

# SDSC6004 Data Analytics and Data Mining II

Zijun Zhang

Parts of contents from textbook, Introduction to Data Mining

# Classification: Definition

¶ Given a collection of records (training set )

- Each record is characterized by a tuple  $(x,y)$ , where  $x$  is the attribute set and  $y$  is the class label
  - ◆  $x$ : attribute, predictor, independent variable, input
  - ◆  $y$ : class, response, dependent variable, output

¶ Task:

- Learn a model that maps each attribute set  $x$  into one of the predefined class labels  $y$

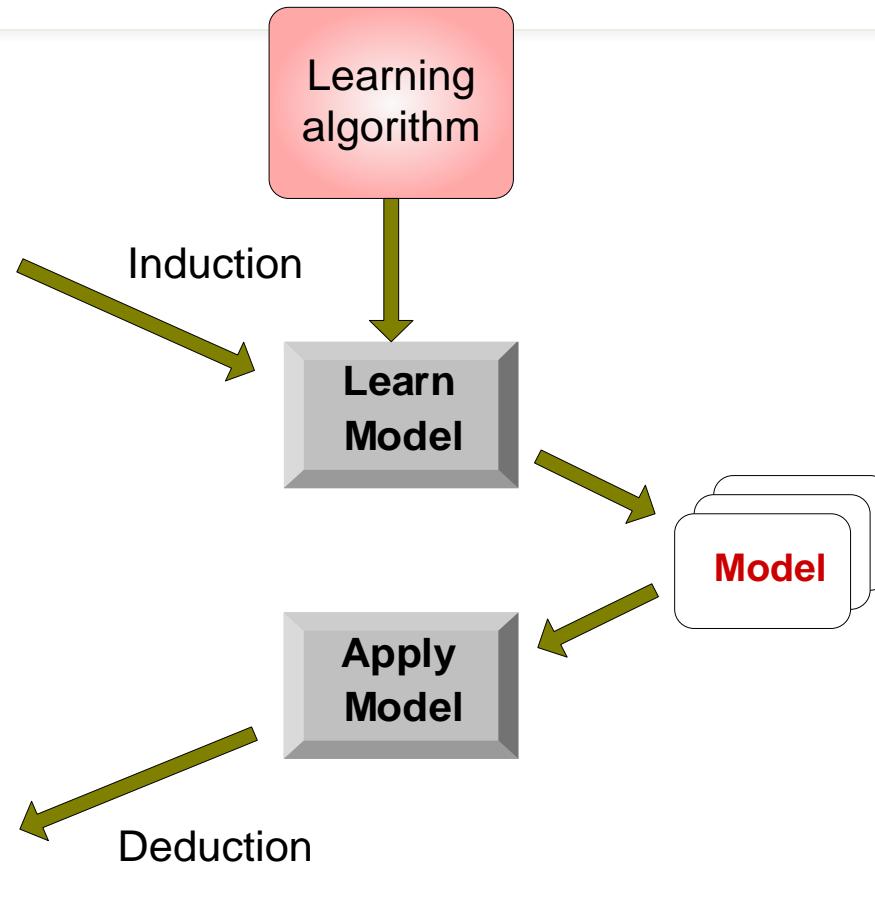
# General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



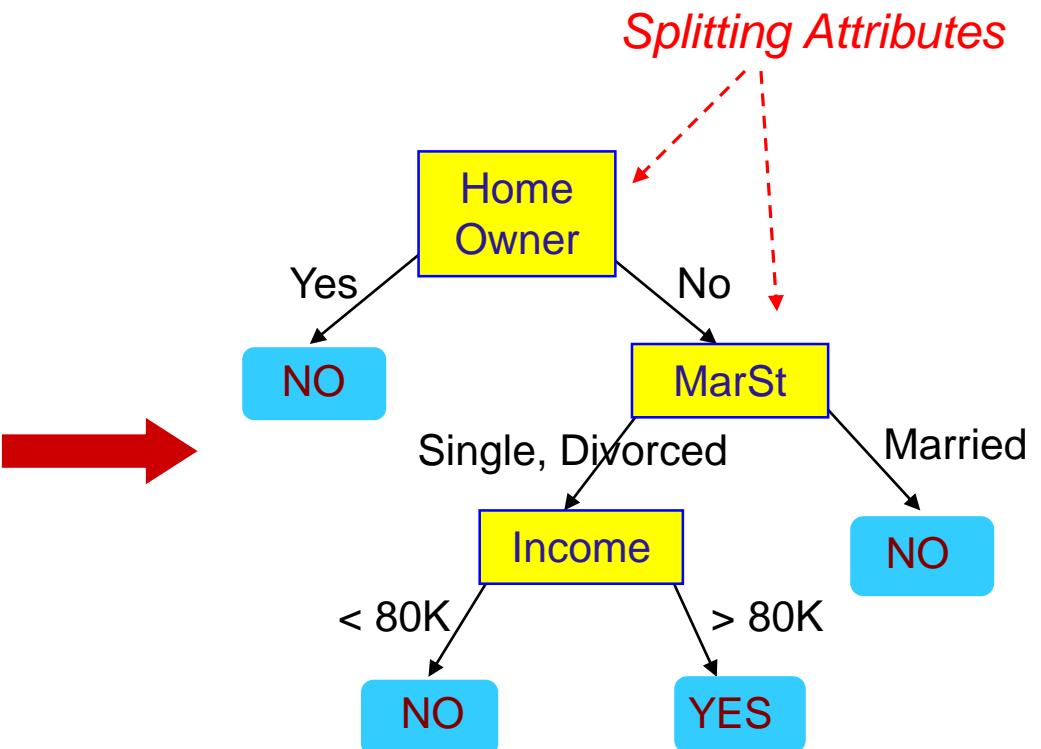
# Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Nearest-neighbor
  - Neural Networks
  - Naïve Bayes
  - Support Vector Machines
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

# Examples of a Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower	class
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	

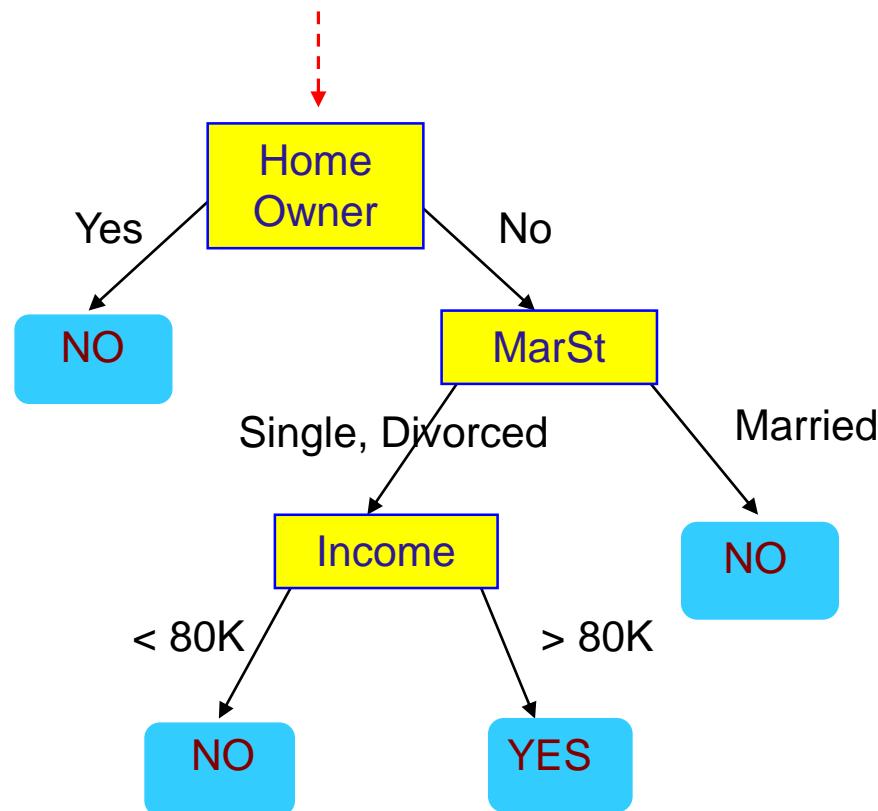
Training Data



Model: Decision Tree

# Apply Model to Test Data

Start from the root of tree.



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

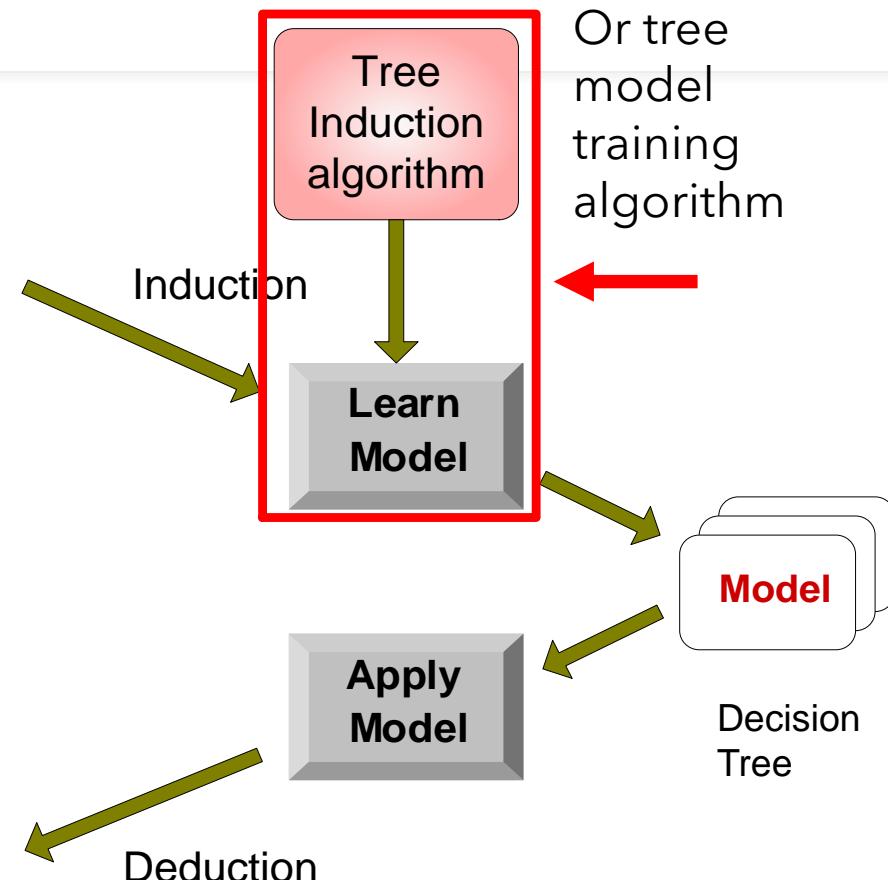
# Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# Decision Tree Training Algorithms

Many algorithms, famous ones:

CART

ID3

C4.5

# Design Issues of Decision Tree Induction

❑ How should training records be split?

- Method for specifying test condition
  - ◆ depending on attribute types
- Measure for evaluating the goodness of a test condition

❑ How should the splitting procedure stop?

- Stop splitting if all the records belong to the same class or have identical attribute values
- Early termination

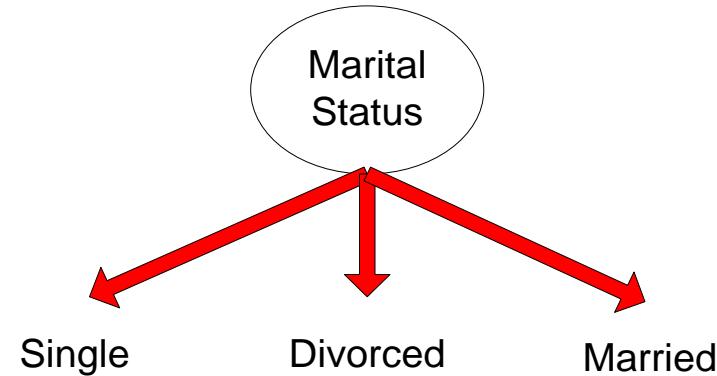
# Methods for Expressing Test Conditions

- ❑ Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous
  
- ❑ Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Test Condition for Nominal Attributes

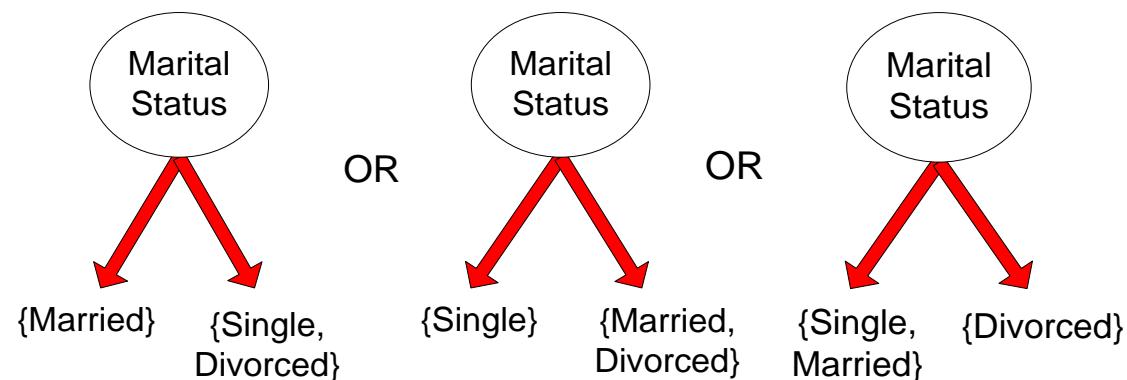
## □ Multi-way split:

- Use as many partitions as distinct values.



