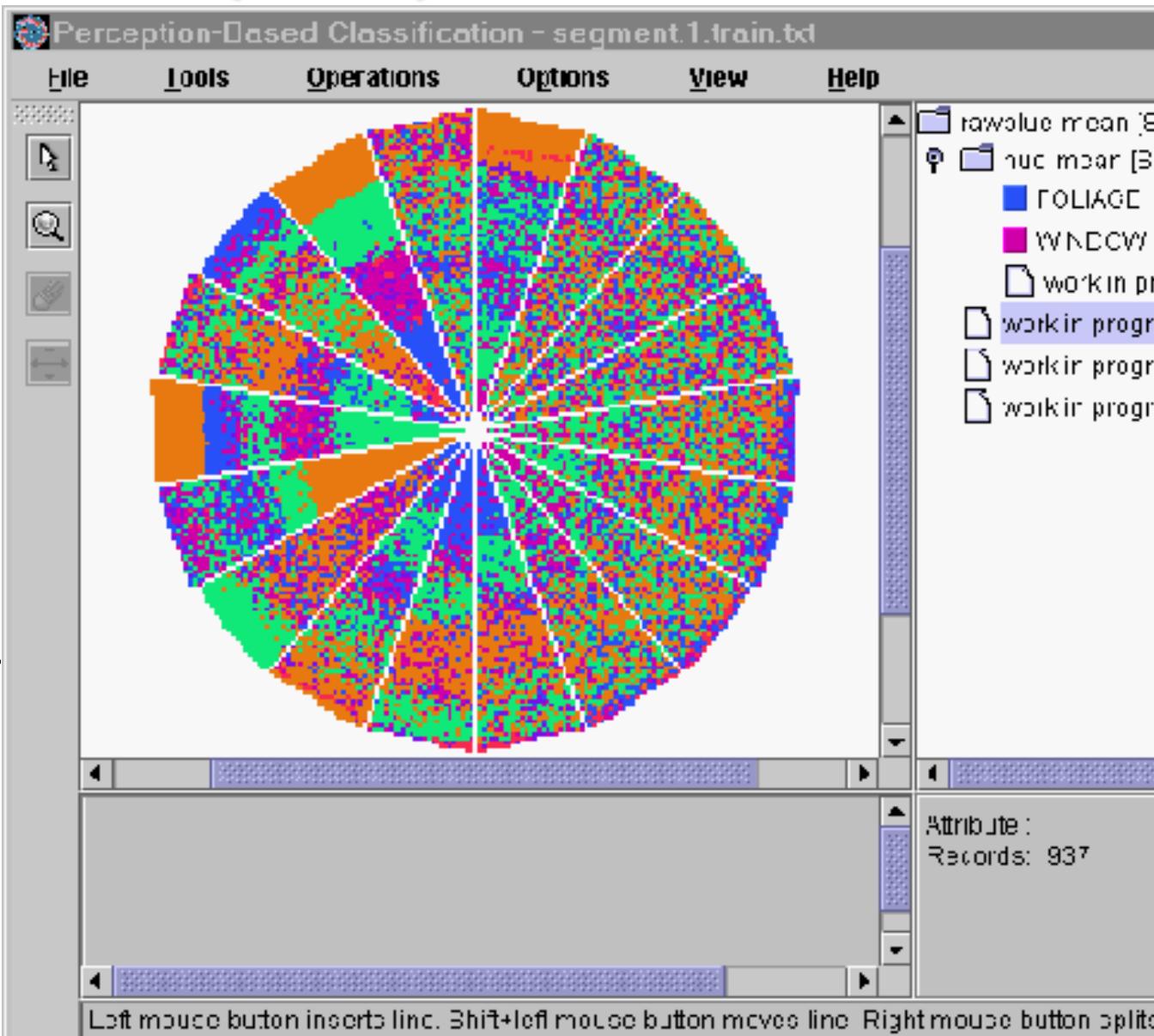


# Visualization of a Decision Tree (in SGI/MineSet 3.0)



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

- ❑ Perception-based classifier (PCB): developed at Univ. of Munich (1999)
- ❑ One color represents one class label
- ❑ One pie represents one attribute (or variable)
- ❑ The pie with random spread implies weak classification power
- ❑ The pie with clearly partitioned color strips implies good classification power
- ❑ One can select a good attribute and regenerate new pie charts for classification at the subsequent levels

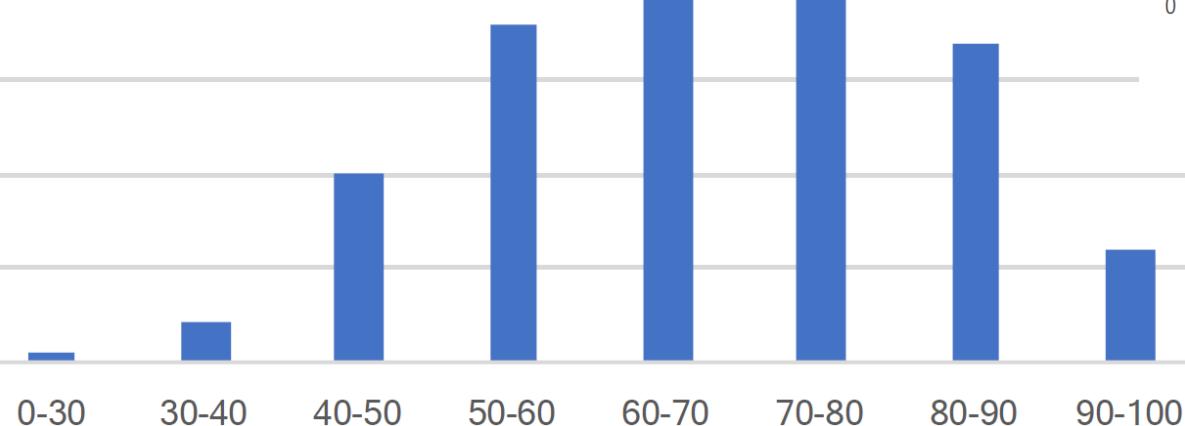


# CS412-Fall 2017: Midterm Statistics

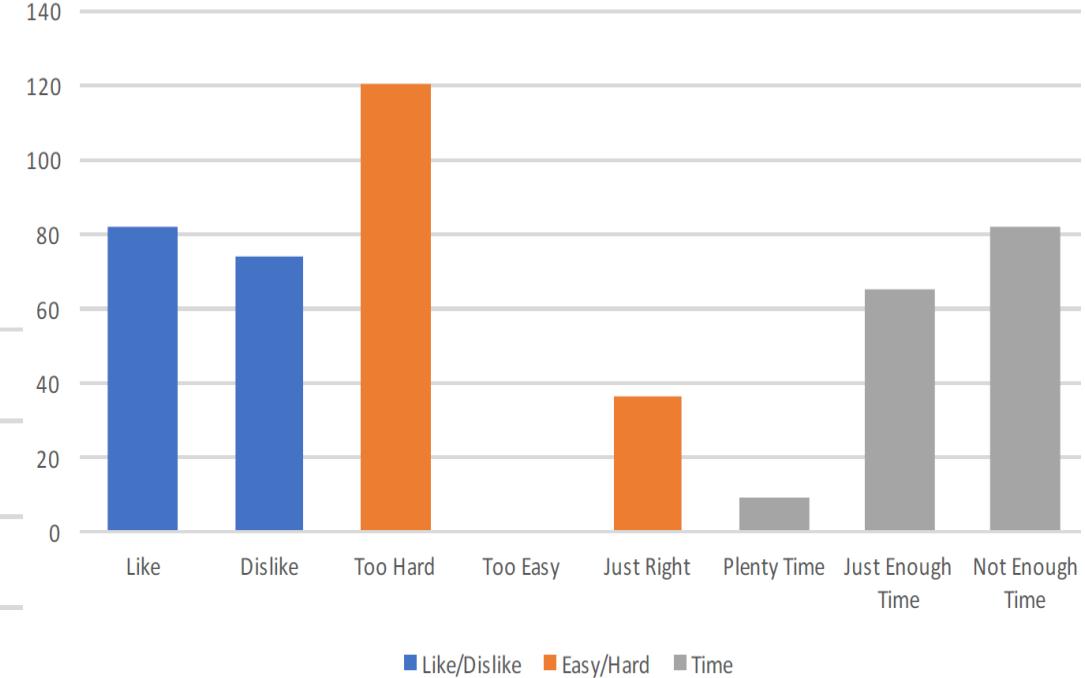
Range	Count
0-30	1
30-40	4
40-50	20
50-60	35
60-70	45
70-80	50
80-90	32
90-100	13

Mean 68.64  
Median 69.5  
1st quartile 57.75  
3rd quartile 79.5

Midterm Scores



Midterm Options



# **Chapter 8. Classification: Basic Concepts**

---

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



# What Is Bayesian Classification?

---

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

# Bayes' Theorem: Basics

- Total probability Theorem:

$$p(B) = \sum_i p(B|A_i)p(A_i)$$

- Bayes' Theorem:

$$p(H|X) = \frac{p(X|H)p(H)}{p(X)}$$

*test data*      *training data*

posterior probability      likelihood      prior probability

What we should choose

What we just see

What we knew previously

- $X$ : a data sample ("evidence")

Prediction can be done based on Bayes' Theorem:

- $H$ :  $X$  belongs to class C

Classification is to derive the maximum posterior

# Naïve Bayes Classifier: Making a Naïve Assumption

- Practical difficulty of Naïve Bayes inference: It requires initial knowledge of many probabilities, which may not be available or involving significant computational cost
- A Naïve Special Case
  - Make an additional assumption to simplify the model, but achieve comparable performance.

attributes are conditionally independent

(i.e., no dependence relation between attributes)

ពួរក្នុងកែយវ៉ាទំ | តម្លៃអាជីវការ

ភូមិសាស្ត្រ

$$p(X|C_i) = \prod_k p(x_k|C_i) = p(x_1|C_i) \cdot p(x_2|C_i) \cdots p(x_n|C_i)$$

- Only need to count the class distribution w.r.t. features

# Naïve Bayes Classifier: Categorical vs. Continuous Valued Features

---

- If feature  $x_k$  is categorical,  $p(x_k = v_k | C_i)$  is the # of tuples in  $C_i$  with  $x_k = v_k$ , divided by  $|C_{i,D}|$  (# of tuples of  $C_i$  in D)

$$p(X|C_i) = \prod_k p(x_k|C_i) = p(x_1|C_i) \cdot p(x_2|C_i) \cdots p(x_n|C_i)$$

- If feature  $x_k$  is continuous-valued,  $p(x_k = v_k | C_i)$  is usually computed based on Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$

$$p(x_k = v_k | C_i) = N(x_k | \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x-\mu_{C_i})^2}{2\sigma^2}}$$

# Naïve Bayes Classifier: Training Dataset

Class:

C1:`buys_computer = 'yes'`

C2:`buys_computer = 'no'`

Data to be classified:

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

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

$$P(H^y | x) = ?$$

$$P(H^N | x) = ?$$

training data

$$= P(x | H^y) P(H^y)$$

ទេរក

prior probability

$$\text{ដែលការសម្រេច } yes = \frac{9}{14}$$

ពេលវេលាដំឡើងតាម yes និយោគនឹង x

ដែលវេលាដំឡើងតាម x និយោគនឹង yes និយោគនឹង y

$$\hat{x} = \text{age} = 42, \text{student} = \text{yes}?$$

$$P(H^y | \hat{x}) = ?$$

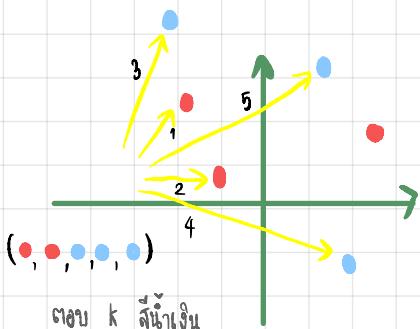
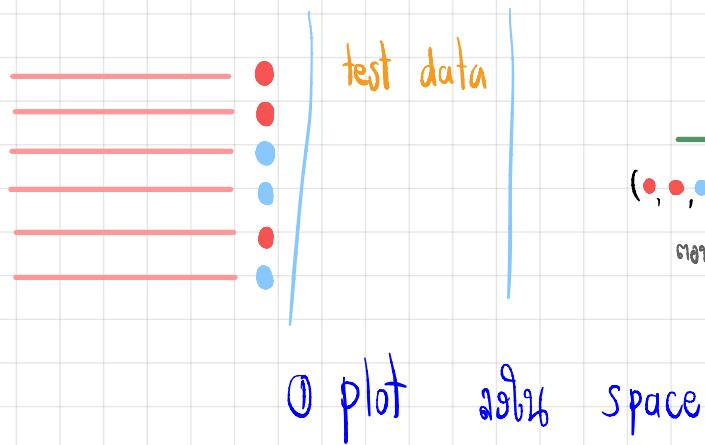
$$P(H=y_{\text{buy}} | (\text{age} = 42, \text{student} = \text{yes})) = P(\text{age} = 42 | y) P(\text{student} = \text{yes} | y_{\text{buy}})$$

$$P(y_{\text{buy}}) \quad \frac{9}{14}$$

$$P(H_{\text{buy}} = N | (\text{age} = 42, \text{student} = \text{yes})) =$$

# K - NN

## K - Nearest Neighbors



② ພາມເຜື່ອນປັນ k ອານ  
( $k = 3$ )

ພາມເຜື່ອນປັນທີ່ປົກປົງໄສແລ້ວກຳຈະຕອບໄດ້

\* k ອະນຸມິດນຸລັບຄໍ

# Naïve Bayes Classifier: An Example

ເຊື້ອ ອີເວມ yes = <sup>(age)</sup>  $\leq 30$  ກໍາລັງ | ອີເວມ <sup>(income)</sup> medium ອິດຍ yes ກໍາລັງ

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

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

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

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

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

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

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

$$0.044 \times 0.643$$

Therefore, X belongs to class ("buys\_computer = yes")

# Avoiding the Zero-Probability Problem

---

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

$$p(X|C_i) = \prod_k p(x_k|C_i) = p(x_1|C_i) \cdot p(x_2|C_i) \cdots p(x_n|C_i)$$

- Example. 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*

$$\text{Prob}(\text{income} = \text{low}) = 1/(1000 + 3)$$

$$\text{Prob}(\text{income} = \text{medium}) = (990 + 1)/(1000 + 3)$$

$$\text{Prob}(\text{income} = \text{high}) = (10 + 1)/(1000 + 3)$$

- The “corrected” probability estimates are close to their “uncorrected” counterparts

# Naïve Bayes Classifier: Strength vs. Weakness

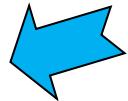
---

- ❑ Strength
  - ❑ Easy to implement
  - ❑ Good results obtained in most of the cases
- ❑ Weakness
  - ❑ Assumption: attributes conditional independence, therefore loss of accuracy
  - ❑ Practically, dependencies exist among variables
    - ❑ E.g., 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?
    - ❑ Use Bayesian Belief Networks (to be covered in the next chapter)

# **Chapter 8. Classification: Basic Concepts**

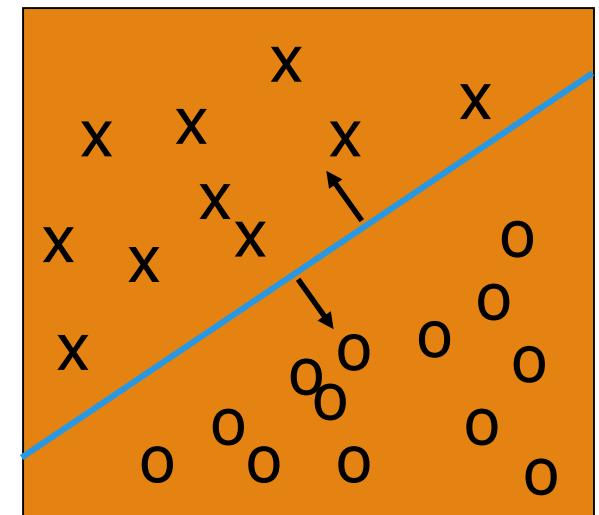
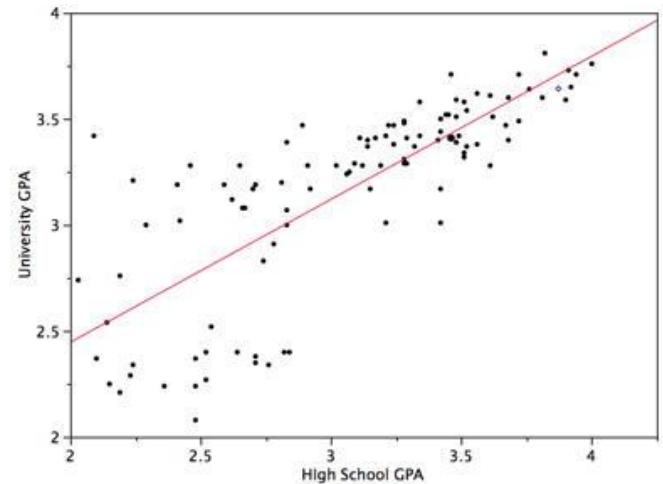
---