## □ Binary split:

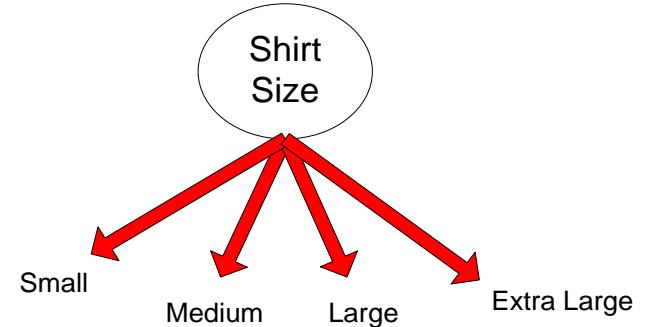
- Divides values into two subsets



# Test Condition for Ordinal Attributes

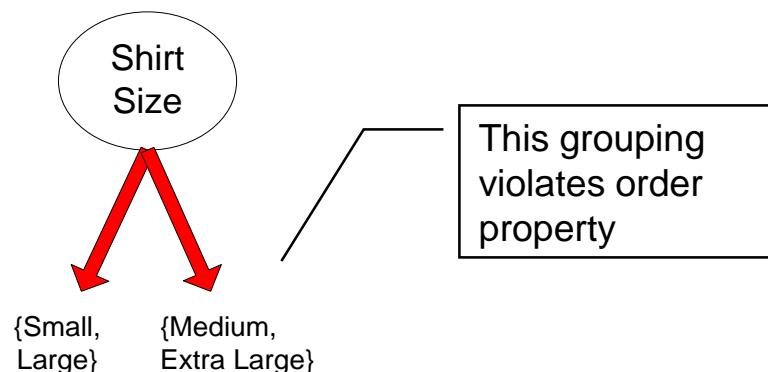
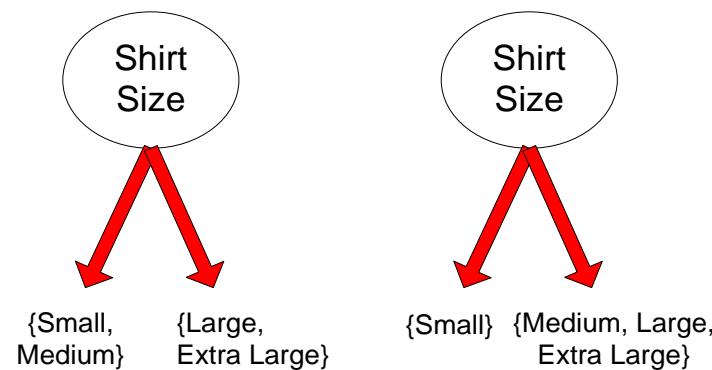
## | Multi-way split:

- Use as many partitions as distinct values



## | Binary split:

- Divides values into two subsets
- Preserve order property among attribute values



# Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute

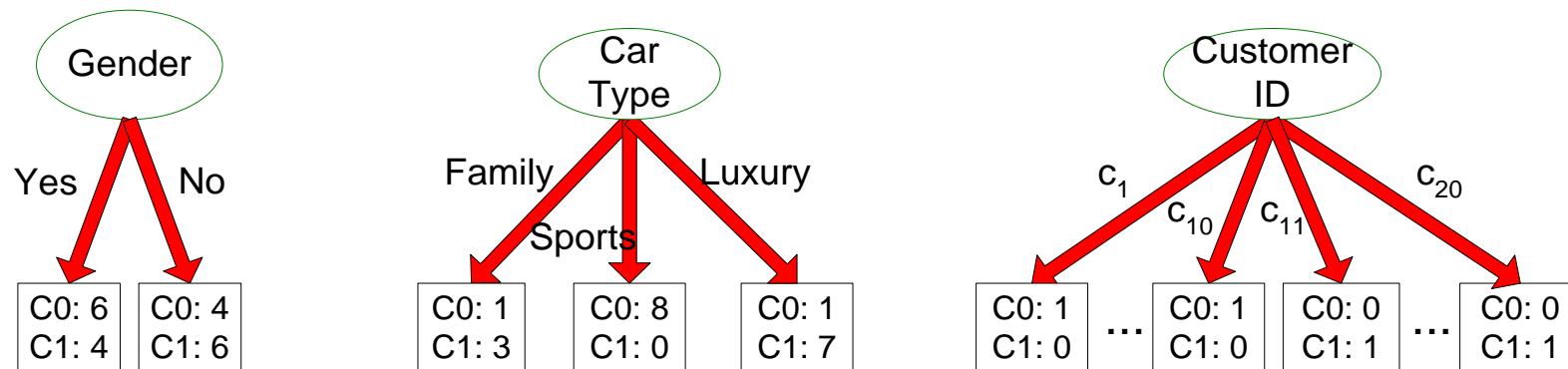
Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

    - Static – discretize once at the beginning
    - Dynamic – repeat at each node
  - **Binary Decision:**  $(A < v)$  or  $(A \geq v)$ 
    - consider all possible splits and finds the best cut
    - can be more compute intensive

# How to determine the Best Split

Before Splitting: 10 records of class 0,  
10 records of class 1

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



Which test condition is the best?

# How to determine the Best Split

- Greedy approach:
  - Nodes with **purer** class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

# Measures of Node Impurity

□ Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where  $p_i(t)$  is the frequency of class  $i$  at node  $t$ , and  $c$  is the total number of classes

□ Entropy

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

□ Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

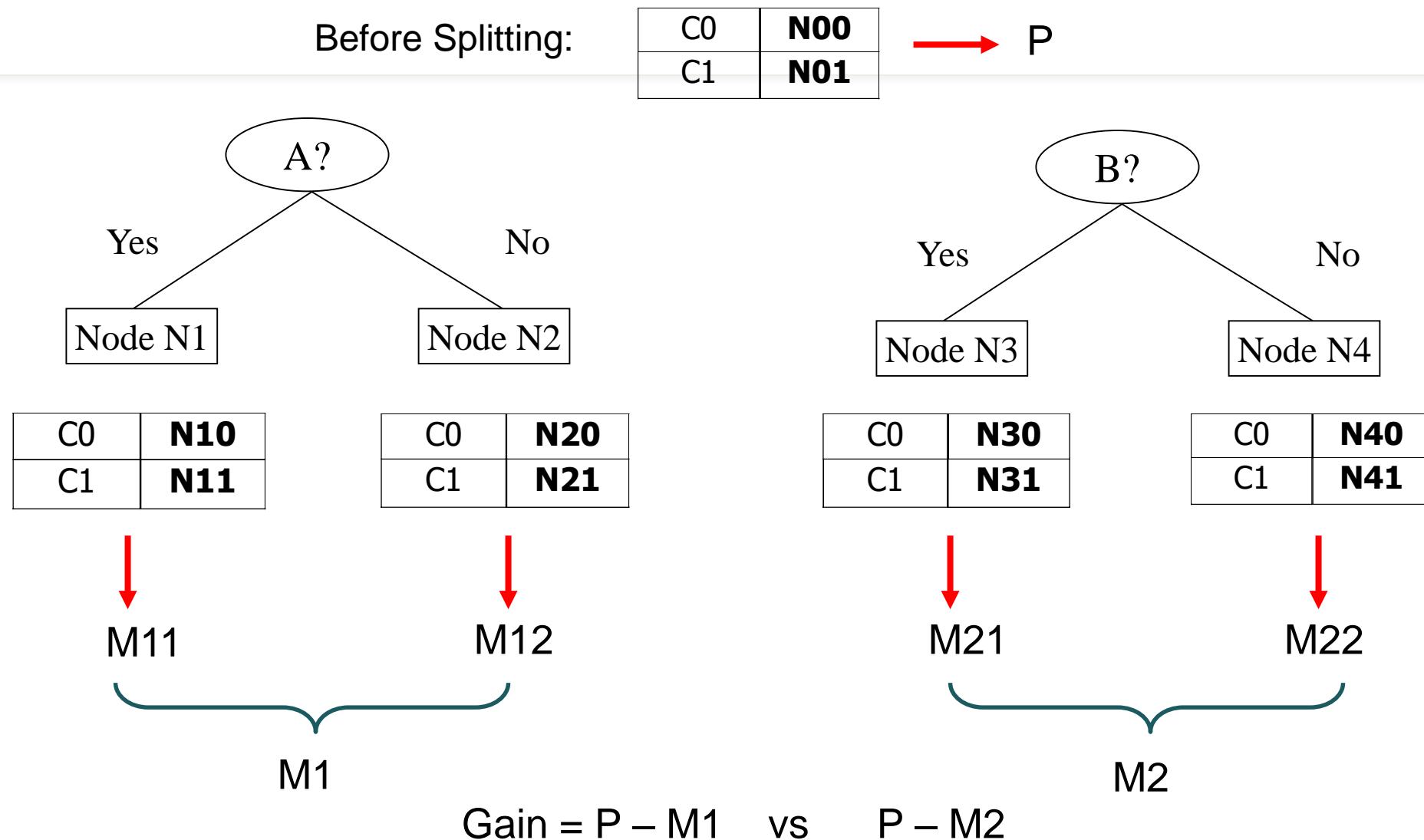
# Finding the Best Split

1. Compute impurity measure ( $P$ ) before splitting
2. Compute impurity measure ( $M$ ) after splitting
  - Compute impurity measure of each child node
  - $M$  is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting ( $M$ )

# Finding the Best Split



# Measure of Impurity: GINI

- Gini Index for a given node  $t$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where  $p_i(t)$  is the frequency of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- Maximum of  $1 - 1/c$  when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

# Measure of Impurity: GINI

- Gini Index for a given node t :

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

- For 2-class problem ( $p, 1 - p$ ):
  - $GINI = 1 - p^2 - (1 - p)^2 = 2p(1-p)$

C1	<b>0</b>
C2	<b>6</b>
<b>Gini=0.000</b>	

C1	<b>1</b>
C2	<b>5</b>
<b>Gini=0.278</b>	

C1	<b>2</b>
C2	<b>4</b>
<b>Gini=0.444</b>	

C1	<b>3</b>
C2	<b>3</b>
<b>Gini=0.500</b>	

# Computing Gini Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

# Computing Gini Index for a Collection of Nodes

- | When a node  $p$  is split into  $k$  partitions (children)

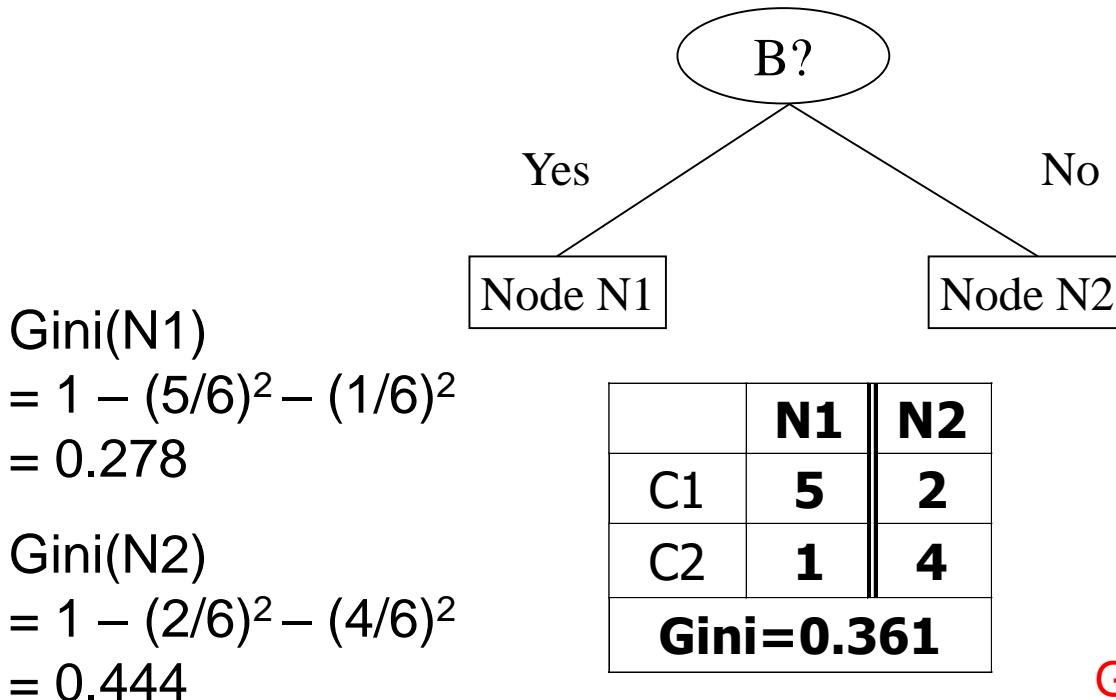
$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where,       $n_i$  = number of records at child  $i$ ,  
                 $n$  = number of records at parent node  $p$ .

- | Choose the attribute that minimizes weighted average Gini index of the children
- | Gini index is used in decision tree algorithms such as CART

# Binary Attributes: Computing GINI Index

- Splits into two partitions (child nodes)
- Effect of Weighing partitions:
  - Larger and purer partitions are sought



	<b>Parent</b>
C1	<b>7</b>
C2	<b>5</b>
<b>Gini = 0.486</b>	

Weighted Gini of N1 N2  
 $= 6/12 * 0.278 +$   
 $6/12 * 0.444$   
 $= 0.361$

**Gain = 0.486 – 0.361 = 0.125**

# 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	8	1
C2	3	0	7
Gini	<b>0.163</b>		

Two-way split

(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	<b>0.468</b>	

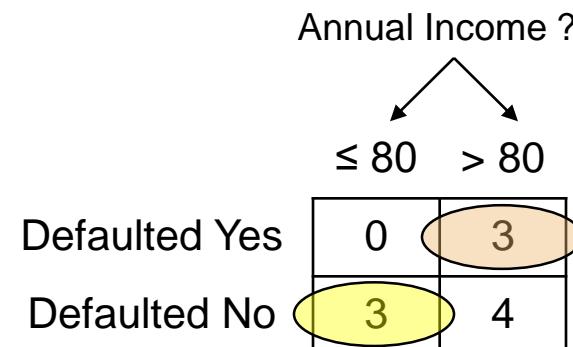
	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	<b>0.167</b>	

Which of these is the best?

# Continuous Attributes: Computing Gini Index

- | Use Binary Decisions based on one value
- | Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- | Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A \leq v$  and  $A > v$
- | 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.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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...