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



# Linear Regression vs. Linear Classifier

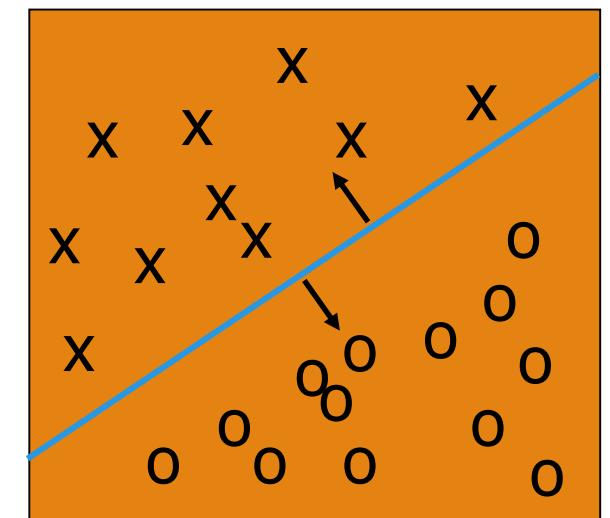
- ❑ Linear regression
  - ❑ Data modeled to fit a straight line
    - ❑ *Linear equation:*  $Y = w X + b$
  - ❑ Often uses the least-square method to fit the line
  - ❑ Used to predict continuous values
  
- ❑ Linear Classifier
  - ❑ Built a classification model using a straight line
  - ❑ Used for (categorical data) binary classification



# Linear Classifier: General Ideas

---

- ❑ Binary Classification
- ❑  $f(x)$  is a linear function based on the example's attribute values
  - ❑ The prediction is based on the value of  $f(x)$
  - ❑ Data above the blue line belongs to class 'x' (i.e.,  $f(x) > 0$ )
  - ❑ Data below blue line belongs to class 'o' (i.e.,  $f(x) < 0$ )
- ❑ Classical Linear Classifiers
  - ❑ Linear Discriminant Analysis (LDA) (not covered)
  - ❑ Logistic Regression
  - ❑ Perceptron (later)
  - ❑ SVM (later)



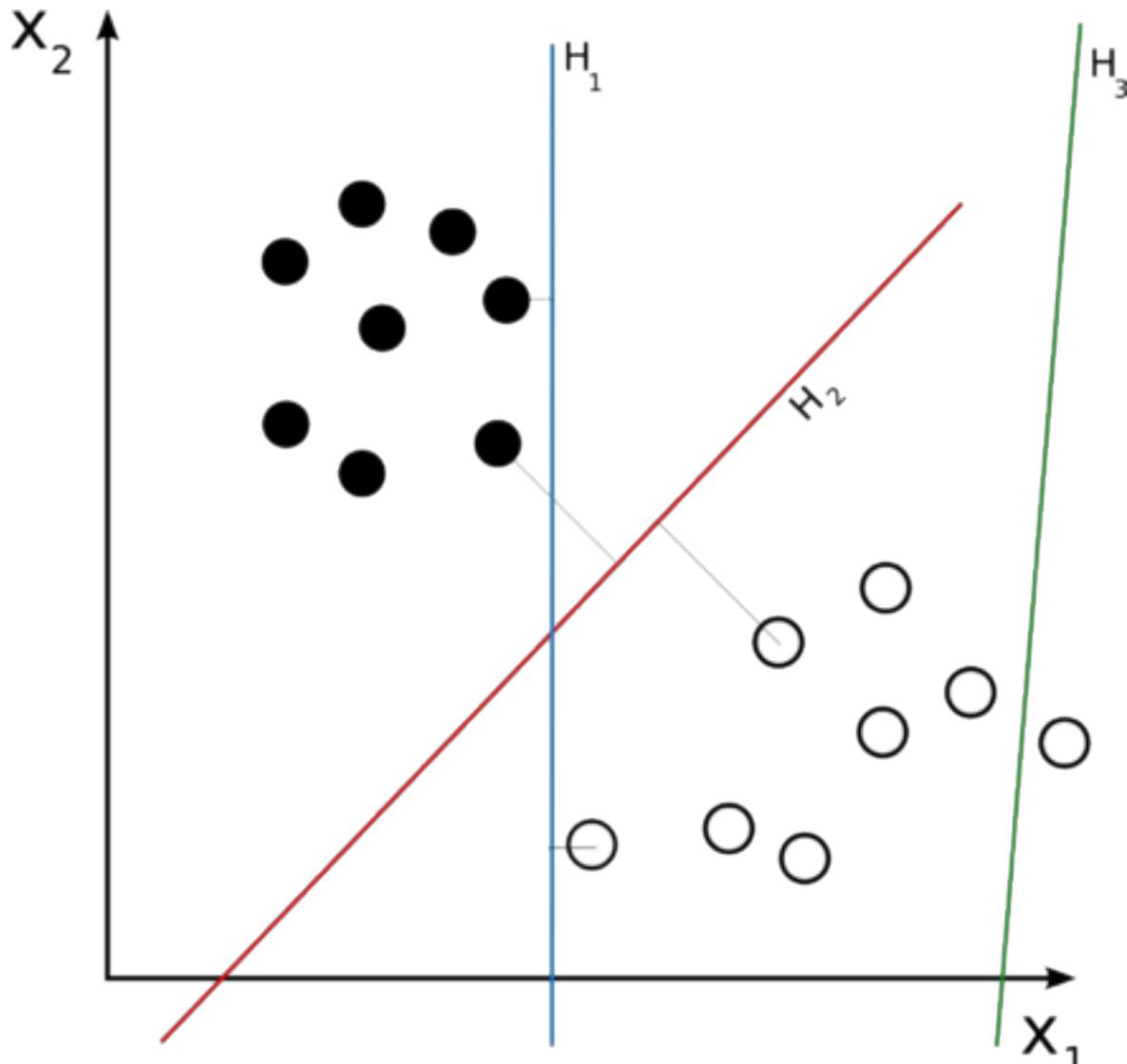
# Linear Classifier: An Example

---

- A toy rule to determine whether a faculty member has tenure
  - Year  $\geq 6$  or Title = “Professor”  $\Leftrightarrow$  Tenure
- How to express the rule as a linear classifier?
- Features
  - $x_1 (x_1 \geq 0)$  is an integer denoting the year
  - $x_2$  is a Boolean denoting whether the title is “Professor”
- A feasible linear classifier:  $f(x) = (x_1 - 5) + 6 \cdot x_2$ 
  - When  $x_2$  is True, because  $x_1 \geq 0$ ,  $f(x)$  is always greater than 0
  - When  $x_2$  is False, because  $f(x) > 0 \Leftrightarrow x_1 \geq 6$
- There are many more feasible classifiers
  - $f(x) = (x_1 - 5.5) + 6 \cdot x_2$
  - $f(x) = 2 \cdot (x_1 - 5) + 11 \cdot x_2$
  - .....

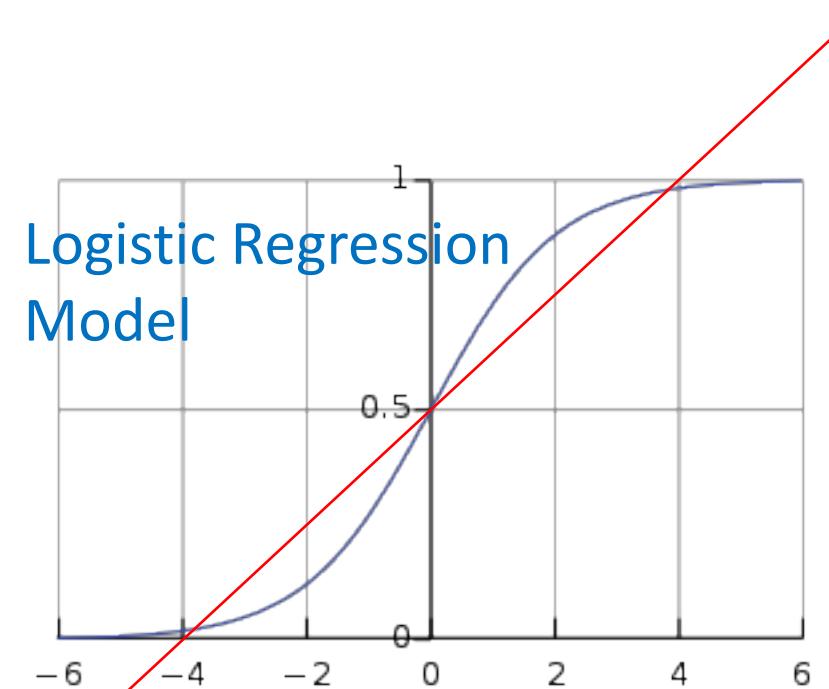
# Key Question: Which Line Is Better?

- ❑ There might be many feasible linear functions
  - ❑ Both  $H_1$  and  $H_2$  will work
- ❑ Which one is better?
  - ❑  $H_2$  looks “better” in the sense that it is also furthest from both groups
  - ❑ We will introduce more in the SVM section



# Logistic Regression: General Ideas

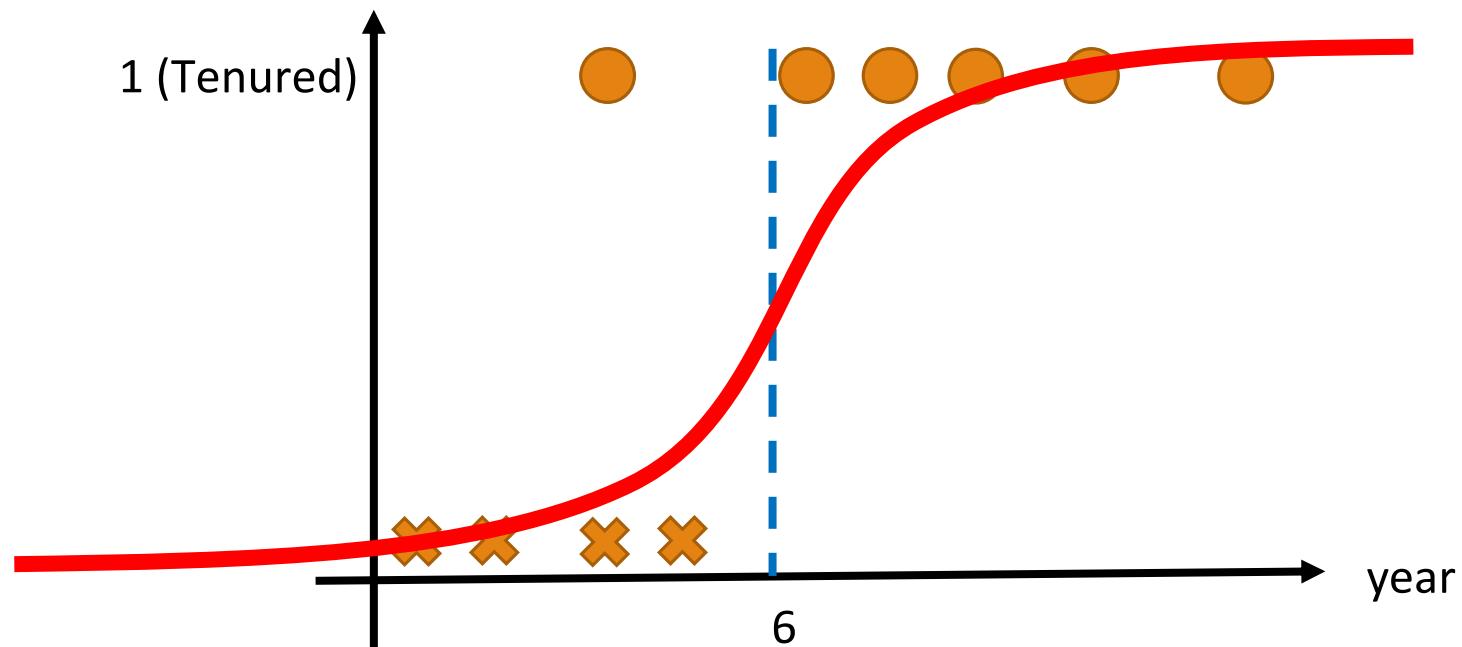
- Key Idea: Turns linear predictions into probabilities
- Sigmoid function:
  - $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$
  - Projects  $(-\infty, +\infty)$  to  $[0, 1]$
- Compare to linear probability model
- More smooth



Linear Probability  
Model

# Logistic Regression: An Example

- ❑ Suppose we only consider the year as feature



# Logistic Regression: Maximum Likelihood

---

- ❑ The prediction function to learn

- ❑  $p(Y = 1 | X = x; \mathbf{w}) = S(w_0 + \sum_{i=1}^n w_i \cdot x_i)$
  - ❑  $\mathbf{w} = (w_0, w_1, w_2, \dots, w_n)$  are the parameters

- ❑ Maximum Likelihood

- ❑ Log likelihood:

$$l(\mathbf{w}) = \sum_{i=1}^N y_i \log p(Y = 1 | X = x_i; \mathbf{w}) + (1 - y_i) \log(1 - p(Y = 1 | X = x_i; \mathbf{w}))$$

- ❑ There's no close form solution

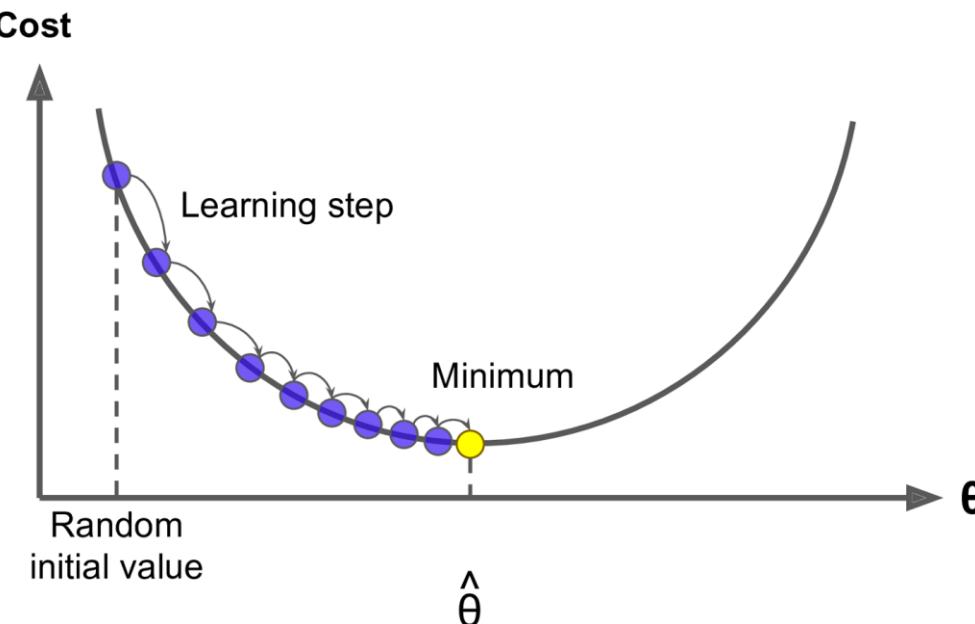
- ❑ Gradient Descent
  - ❑ Update  $\mathbf{w}$  based on training data
  - ❑ Chain-rule for the gradient

# Gradient Descent

- Gradient Descent is an iterative optimization algorithm for finding the minimum of a function (e.g., the negative log likelihood)
- For a function  $F(x)$  at a point  $a$ ,  $F(x)$  decreases fastest if we go in the direction of the negative gradient of  $a$

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

When the gradient is zero, we arrive at the local minimum



# Generative vs. Discriminative Classifiers

---

- X: observed variables (features)
- Y: target variables (class labels)
- A generative classifier models  $p(Y, X)$ 
  - It models how the data was "generated"? "what is the likelihood this or that class generated this instance?" and pick the one with higher probability
- Naïve Bayes
- Bayesian Networks
- A discriminative classifier models  $p(Y | X)$ 
  - It uses the data to create a decision boundary
- Logistic Regression
- Support Vector Machines

# Further Comments on Discriminative Classifiers

---

- Strength
  - Prediction accuracy is generally high
  - As compared to generative models
  - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
    - Comparing to [\(covered in future\)](#) Bayesian networks (which are normally slow)
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