- | For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No		
Annual Income												
→	60	70	75	85	90	95	100	120	125	220		
→	55	65	72	80	87	92	97	110	122	172	230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	1	2	2	1	3	0
No	0	7	1	6	2	5	3	4	3	4	3	0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420	

Sorted Values  
Split Positions

# Measure of Impurity: Entropy

## I Entropy at a given node $t$

$$\text{Entropy} = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

Where  $p_i(t)$  is the frequency of class  $i$  at node  $t$ , and  $c$  is the total number of classes

- ◆ Maximum of  $\log_2 c$  when records are equally distributed among all classes, implying the least beneficial situation for classification
  - ◆ Minimum of 0 when all records belong to one class, implying most beneficial situation for classification
- 
- Entropy based computations are quite similar to the GINI index computations

# Computing Entropy of a Single Node

$$Entropy = - \sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t)$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = - 0 \log 0 - 1 \log 1 = - 0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = - (1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$$

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Entropy} = - (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

# Computing Information Gain After Splitting

## I Information Gain:

$$Gain_{split} = Entropy(p) - \sum_{i=1}^k \frac{n_i}{n} Entropy(i)$$

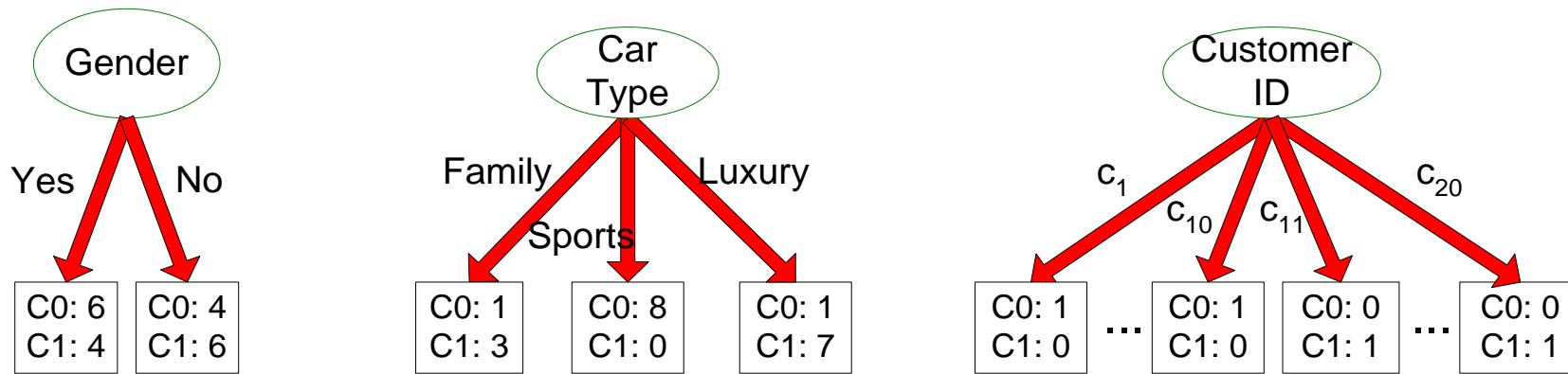
Parent Node,  $p$  is split into  $k$  partitions (children)

$n_i$  is number of records in child node  $i$

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms
- Information gain is the mutual information between the class variable and the splitting variable

# Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



- Customer ID has highest information gain because entropy for all the children is zero

# Gain Ratio

| Gain Ratio:

$$\text{Gain Ratio} = \frac{\text{Gain}_{\text{split}}}{\text{Split Info}} \quad \text{Split Info} = - \sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node,  $p$  is split into  $k$  partitions (children)

$n_i$  is number of records in child node  $i$

- Adjusts Information Gain by the entropy of the partitioning (*Split Info*).
  - ◆ Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain

# Gain Ratio

I Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info}$$

$$Split\ Info = \sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node,  $p$  is split into  $k$  partitions (children)

$n_i$  is number of records in child node  $i$

CarType			
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

$$\text{SplitINFO} = 1.52$$

CarType		
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

$$\text{SplitINFO} = 0.72$$

CarType		
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

$$\text{SplitINFO} = 0.97$$

## Measure of Impurity: Classification Error

### I Classification error at a node $t$

$$Error(t) = 1 - \max_i[p_i(t)]$$

- Maximum of  $1 - 1/c$  when records are equally distributed among all classes, implying the least interesting situation
- Minimum of 0 when all records belong to one class, implying the most interesting situation

# Computing Error of a Single Node

$$Error(t) = 1 - \max_i[p_i(t)]$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

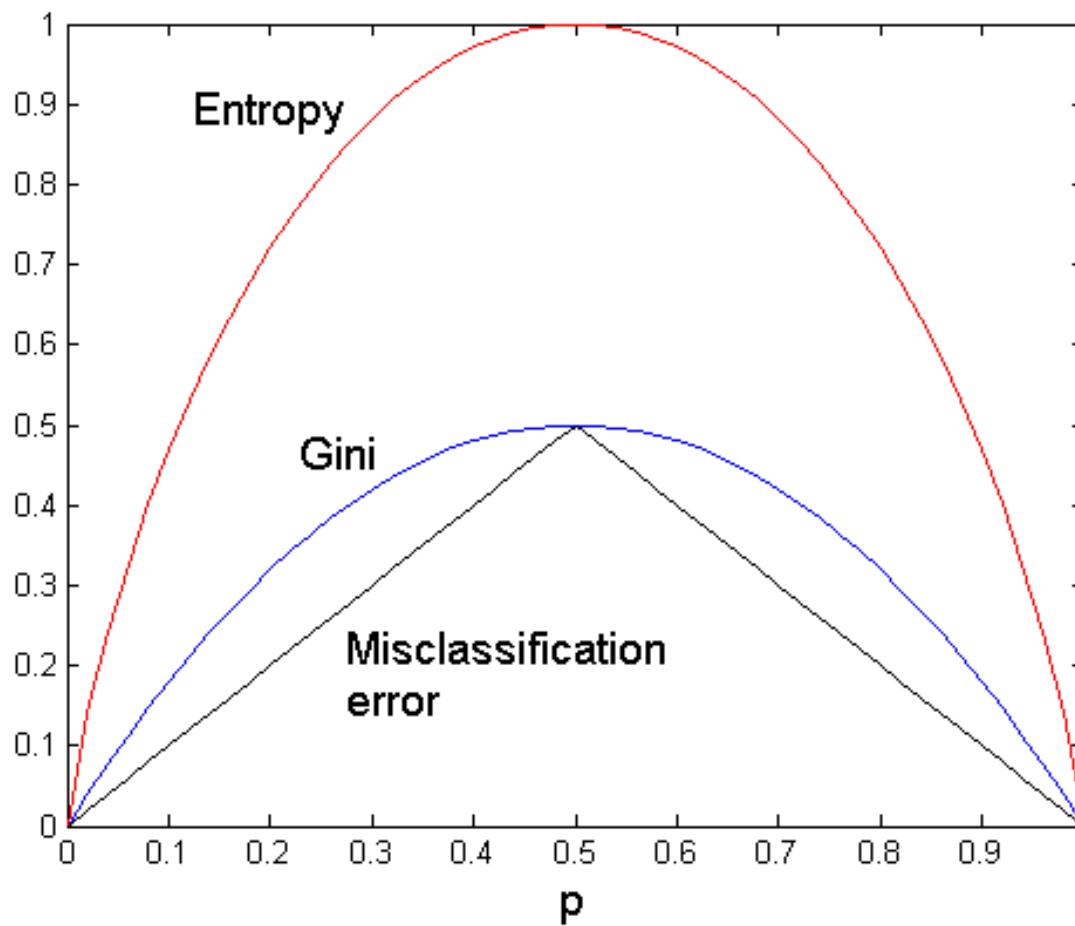
C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

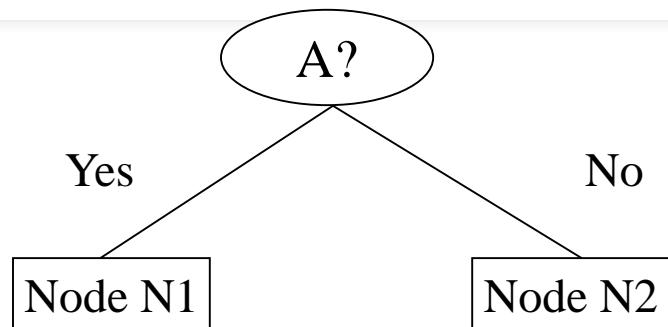
$$\text{Error} = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

# Comparison among Impurity Measures

For a 2-class problem:



# Misclassification Error vs Gini Index



	<b>Parent</b>
C1	<b>7</b>
C2	<b>3</b>
<b>Gini = 0.42</b>	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (3/3)^2 - (0/3)^2 \\ &= 0 \end{aligned}$$

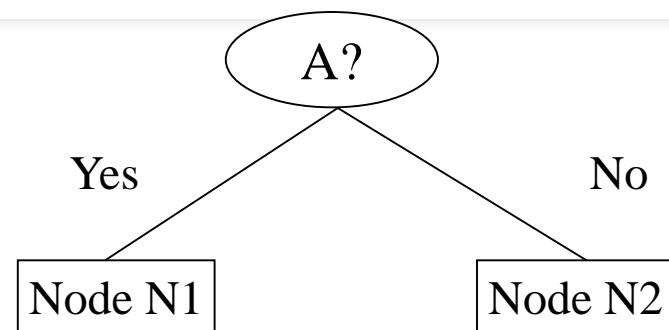
$$\begin{aligned} \text{Gini}(N2) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489 \end{aligned}$$

	<b>N1</b>	<b>N2</b>
C1	<b>3</b>	<b>4</b>
C2	<b>0</b>	<b>3</b>
<b>Gini=0.342</b>		

$$\begin{aligned} \text{Gini(Children)} &= 3/10 * 0 \\ &+ 7/10 * 0.489 \\ &= 0.342 \end{aligned}$$

Gini improves but  
error remains the  
same!!

# Misclassification Error vs Gini Index



	<b>Parent</b>
C1	<b>7</b>
C2	<b>3</b>
<b>Gini = 0.42</b>	

	<b>N1</b>	<b>N2</b>
C1	<b>3</b>	<b>4</b>
C2	<b>0</b>	<b>3</b>
<b>Gini=0.342</b>		

	<b>N1</b>	<b>N2</b>
C1	<b>3</b>	<b>4</b>
C2	<b>1</b>	<b>2</b>
<b>Gini=0.416</b>		

Misclassification error for all three cases = 0.3 !

# Decision Tree Based Classification

## I Advantages:

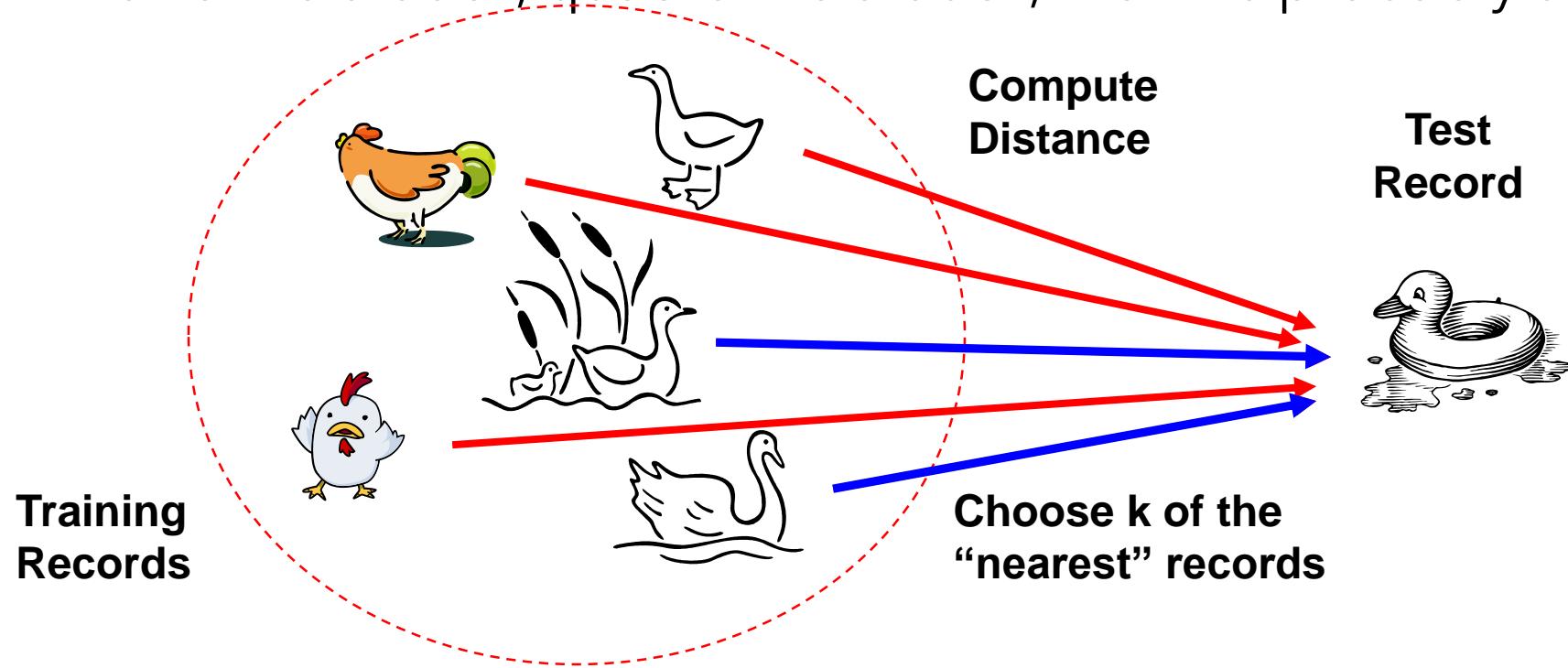
- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Robust to noise (especially when methods to avoid overfitting are employed)
- Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)

## I Disadvantages:

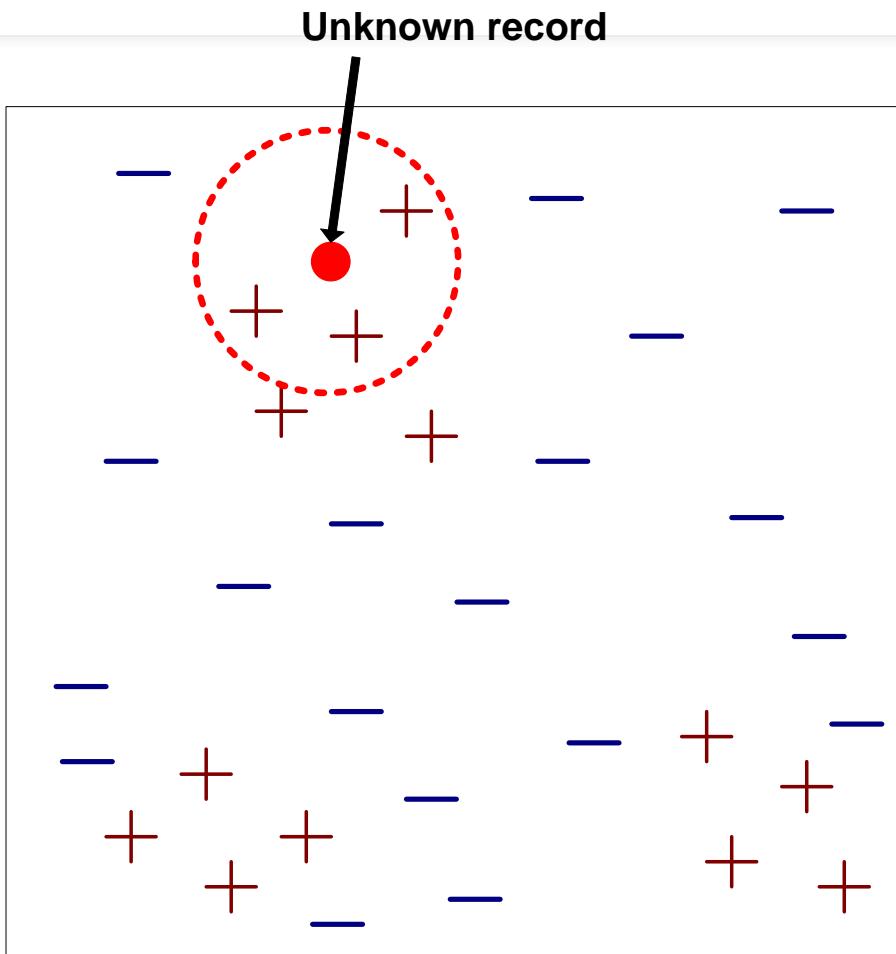
- Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
- Does not take into account interactions between attributes
- Each decision boundary involves only a single attribute

# Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



# Nearest-Neighbor Classifiers



- Requires three things
  - The set of labeled records
  - Distance metric to compute distance between records
  - The value of  $k$ , the number of nearest neighbors to retrieve
  
- To classify an unknown record:
  - Compute distance to other training records
  - Identify  $k$  nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

# Nearest Neighbor Classification

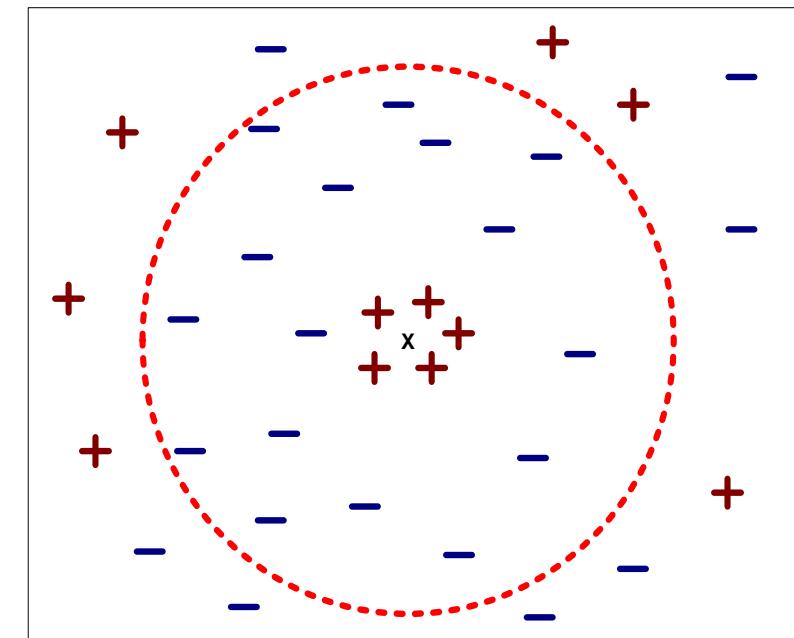
- Compute proximity between two points:
  - Example: Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the k-nearest neighbors
  - Weight the vote according to distance
    - weight factor,  $w = 1/d^2$

# Nearest Neighbor Classification...

- Choosing the value of  $k$ :
  - If  $k$  is too small, sensitive to noise points
  - If  $k$  is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification...

- **Choice of proximity measure matters**
  - For documents, cosine is better than correlation or Euclidean

1 1 1 1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0 0 0 1

vs

0 1 1 1 1 1 1 1 1 1 1

1 0 0 0 0 0 0 0 0 0 0

Euclidean distance = 1.4142 for both pairs

# Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:  
$$P(Y | X) = \frac{P(X, Y)}{P(X)}$$
  
$$P(X | Y) = \frac{P(X, Y)}{P(Y)}$$
- Bayes theorem:  
$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$$

# Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes ( $X_1, X_2, \dots, X_d$ )
  - Goal is to predict class  $Y$
  - Specifically, we want to find the value of  $Y$  that maximizes  $P(Y|X_1, X_2, \dots, X_d)$
- Can we estimate  $P(Y|X_1, X_2, \dots, X_d)$  directly from data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

# Example Data

**Given a Test Record:**

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Can we estimate  
 $P(\text{Evade} = \text{Yes} | X)$  and  $P(\text{Evade} = \text{No} | X)$ ?

In the following we will replace  
Evade = Yes by Yes, and  
Evade = No by No

# Using Bayes Theorem for Classification

- Approach:
  - compute posterior probability  $P(Y | X_1, X_2, \dots, X_d)$  using the Bayes theorem

$$P(Y | X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d | Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

- *Maximum a-posteriori:* Choose  $Y$  that maximizes  $P(Y | X_1, X_2, \dots, X_d)$
- Equivalent to choosing value of  $Y$  that maximizes  $P(X_1, X_2, \dots, X_d | Y) P(Y)$
- How to estimate  $P(X_1, X_2, \dots, X_d | Y)$ ?

# Example Data

Given a Test Record:  $X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

Tid	Refund	Marital Status	Taxable Income	Evade
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

## Using Bayes Theorem:

- $P(\text{Yes} | X) = \frac{P(X | \text{Yes})P(\text{Yes})}{P(X)}$
- $P(\text{No} | X) = \frac{P(X | \text{No})P(\text{No})}{P(X)}$
- How to estimate  $P(X | \text{Yes})$  and  $P(X | \text{No})$ ?

# Naïve Bayes Classifier

- Assume independence among attributes  $X_i$  when class is given:
  - $P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$
  - Now we can estimate  $P(X_i | Y_j)$  for all  $X_i$  and  $Y_j$  combinations from the training data
  - New point is classified to  $Y_j$  if  $P(Y_j) \prod P(X_i | Y_j)$  is maximal.

# Naïve Bayes on Example Data

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Tid	Refund	Marital Status	Taxable Income	Evaide
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

$$P(X | \text{Yes}) =$$

$$\begin{aligned} & P(\text{Refund} = \text{No} | \text{Yes}) \times \\ & P(\text{Divorced} | \text{Yes}) \times \\ & P(\text{Income} = 120\text{K} | \text{Yes}) \end{aligned}$$

$$P(X | \text{No}) =$$

$$\begin{aligned} & P(\text{Refund} = \text{No} | \text{No}) \times \\ & P(\text{Divorced} | \text{No}) \times \\ & P(\text{Income} = 120\text{K} | \text{No}) \end{aligned}$$

# Estimate Probabilities from Data

- $P(y)$  = fraction of instances of class  $y$ 
  - e.g.,  $P(\text{No}) = 7/10$ ,  
 $P(\text{Yes}) = 3/10$

Tid	Refund	Marital Status	Taxable Income	Evade
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

- For categorical attributes:
$$P(X_i = c | y) = n_c / n$$
  - where  $|X_i = c|$  is number of instances having attribute value  $X_i = c$  and belonging to class  $y$
  - Examples:

$$P(\text{Status}=\text{Married}|\text{No}) = 4/7$$
$$P(\text{Refund}=\text{Yes}|\text{Yes})=0$$

# Estimate Probabilities from Data

- For continuous attributes:
  - **Discretization:** Partition the range into bins:
    - ◆ Replace continuous value with bin value
      - Attribute changed from continuous to ordinal
  - **Probability density estimation:**
    - ◆ Assume attribute follows a normal distribution
    - ◆ Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - ◆ Once probability distribution is known, use it to estimate the conditional probability  $P(X_i|Y)$

# Estimate Probabilities from Data

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Normal distribution:

$$P(X_i | Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

– One for each  $(X_i, Y_j)$  pair

- For (Income, Class=No):

– If Class=No

- ◆ sample mean = 110
- ◆ sample variance = 2975

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

# Example of Naïve Bayes Classifier

**Given a Test Record:**

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

- $P(X | \text{No}) = P(\text{Refund}=\text{No} | \text{No})$   
 $\times P(\text{Divorced} | \text{No})$   
 $\times P(\text{Income}=120\text{K} | \text{No})$   
 $= 4/7 \times 1/7 \times 0.0072 = 0.0006$

- $P(X | \text{Yes}) = P(\text{Refund}=\text{No} | \text{Yes})$   
 $\times P(\text{Divorced} | \text{Yes})$   
 $\times P(\text{Income}=120\text{K} | \text{Yes})$   
 $= 1 \times 1/3 \times 1.2 \times 10^{-9} = 4 \times 10^{-10}$

For Taxable Income:

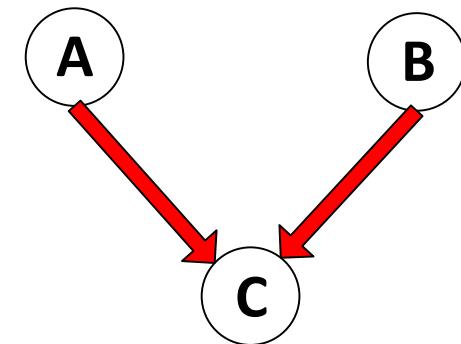
If class = No: sample mean = 110  
sample variance = 2975

If class = Yes: sample mean = 90  
sample variance = 25

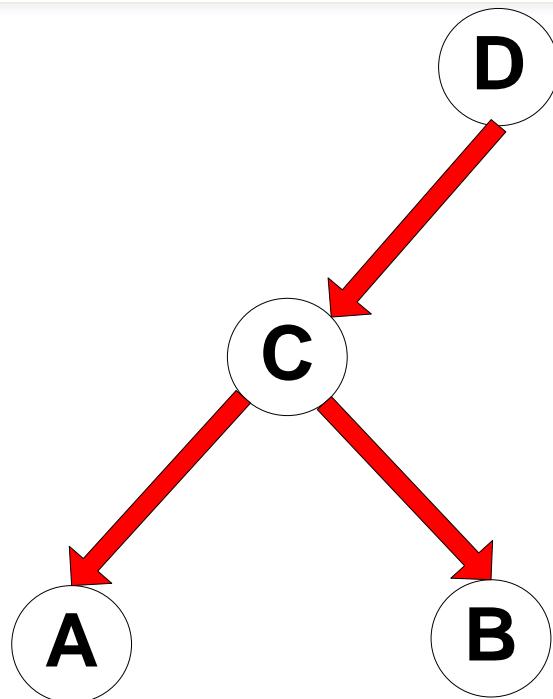
Since  $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$   
Therefore  $P(\text{No}|X) > P(\text{Yes}|X)$   
 $\Rightarrow \text{Class} = \text{No}$

# Bayesian Belief Networks

- Provides graphical representation of probabilistic relationships among a set of random variables
- Consists of:
  - A directed acyclic graph (dag)
    - ◆ Node corresponds to a variable
    - ◆ Arc corresponds to dependence relationship between a pair of variables
  - A probability table associating each node to its immediate parent



# Conditional Independence



**D is parent of C**

**A is child of C**

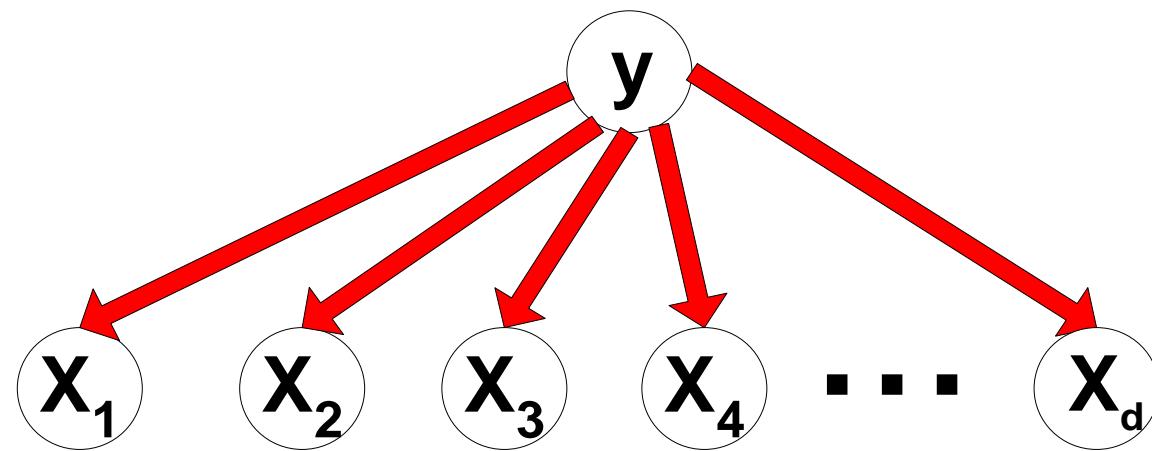
**B is descendant of D**

**D is ancestor of A**

- A node in a Bayesian network is conditionally independent of all of its nondescendants, if its parents are known