# **Chapter 8. Classification: Basic Concepts**

---

- ❑ Classification: Basic Concepts
- ❑ Decision Tree Induction
- ❑ Bayes Classification Methods
- ❑ Linear Classifier
- ❑ Model Evaluation and Selection
- ❑ Techniques to Improve Classification Accuracy: Ensemble Methods
- ❑ Additional Concepts on Classification
- ❑ 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
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - 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)

□ In a confusion matrix w.  $m$  classes,  $CM_{i,j}$  indicates # of tuples in class  $i$  that were labeled by the classifier as class  $j$

□ May have extra rows/columns to provide totals

## □ 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

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

---

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

- Classifier accuracy, or recognition rate
  - Percentage of test set tuples that are correctly classified
$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{All}$$
- 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
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - $\text{Sensitivity} = \text{TP}/\text{P}$
    - **Specificity**: True negative recognition rate
      - $\text{Specificity} = \text{TN}/\text{N}$

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

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

$$P = \text{Precision} = \frac{TP}{TP + FP}$$

- ❑ **Recall:** Completeness: what % of positive tuples did the classifier label as positive?

$$R = \text{Recall} = \frac{TP}{TP + FN}$$

- ❑ Range: [0, 1]
- ❑ The “inverse” relationship between precision & recall
- ❑ **F measure (or F-score):** harmonic mean of precision and recall
- ❑ In general, it is the weighted measure of precision & recall

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

Assigning  $\beta$  times as much weight to recall as to precision)

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

- ❑ That is, when  $\beta = 1$ ,

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

# Classifier Evaluation Metrics: Example

---

- ❑ Use the same confusion matrix, calculate the measure just introduced

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	<b>90</b>	<b>210</b>	300	30.00 ( <i>sensitivity</i> )
cancer = no	<b>140</b>	<b>9560</b>	9700	98.56 ( <i>specificity</i> )
Total	230	9770	10000	96.50 ( <i>accuracy</i> )

- ❑ Sensitivity =  $TP/P = 90/300 = 30\%$
- ❑ Specificity =  $TN/N = 9560/9700 = 98.56\%$
- ❑ Accuracy =  $(TP + TN)/All = (90+9560)/10000 = 96.50\%$
- ❑ Error rate =  $(FP + FN)/All = (140 + 210)/10000 = 3.50\%$
- ❑ Precision =  $TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13\%$
- ❑ Recall =  $TP / (TP + FN) = 90/(90 + 210) = 90/300 = 30.00\%$
- ❑  $F1 = 2 P \times R / (P + R) = 2 \times 39.13\% \times 30.00\% / (39.13\% + 30\%) = 33.96\%$

# Classifier Evaluation: Holdout & Cross-Validation

---

- **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
  - Repeated random sub-sampling validation: 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 distribution, in each fold is approximately the same as that in the initial data

# Classifier Evaluation: Bootstrap

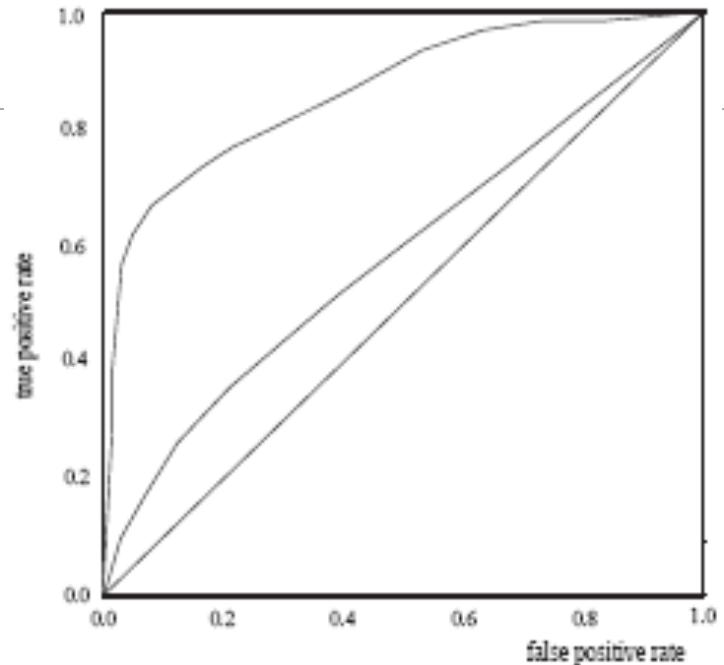
---

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

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

# Model Selection: ROC Curves

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



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

# Issues Affecting Model Selection

---

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

# **Chapter 8. Classification: Basic Concepts**

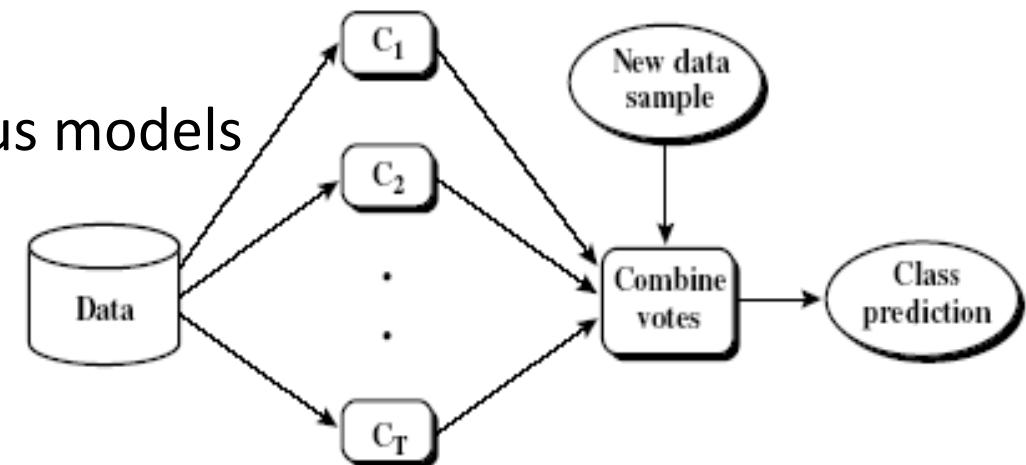
---

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



# Ensemble Methods: Increasing the Accuracy

- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of  $k$  learned models,  $M_1, M_2, \dots, M_k$ , with the aim of creating an improved model  $M^*$
- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set, and models learned in parallel
  - Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order



# 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: It can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy: Improved accuracy in prediction
  - Often significantly better than a single classifier derived from  $D$
  - For noise data: Not considerably worse, more robust

# Random Forest: Basic Concepts

---

- Random Forest (first proposed by L. Breiman in 2001)
  - A variation of bagging for *decision trees*
  - *Data bagging*
    - Use a subset of training data by sampling with replacement for each tree
  - *Feature bagging*
    - At each node use a random selection of attributes as candidates and split by the best attribute among them
  - Compared to original bagging, increases the diversity among generated trees
  - During classification, each tree votes and the most popular class is returned

# Random Forest

---

- 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 typical bagging or boosting

# 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)

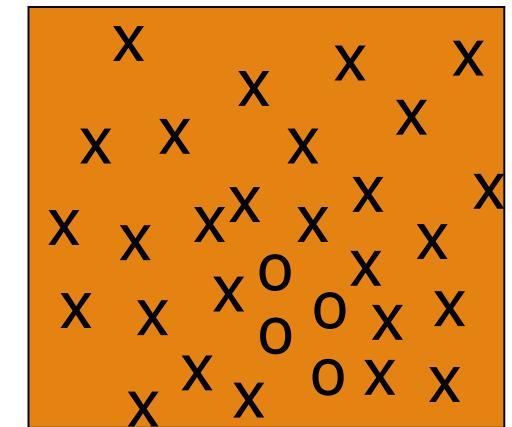
---

- 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; otherwise, it is decreased
- Error rate:  $\text{err}(\mathbf{X}_j)$  is the misclassification error of tuple  $\mathbf{X}_j$ . Classifier  $M_i$  error rate is the sum of the weights of the misclassified tuples:
- The weight of classifier  $M_i$ 's vote is 
$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

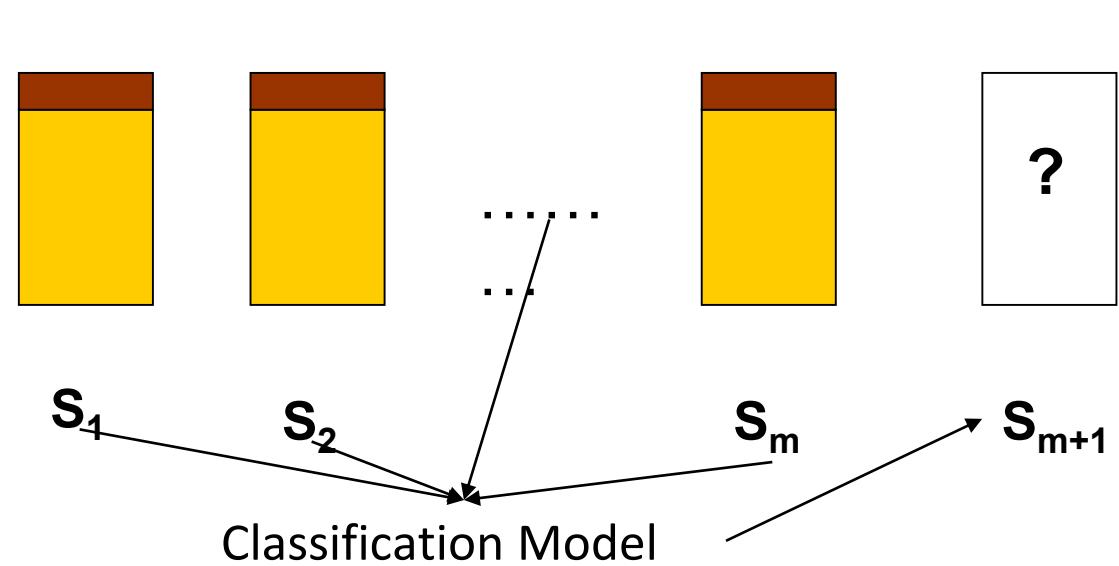
$$\text{error}(M_i) = \sum_j w_j \times \text{err}(\mathbf{X}_j)$$

# Classification of Class-Imbalanced Data Sets

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



# Classifying Data Streams with Skewed Distribution

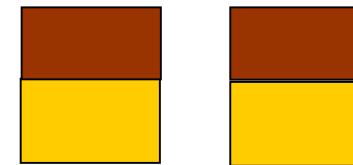


J. Gao, et al., "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions", SDM'07

## Biased Sampling



## Ensemble



$$f^E(x) = \frac{1}{k} \sum_{i=1}^k f^i(x)$$

- Classify data stream with skewed distribution (i.e., rare events)
- **Biased sampling:** Save only the positive examples in the streams
- **Ensemble:** Partition negative examples of  $S_m$  into  $k$  portions to build  $k$  classifiers
- Effectively reduce classification errors on the minority class

# **Chapter 8. Classification: Basic Concepts**

---

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



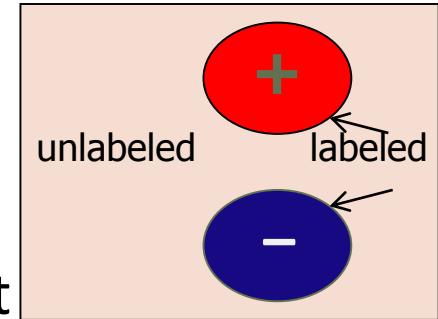
# Multiclass Classification

---

- Classification involving more than two classes (i.e.,  $> 2$  Classes)
- Methodology: Reducing the multi-class problem into multiple binary problems
- Method 1. **One-vs.-rest** (or **one-vs.-all**)
  - Given  $m$  classes, train  $m$  classifiers: one for each class
  - Classifier  $j$ : treat tuples in class  $j$  as *positive* & **all the rest** as *negative*
  - To classify a tuple  $X$ , the set of classifiers vote as an ensemble
- Method 2. **one-vs.-one** (or **all-vs.-all**): Learn a classifier for each pair of classes
  - Given  $m$  classes, construct  $m(m - 1)/2$  binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple  $X$ , each classifier votes
    - $X$  is assigned to the class with maximal vote
- Comparison: One-vs.-one tends to perform better than one-vs.-rest
- Many new algorithms have been developed to go beyond binary classifier method

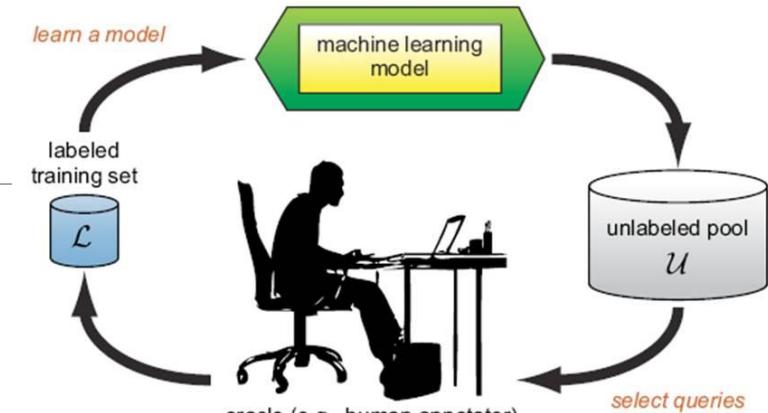
# Semi-Supervised Classification

- ❑ Semi-supervised: Uses labeled and unlabeled data to build a classifier
- ❑ Self-training
  - ❑ Build a classifier using the labeled data
  - ❑ Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - ❑ Repeat the above process
  - ❑ Adv.: easy to understand; Disadv.: may reinforce errors
- ❑ Co-training: Use two or more classifiers to teach each other
  - ❑ Each learner uses a mutually independent set of features of each tuple to train a good classifier, say  $f_1$  and  $f_2$
  - ❑ Then  $f_1$  and  $f_2$  are used to predict the class label for unlabeled data X
  - ❑ Teach each other: The tuple having the most confident prediction from  $f_1$  is added to the set of labeled data for  $f_2$  & vice versa
- ❑ Other methods include joint probability distribution of features and labels



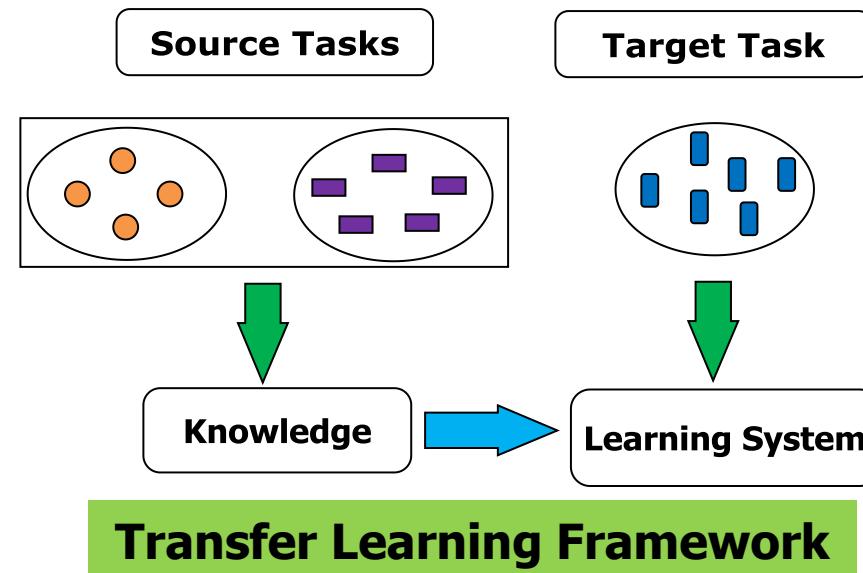
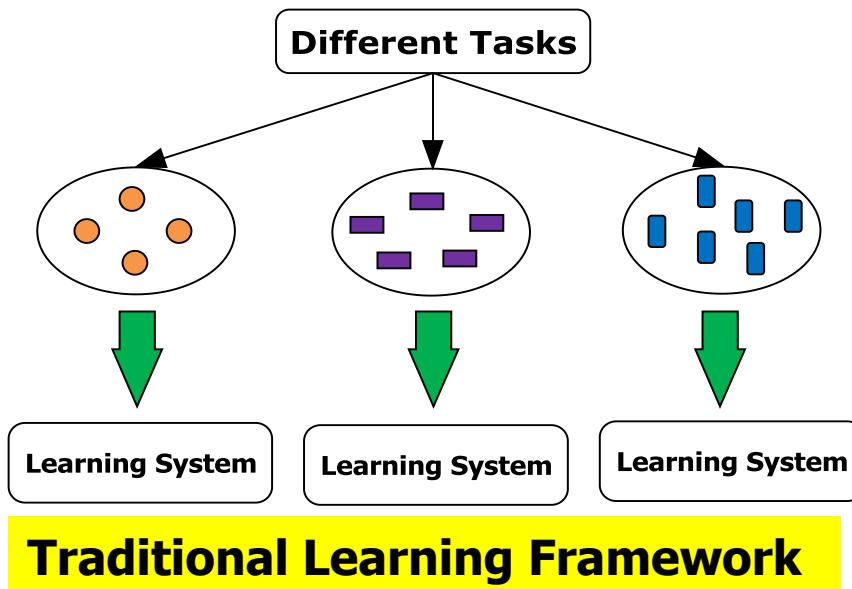
# Active Learning

- A special case of semi-supervised learning
  - Unlabeled data: Abundant
  - Class labels are expensive to obtain
- Active learner: Interactively query teachers (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: **Achieve high accuracy using as few labeled data as possible**
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- A lot of algorithms have been developed for active learning



# Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks (e.g., recognizing cars) and apply the knowledge to a target task (e.g., recognizing trucks)
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks
- Many algorithms are developed, applied to text classification, spam filtering, etc.