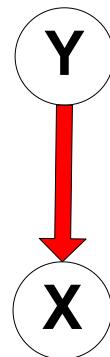
# Conditional Independence

- Naïve Bayes assumption:



# Probability Tables

- If  $X$  does not have any parents, table contains prior probability  $P(X)$
- If  $X$  has only one parent ( $Y$ ), table contains conditional probability  $P(X|Y)$
- If  $X$  has multiple parents ( $Y_1, Y_2, \dots, Y_k$ ), table contains conditional probability  $P(X|Y_1, Y_2, \dots, Y_k)$



# Example of Bayesian Belief Network

Exercise=Yes	0.7
Exercise>No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	HD=Yes	HD>No
CP=Yes	0.8	0.01
CP>No	0.2	0.99

	HD=Yes	HD>No
BP=High	0.85	0.2
BP=Low	0.15	0.8

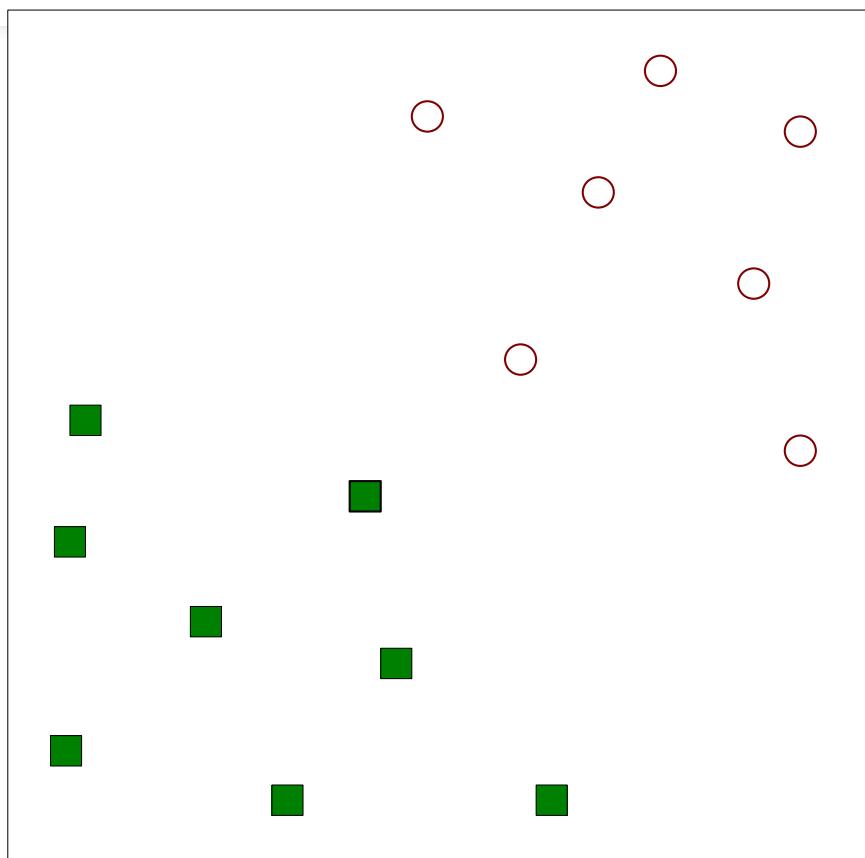
# Example of Inferencing using BBN

- Given:  $X = (E=\text{No}, D=\text{Yes}, CP=\text{Yes}, BP=\text{High})$ 
  - Compute  $P(HD|E,D,CP,BP)$ ?
- $P(HD=\text{Yes} | E=\text{No}, D=\text{Yes}) = 0.55$   
 $P(CP=\text{Yes} | HD=\text{Yes}) = 0.8$   
 $P(BP=\text{High} | HD=\text{Yes}) = 0.85$ 
  - $P(HD=\text{Yes} | E=\text{No}, D=\text{Yes}, CP=\text{Yes}, BP=\text{High})$   
 $\approx 0.55 \times 0.8 \times 0.85 = 0.374$
- $P(HD=\text{No} | E=\text{No}, D=\text{Yes}) = 0.45$   
 $P(CP=\text{Yes} | HD=\text{No}) = 0.01$   
 $P(BP=\text{High} | HD=\text{No}) = 0.2$ 
  - $P(HD=\text{No} | E=\text{No}, D=\text{Yes}, CP=\text{Yes}, BP=\text{High})$   
 $\approx 0.45 \times 0.01 \times 0.2 = 0.0009$

**Classify X  
as Yes**

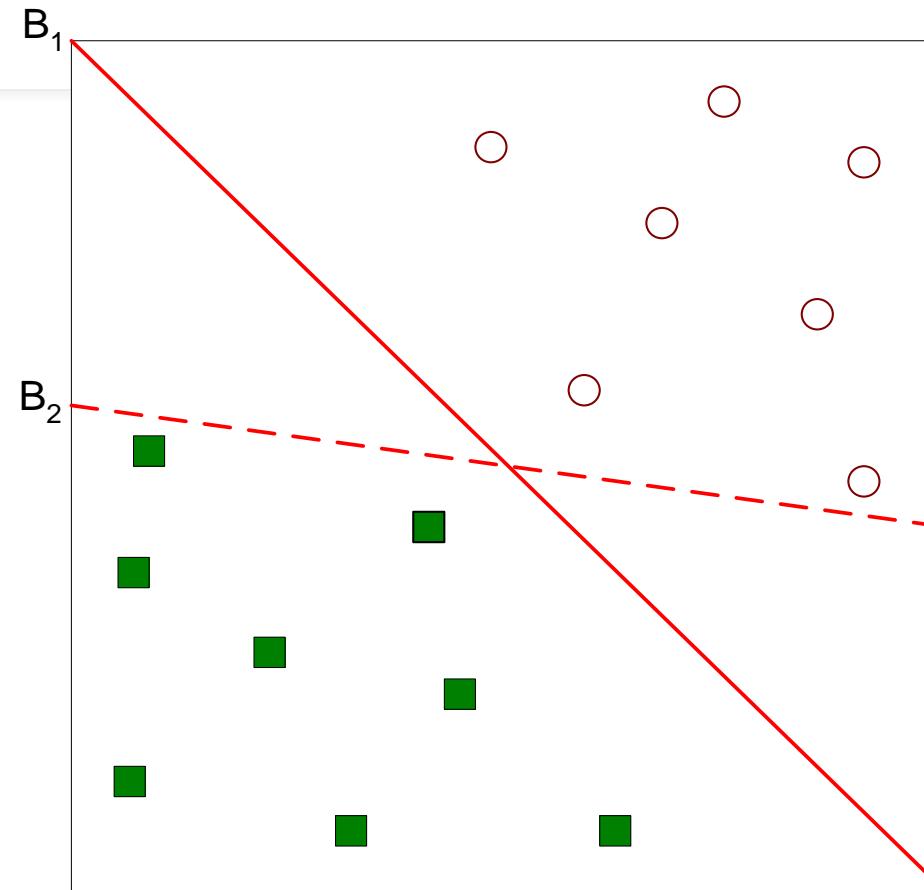


# Support Vector Machines



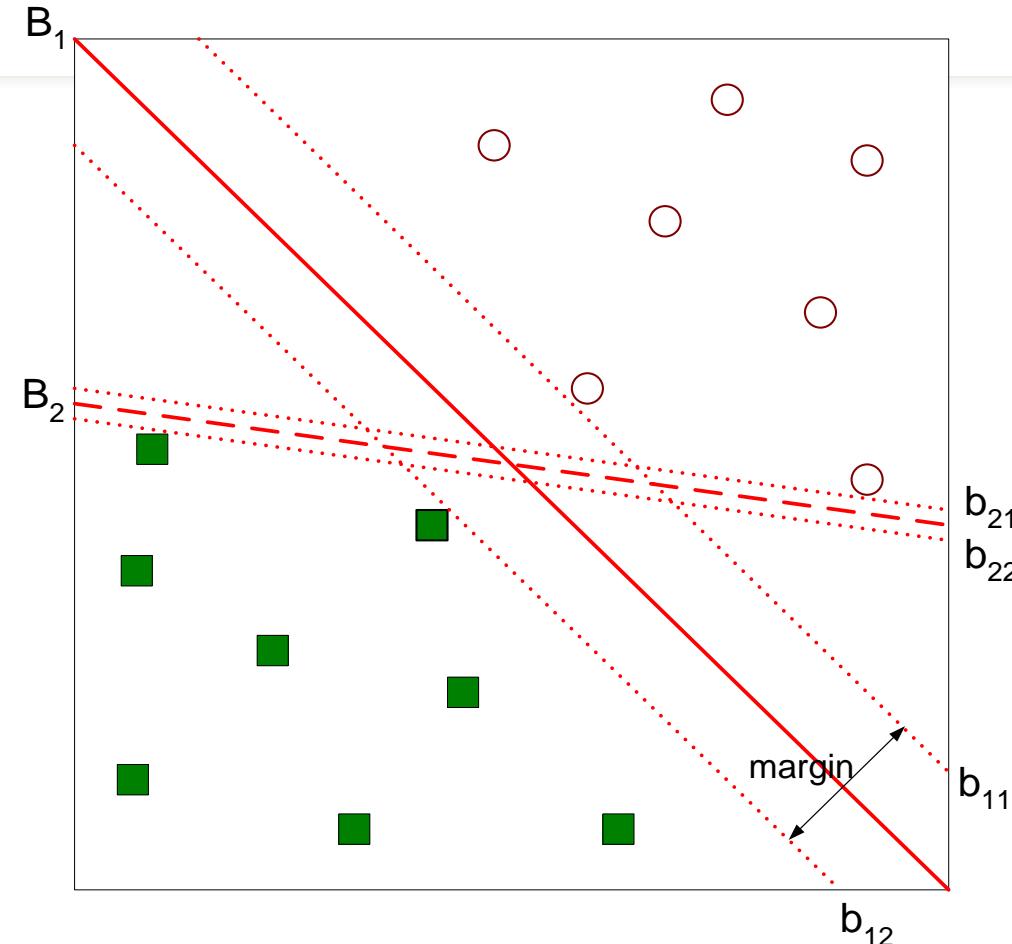
- Find a linear hyperplane (decision boundary) that will separate the data

# Support Vector Machines



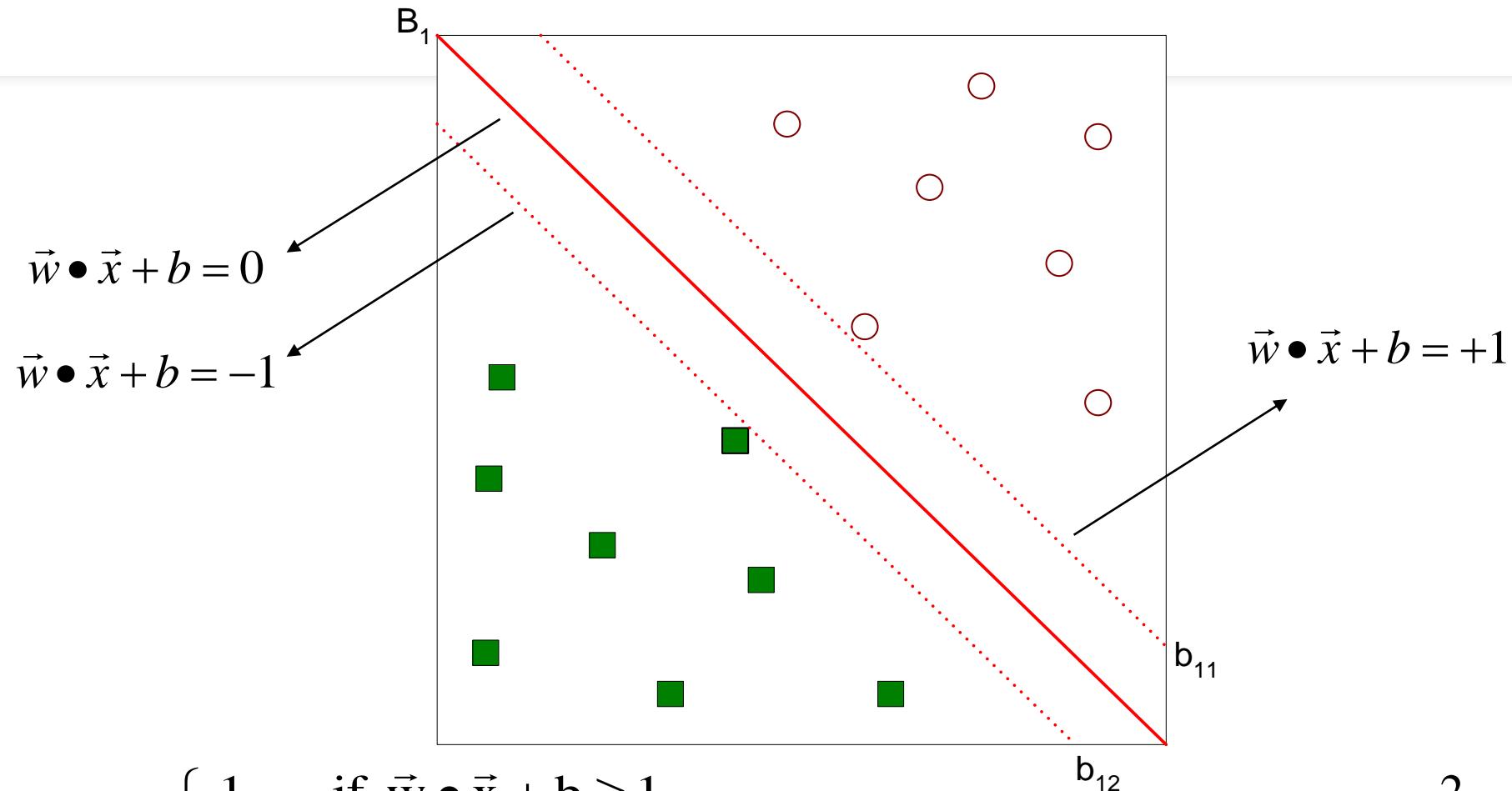
- Which one is better?  $B_1$  or  $B_2$ ?
- How do you define better?

# Support Vector Machines



- Find hyperplane **maximizes** the margin =>  $B_1$  is better than  $B_2$

# Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

# Linear SVM

- Linear model:  $f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$
- Learning the model is equivalent to determining the values of  $\vec{w}$  and  $b$ 
  - How to find  $\vec{w}$  and  $b$  from training data?

# Learning Linear SVM

- Objective is to maximize: Margin =  $\frac{2}{\|\vec{w}\|}$ 
  - Which is equivalent to minimizing:  $L(\vec{w}) = \frac{\|\vec{w}\|^2}{2}$
  - Subject to the following constraints:

or

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

$$y_i(\mathbf{w} \bullet \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

- This is a constrained optimization problem
  - Solve it using Lagrange multiplier method

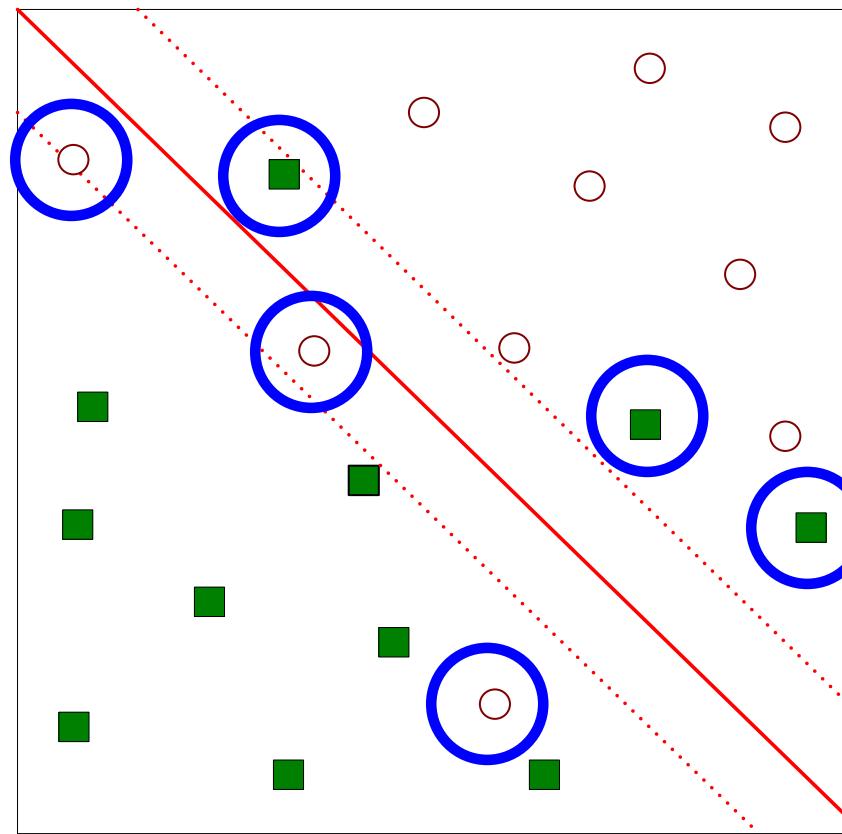
# Learning Linear SVM

- Decision boundary depends only on support vectors
  - If you have data set with same support vectors, decision boundary will not change
  - How to classify using SVM once  $\mathbf{w}$  and  $b$  are found? Given a test record,  $\mathbf{x}_i$

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

# Support Vector Machines

- What if the problem is not linearly separable?



# Support Vector Machines

- What if the problem is not linearly separable?

- Introduce slack variables

- Need to minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left( \sum_{i=1}^N \xi_i^k \right)$$

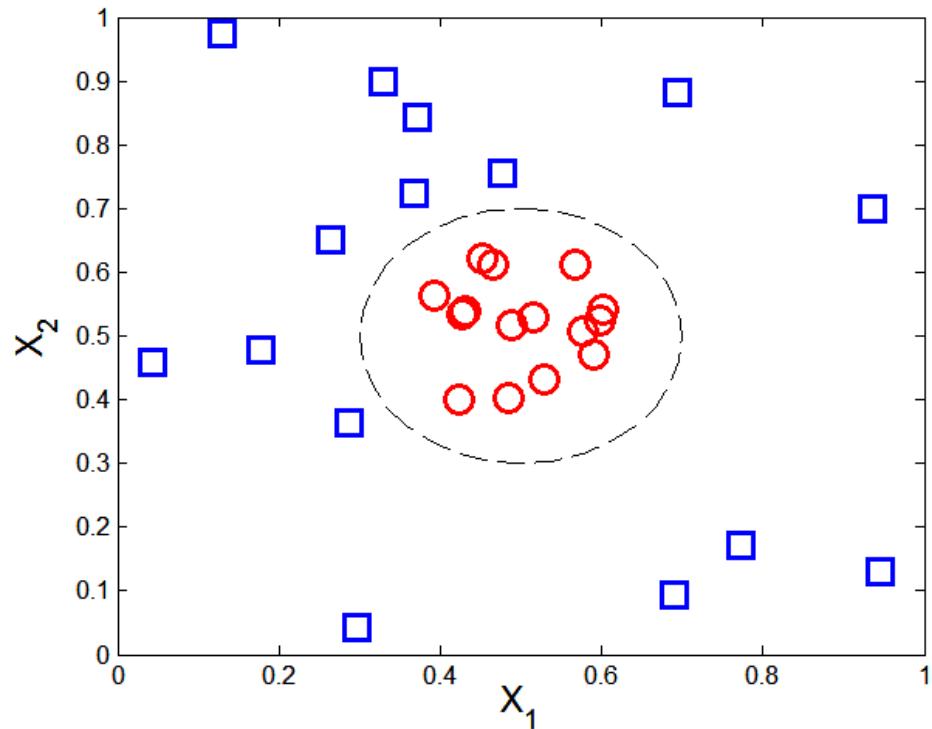
- Subject to:

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

- If k is 1 or 2, this leads to same objective function as linear SVM but with different constraints

# Nonlinear Support Vector Machines

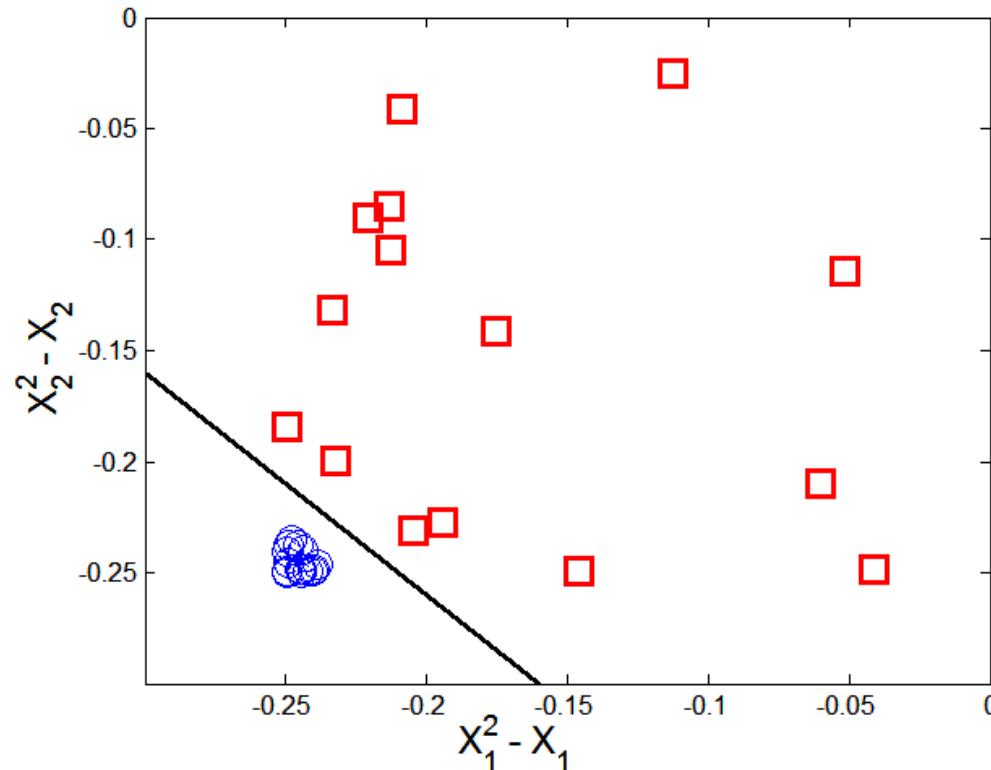
- What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

# Nonlinear Support Vector Machines

- Trick: Transform data into higher dimensional space



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \bullet \Phi(\vec{x}) + b = 0$$

# Learning Nonlinear SVM

- Optimization problem:

$$\begin{aligned} & \min_w \frac{\|\mathbf{w}\|^2}{2} \\ & \text{subject to} \quad y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \quad \forall \{(\mathbf{x}_i, y_i)\} \end{aligned}$$

- Which leads to the same set of equations (but involve  $\Phi(x)$  instead of  $x$ )

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad \lambda_i \{y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1\} = 0,$$

$$f(\mathbf{z}) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = \text{sign}(\sum_{i=1}^n \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b).$$

# Learning NonLinear SVM

- Issues:
  - What type of mapping function  $\Phi$  should be used?
  - How to do the computation in high dimensional space?
    - Most computations involve dot product  $\Phi(x_i) \bullet \Phi(x_j)$
    - Curse of dimensionality?

# Learning Nonlinear SVM

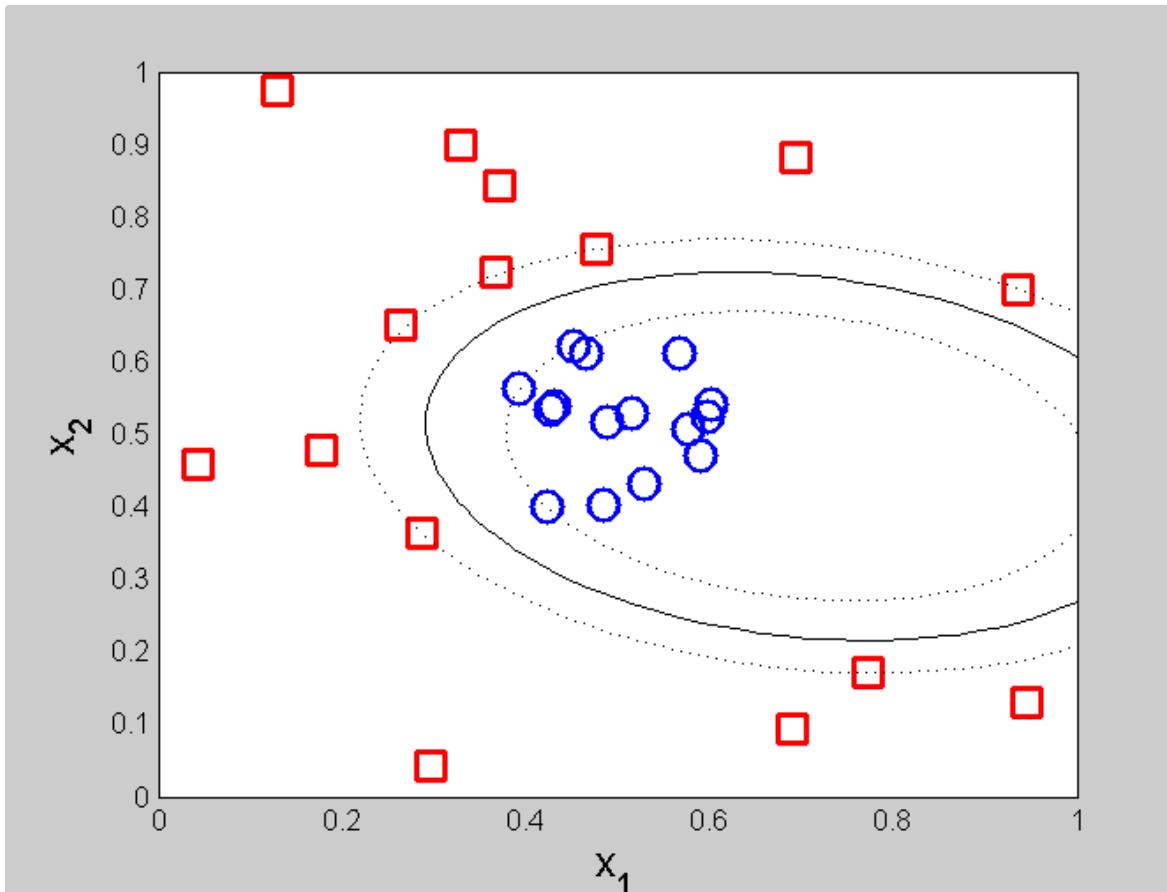
- Kernel Trick:
  - $\Phi(x_i) \bullet \Phi(x_j) = K(x_i, x_j)$
  - $K(x_i, x_j)$  is a kernel function (expressed in terms of the coordinates in the original space)
    - Examples:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$$