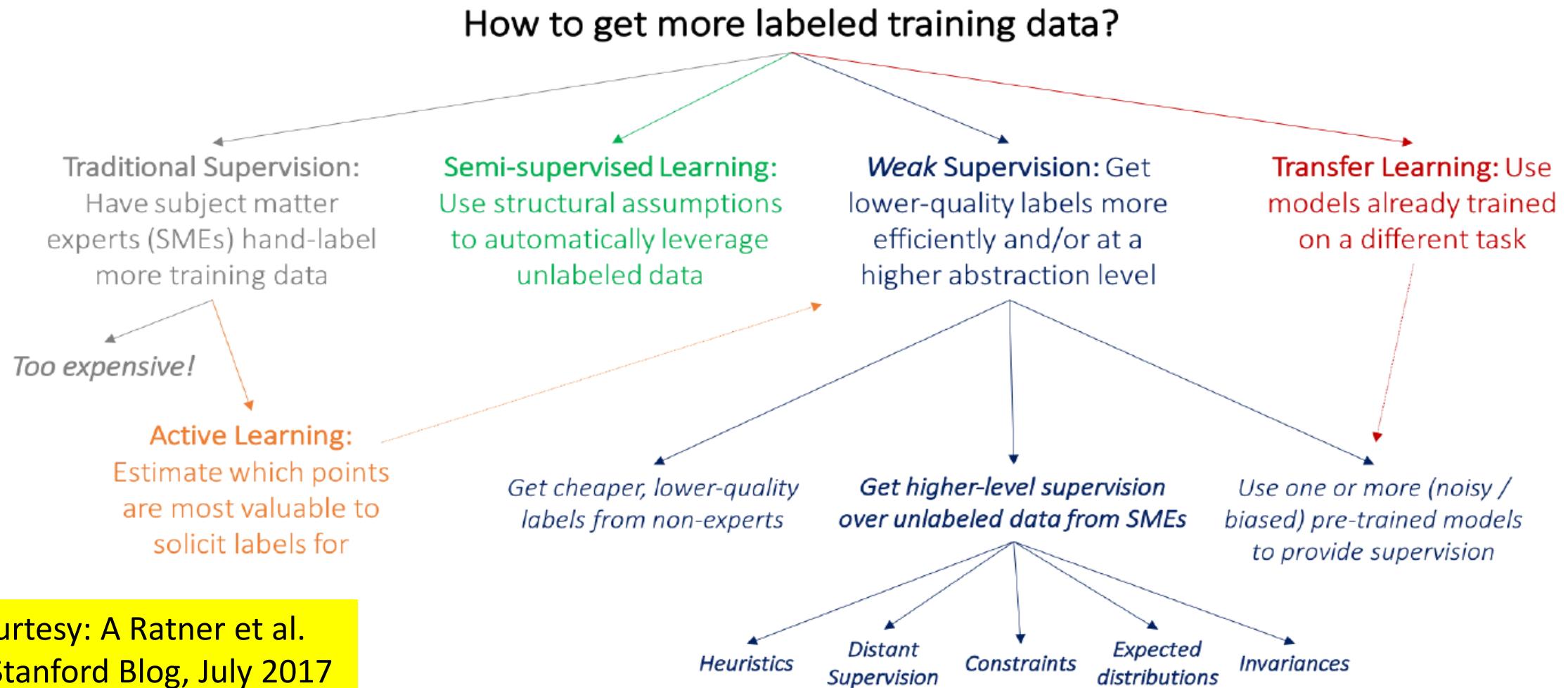


# **Weak Supervision: A New Programming Paradigm for Machine Learning**

---

- ❑ Overcome the training data bottleneck
  - ❑ Leverage higher-level and/or noisier input from experts
- ❑ Exploring weak label distributions provided more cheaply and efficiently by
  - ❑ Higher-level, less precise supervision (e.g., heuristic rules, expected label distributions)
  - ❑ Cheaper, lower-quality supervision (e.g. crowdsourcing)
  - ❑ Existing resources (e.g. knowledge bases, pre-trained models)
- ❑ These weak label distributions could take many forms
  - ❑ Weak Labels from crowd workers, output of heuristic rules, or the result of distant supervision (from KBs), or the output of other classifiers, etc.
  - ❑ Constraints and invariances (e.g., from physics, logic, or other experts)
  - ❑ Probability distributions (e.g., from weak or biased classifiers or user-provided label or feature expectations or measurements)

# Relationships Among Different Kinds of Supervisions



*Many areas of machine learning are motivated by the bottleneck of labeled training data, but are divided at a high-level by what information they leverage instead.*

# Chapter 8. Classification: Basic Concepts

---

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



# Summary

---

- Classification: Model construction from a set of training data
- Effective and scalable methods
  - Decision tree induction, Bayes classification methods, linear classifier, ...
  - No single method has been found to be superior over all others for all data sets
- Evaluation metrics: Accuracy, sensitivity, specificity, precision, recall,  $F$  measure
- Model evaluation: Holdout, cross-validation, bootstrapping, ROC curves (AUC)
- Improve Classification Accuracy: Bagging, boosting
- Additional concepts on classification: Multiclass classification, semi-supervised classification, active learning, transfer learning, weak supervision

# References (1)

---

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules.** Future Generation Computer Systems, 13, 1997
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning.** KDD'95
- A. J. Dobson. **An Introduction to Generalized Linear Models.** Chapman & Hall, 1990.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation.** AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting.** J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets.** VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction.** SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.

# References (2)

---

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

# References (3)

---

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning.** VLDB'98
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining.** VLDB'96
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning.** Morgan Kaufmann, 1990
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining.** Addison Wesley, 2005
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.** Morgan Kaufman, 1991
- S. M. Weiss and N. Indurkhy. **Predictive Data Mining.** Morgan Kaufmann, 1997
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques,** 2ed. Morgan Kaufmann, 2005



# Bayes' Theorem: Basics

---

- Total probability Theorem:

$$P(B) = \sum_{i=1}^M P(B|A_i)P(A_i)$$

- Bayes' Theorem:

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

- Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
- Let  $H$  be a *hypothesis* that  $\mathbf{X}$  belongs to class C
- Classification is to determine  $P(H|\mathbf{X})$ , (i.e., *posteriori probability*): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$
- $P(H)$  (*prior probability*): the initial probability
  - E.g.,  $\mathbf{X}$  will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$ : probability that sample data is observed
- $P(\mathbf{X}|H)$  (likelihood): the probability of observing the sample  $\mathbf{X}$ , given that the hypothesis holds
  - E.g., Given that  $\mathbf{X}$  will buy computer, the prob. that  $\mathbf{X}$  is 31..40, medium income