# Example of Nonlinear SVM



SVM with polynomial  
degree 2 kernel

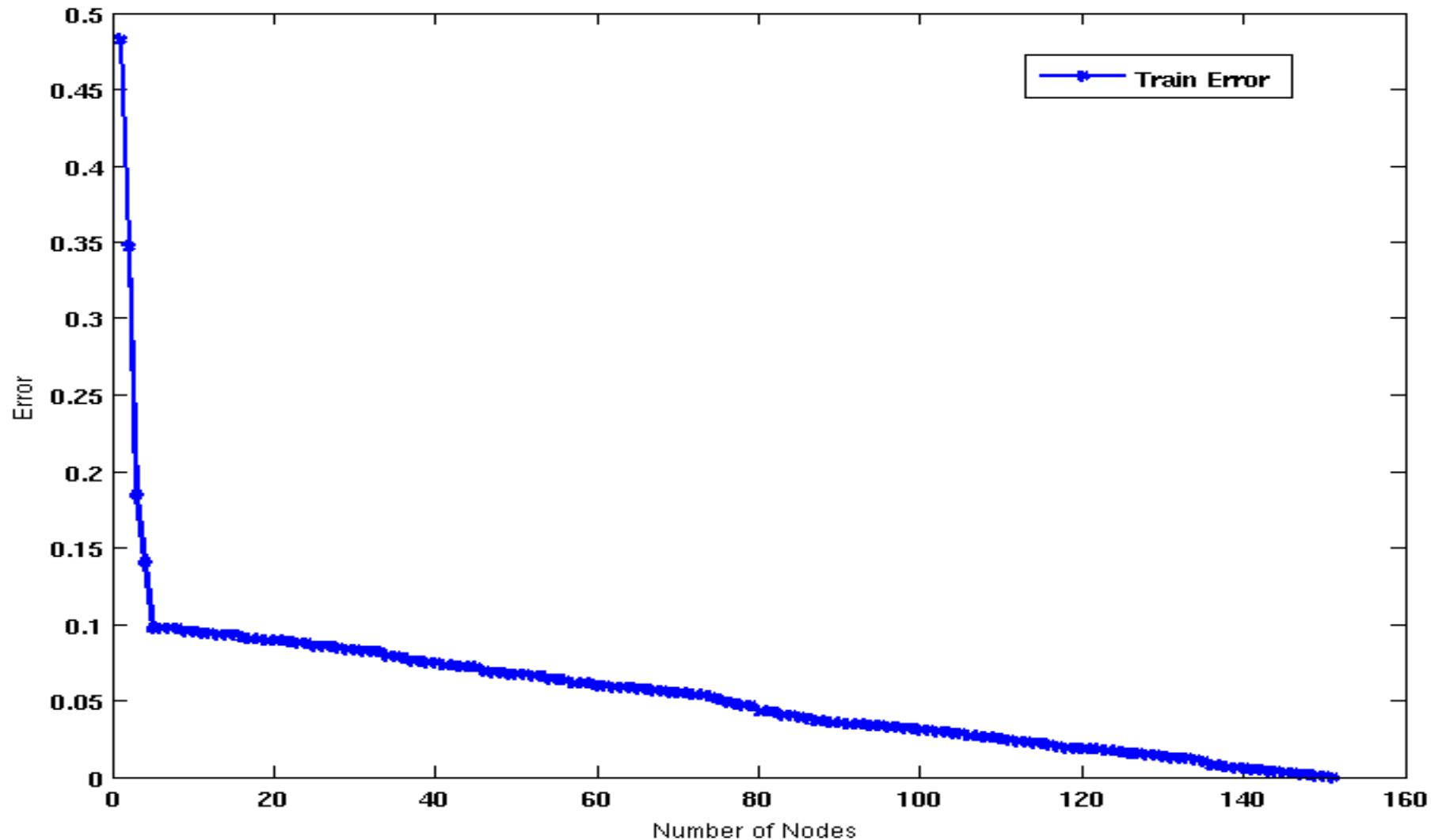
# Learning Nonlinear SVM

- Advantages of using kernel:
  - Don't have to know the mapping function  $\Phi$
  - Computing dot product  $\Phi(x_i) \bullet \Phi(x_j)$  in the original space avoids curse of dimensionality
- Not all functions can be kernels
  - Must make sure there is a corresponding  $\Phi$  in some high-dimensional space
  - Mercer's theorem (see textbook)

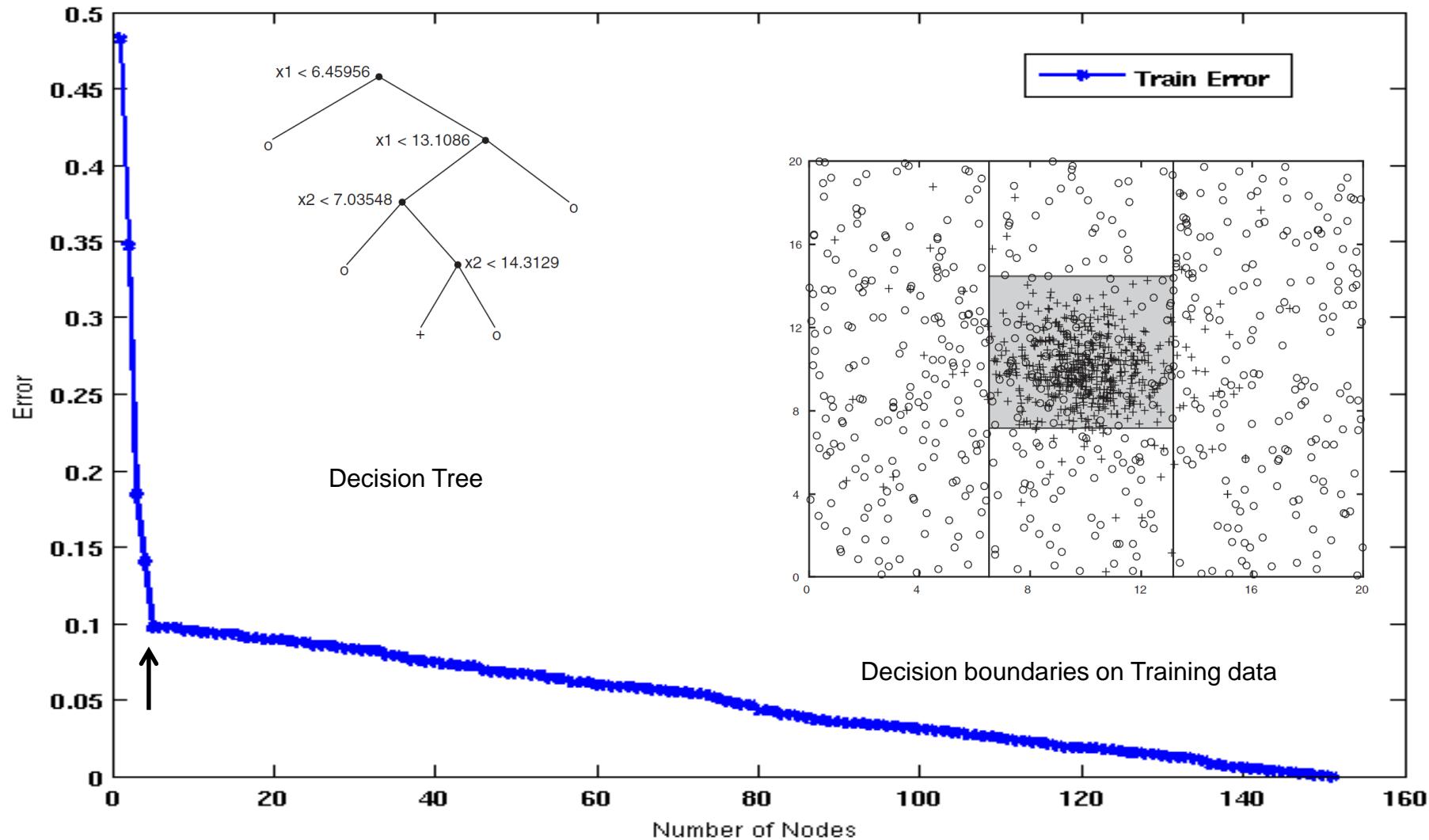
# Classification Errors

- Training errors (apparent errors)
  - Errors committed on the training set
- Test errors
  - Errors committed on the test set
- Generalization errors
  - Expected error of a model over random selection of records from same distribution

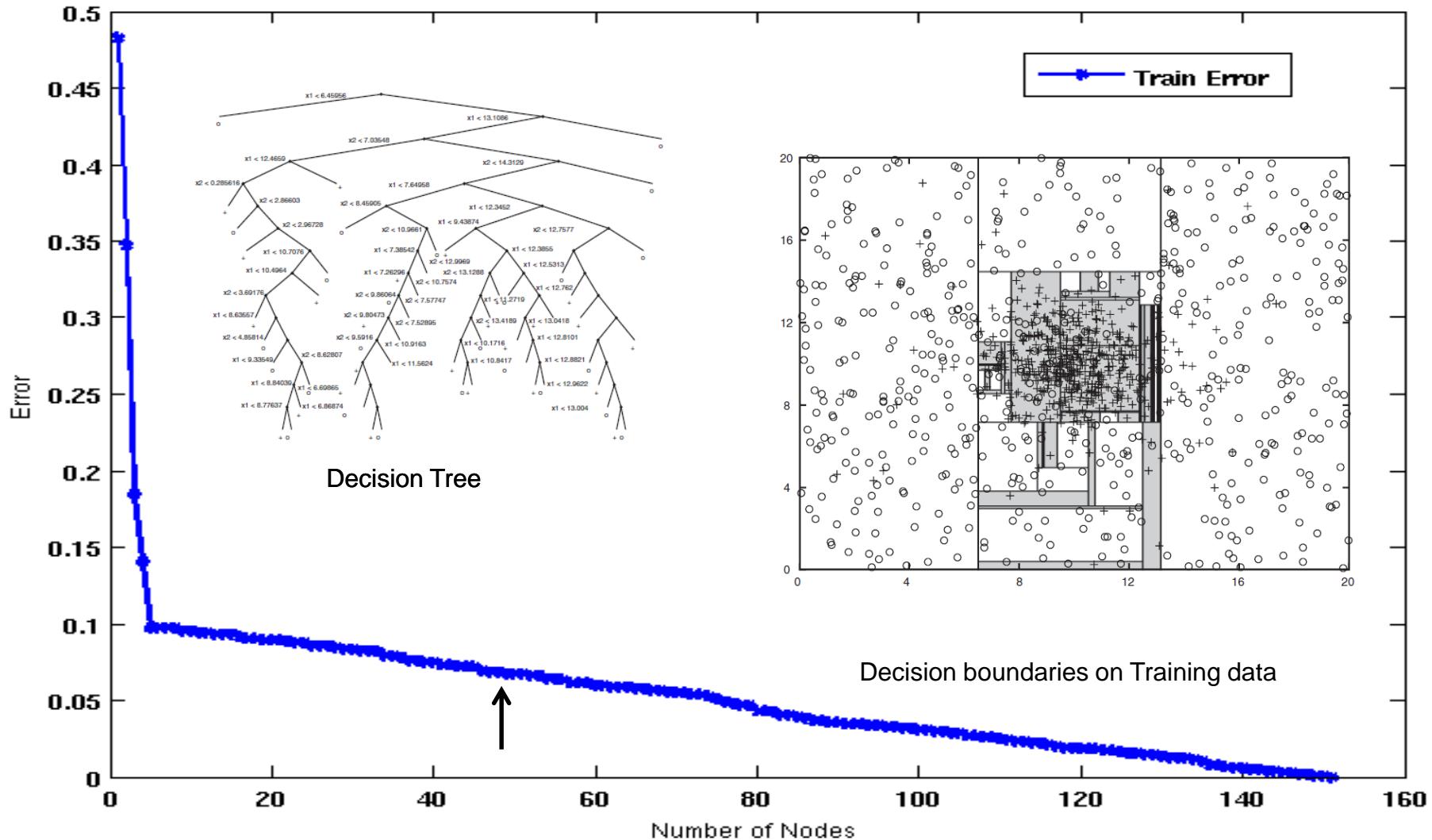
## Increasing number of nodes in Decision Trees