# Classification Is to Derive the Maximum Posteriori

---

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

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

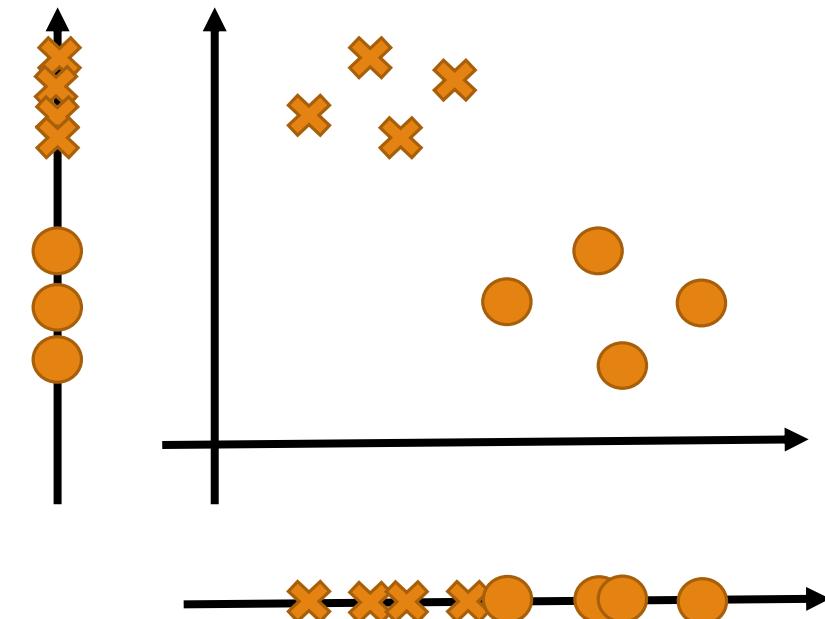
- Since  $P(X)$  is constant for all classes, only

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

needs to be maximized

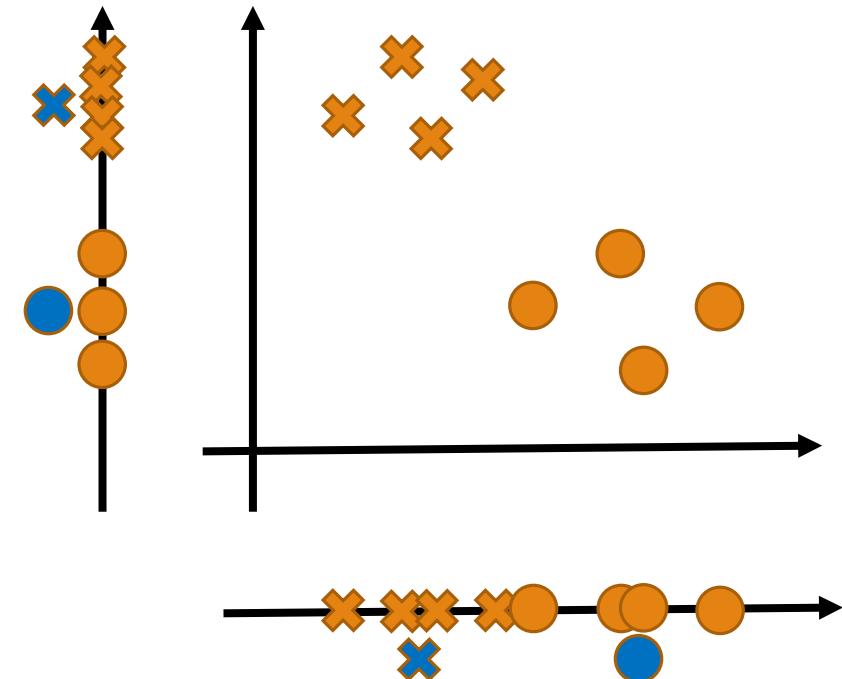
# Linear Discriminant Analysis (LDA)

- Linear Discriminant Analysis (LDA) works when the attributes are all continuous
  - For the categorical attributes, discriminant correspondence analysis is the equivalent technique
- Basic Ideas: Project all samples on a line such that different classes are well separated
- Example: Suppose we have 2 classes and 2-dimensional samples  $x_1, \dots, x_n$ 
  - $n_1$  samples come from class 1
  - $n_2$  samples come from class 2
- Let the line direction be given by unit vector  $\nu$
- There are two candidates of projections
  - Vertical:  $\nu = (0,1)$
  - Horizontal:  $\nu = (1,0)$
- Which one looks better?
- How to mathematically measure it?



# Fisher's LDA (Linear Discriminant Analysis)

- $v^T x_i$  is the distance of projection of  $x_i$  from the origin
- Let  $\mu_1$  and  $\mu_2$  be the means of class 1 and class 2 in the original space
  - $\mu_1 = \frac{1}{n_1} \sum_{i \in \text{class 1}} x_i$
  - $\mu_2 = \frac{1}{n_2} \sum_{i \in \text{class 2}} x_i$
- The distance between the means of the projected points
  - $|v^T \mu_1 - v^T \mu_2|$
  - Good? No. Horizontal one may have larger distance



# Fisher's LDA (con't)

- ❑ Normalization needed
- ❑ Scatter: Sample variance multiplied by  $n$

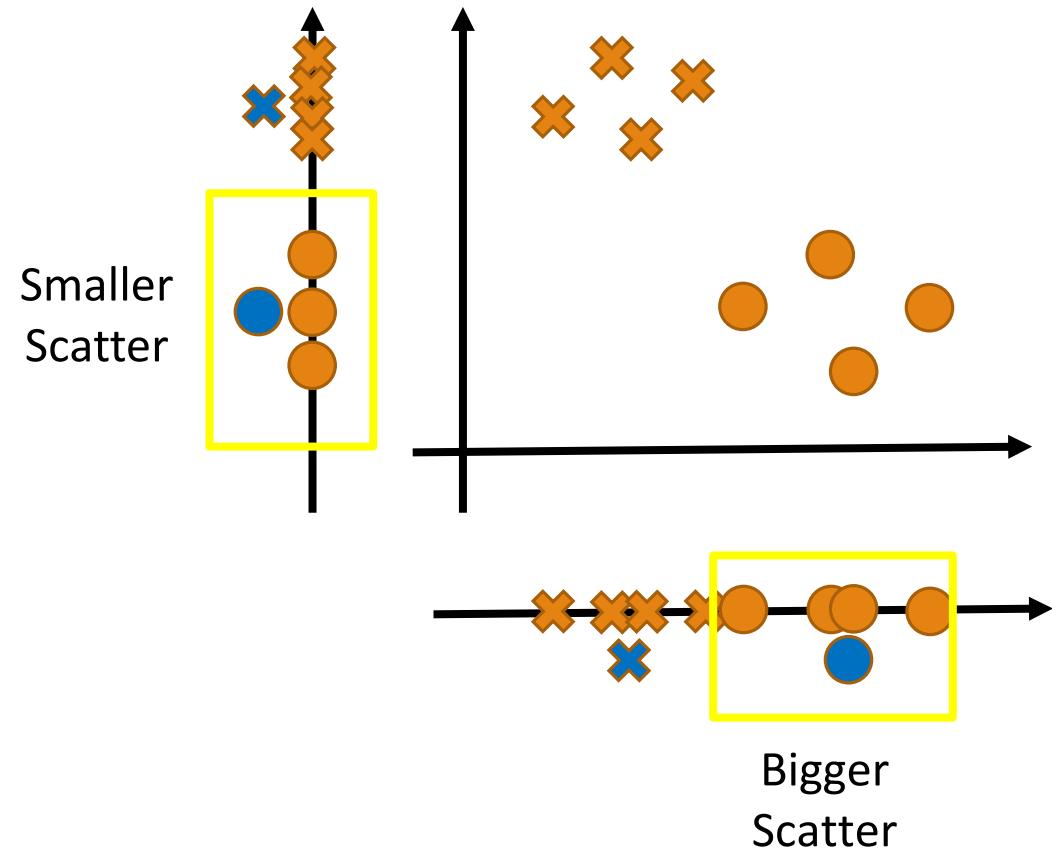
- ❑  $s_1 = \sum_{i \in \text{class 1}} (\mathbf{v}^T \mathbf{x}_i - \mathbf{v}^T \boldsymbol{\mu}_1)^2$

- ❑  $s_2 = \sum_{i \in \text{class 2}} (\mathbf{v}^T \mathbf{x}_i - \mathbf{v}^T \boldsymbol{\mu}_2)^2$

- ❑ Fisher's LDA

- ❑ Maximize  $J(\mathbf{v}) = \frac{(\mathbf{v}^T \boldsymbol{\mu}_1 - \mathbf{v}^T \boldsymbol{\mu}_2)^2}{s_1 + s_2}$

- ❑ Closed-form optimal solution



# Fisher's LDA: Summary

---

- Advantages
  - Useful for dimension reduction
  - Easy to extend to multi-classes
  
- Fisher's LDA will fail
  - When  $\mu_1 = \mu_2, J(\nu)$  is always 0.
  - When classes have large overlap when projected to any line