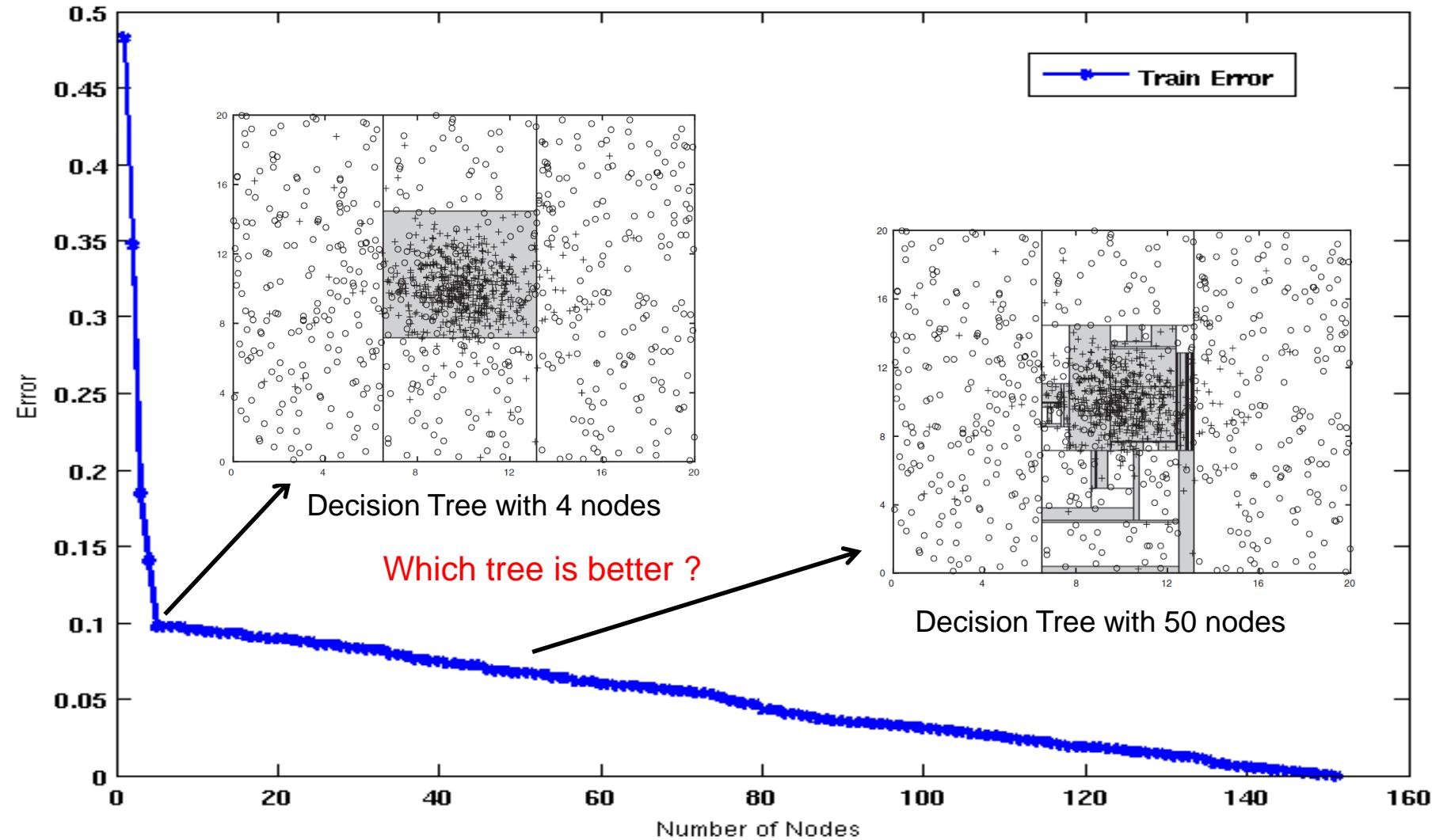
## Decision Tree with 4 nodes



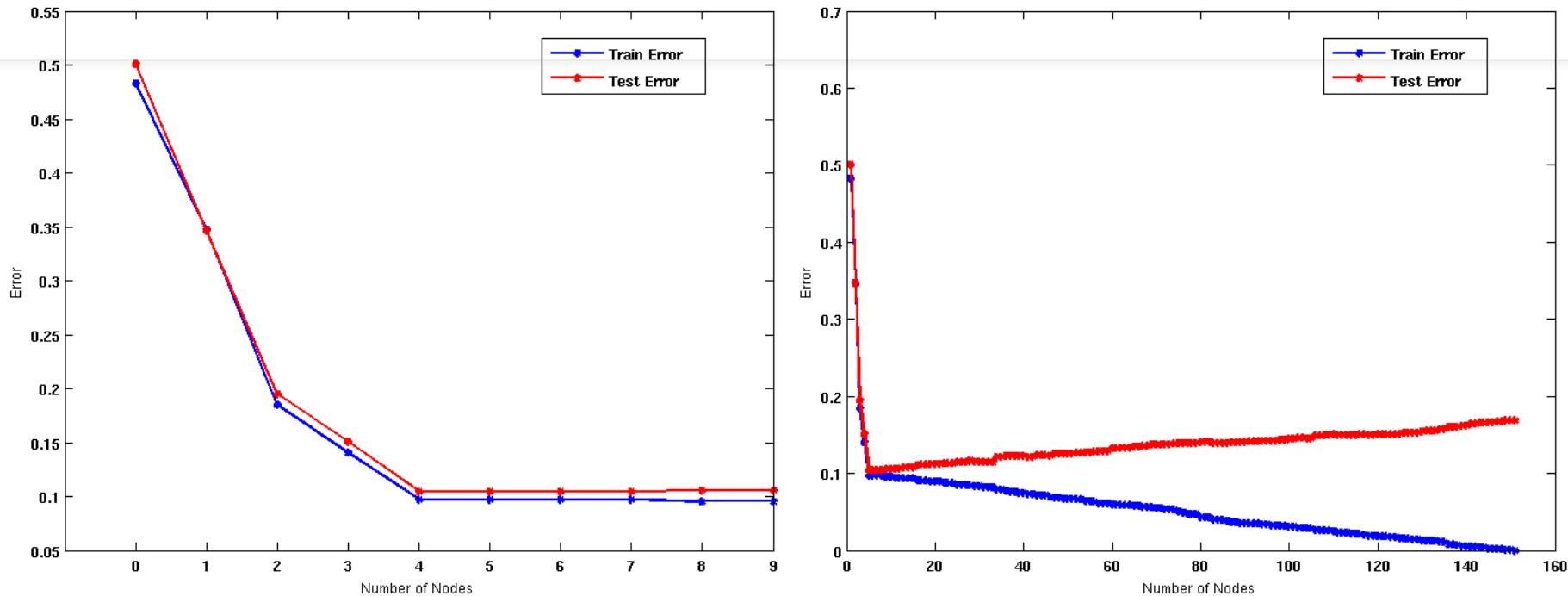
# Decision Tree with 50 nodes



# Which tree is better?



# Model Overfitting

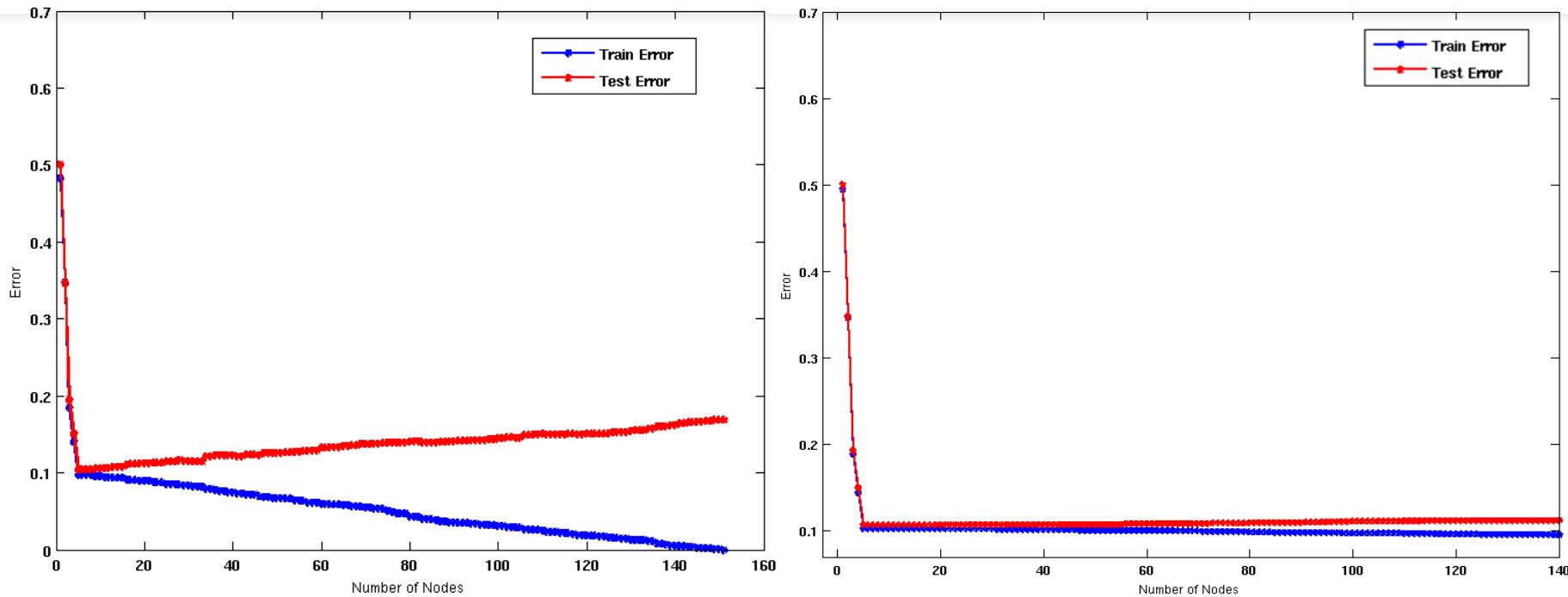


- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

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

Overfitting: when model is too complex, training error is small but test error is large

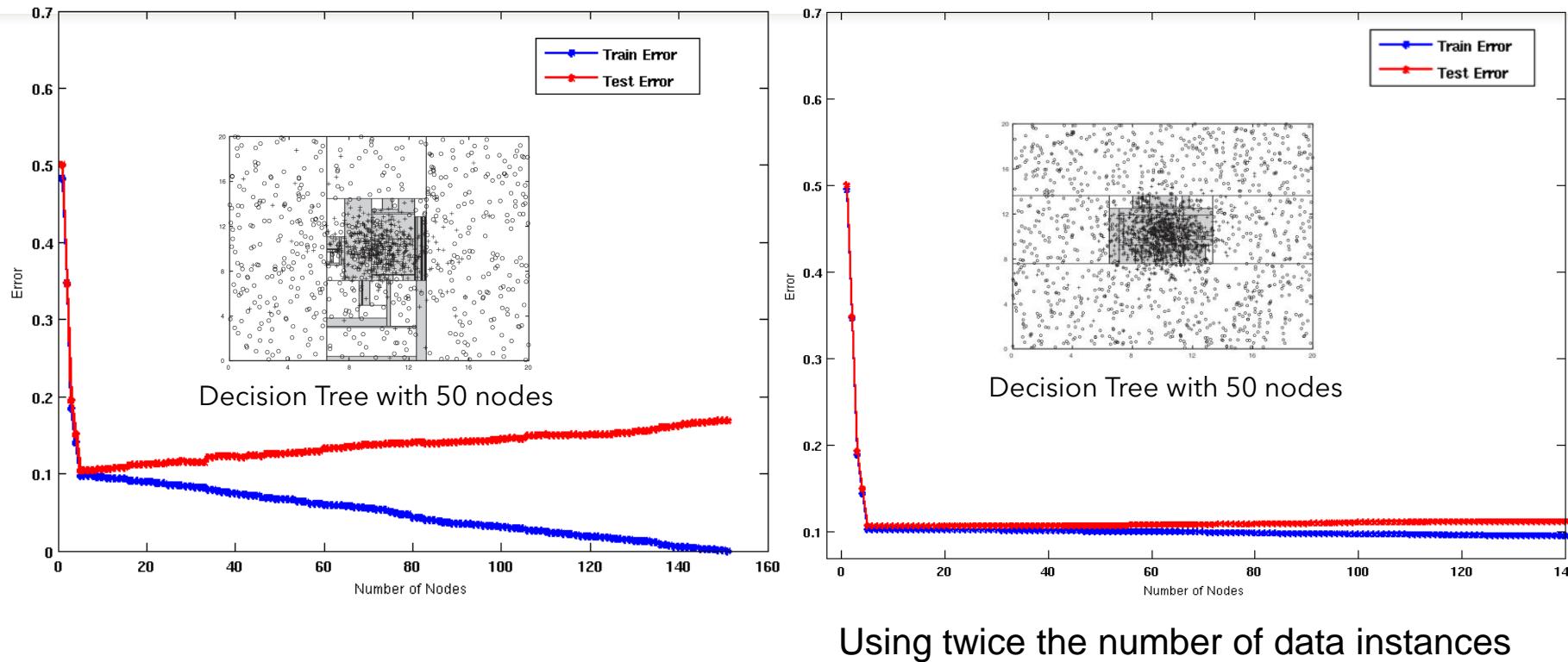
# Model Overfitting



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

# Model Overfitting



- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

# Reasons for Model Overfitting

- Limited Training Size
- High Model Complexity

# Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set

# Model Selection: Using Validation Set

- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set
- Drawback:
  - Less data available for training

# Model Selection for Decision Trees

- 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).
    - Stop if estimated generalization error falls below certain threshold

# Model Selection for Decision Trees

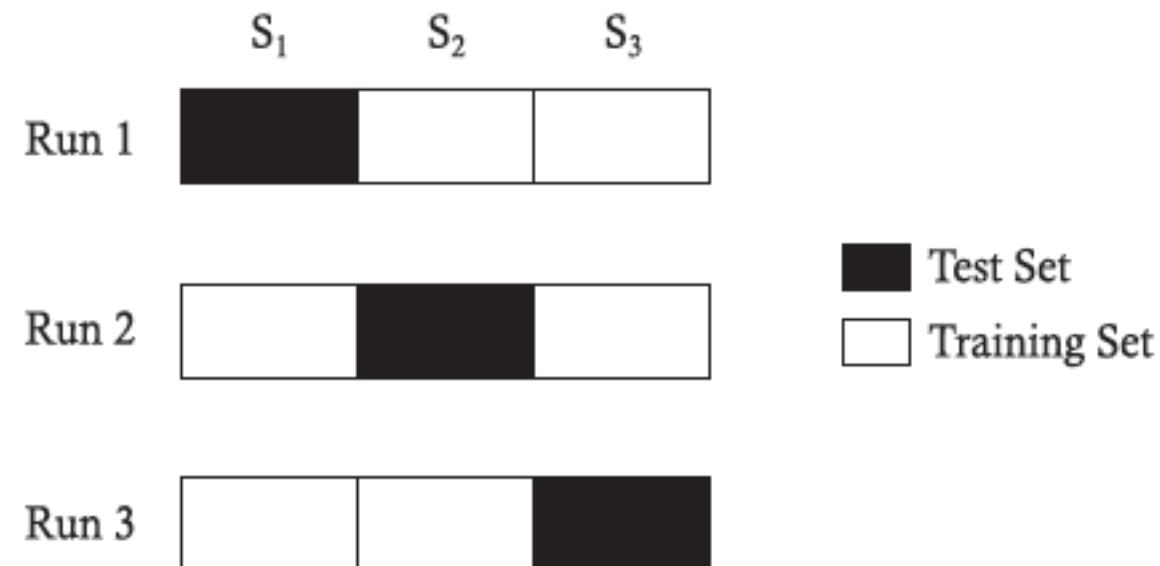
- Post-pruning
  - Grow decision tree to its entirety
  - Subtree replacement
    - Trim the nodes of the decision tree in a bottom-up fashion
    - If generalization error improves after trimming, replace sub-tree by a leaf node
    - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Subtree raising
    - Replace subtree with most frequently used branch

# Model Evaluation

- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout
  - Reserve  $k\%$  for training and  $(100-k)\%$  for testing
  - Random subsampling: repeated holdout
- Cross validation
  - Partition data into  $k$  disjoint subsets
  - $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - Leave-one-out:  $k=n$

# Cross-validation Example

- 3-fold cross-validation



# Variations on Cross-validation

- Repeated cross-validation
  - Perform cross-validation a number of times
  - Gives an estimate of the variance of the generalization error
- Stratified cross-validation
  - Guarantee the same percentage of class labels in training and test
  - Important when classes are imbalanced and the sample is small
- Use nested cross-validation approach for model selection and evaluation