

# **SC4000/CZ4041/CE4041:**

# **Machine Learning**

## **Lesson 2a: Overview of Bayesian Classifiers**

Kelly KE

College of Computing and Data Science  
NTU, Singapore

# Uncertainty in Prediction

- Recall: in supervised learning, given a set of  $\{\mathbf{x}_i, y_i\}$  for  $i = 1, \dots, N$ , the goal is to learn a mapping  $f: \mathbf{x} \rightarrow y$  by requiring  $f(\mathbf{x}_i) = y_i$
- In many applications, the mapping or relationship  $f$  between the input features and the output labels is non-deterministic (uncertain)
- For example, suppose you are asked to predict a result of the final of a football cup between Team 1 and Team 2: which team will win?

# Bayesian Classifiers

- From a probability point of view, the mapping  $f: \mathbf{x} \rightarrow y$  can be modeled as a conditional probability  $P(y|\mathbf{x})$
- Bayesian classifiers aim to learn the mapping  $f: \mathbf{x} \rightarrow y$  for supervised learning in the form of conditional probability  $P(y|\mathbf{x})$ , such that for any input  $\mathbf{x}^*$ , one can use  $P(y = c|\mathbf{x}^*)$  to predict the probability of  $\mathbf{x}^*$  belonging to class  $c$ , where  $c \in \{0, \dots, C - 1\}$ 
  - How to estimate  $P(y = c|\mathbf{x}^*)$  for different classes?
  - How to make use of  $P(y = c|\mathbf{x}^*)$ 's to make a prediction?
  - We first review some important probability concepts

# Marginal Probability

- Let  $A$  be a random variable (an input feature / class label in machine learning)
- Marginal probability

$$P(A = a) \quad 0 \leq P(A = a) \leq 1$$

refers to the probability that variable  $A = a$

$$\sum_{a_i} P(A = a_i) = 1$$

# Joint Probability

- Let  $A$  and  $B$  be a pair of random variables (features/labels in machine learning).
- Their joint probability

$$P(A = a, B = b)$$

refers to the probability that variable  $A = a$  and variable  $B = b$

# Conditional Probability

- Conditional probability:

$$P(B = b | A = a)$$

refers to the probability that the variable  $B$  will take on the value  $b$ , given that the variable  $A$  is observed to have the value  $a$

$$\sum_{b_i} P(B = b_i | A = a) = 1$$

# Sum Rule

- The connection between joint probability of  $A$  and  $B$  and marginal probability of  $A$ :

$$P(A = a) = \sum_{b_i} P(A = a, B = b_i) \quad \text{OR} \quad P(A) = \sum_B P(A, B)$$

$$P(A = a) = \sum_{c_j} \sum_{b_i} P(A = a, B = b_i, C = c_j) \quad \text{OR} \quad P(A) = \sum_C \sum_B P(A, B, C)$$

# Product Rule

- The connections between joint, conditional and marginal probabilities for  $A$  and  $B$ :

$$\begin{aligned} P(A = a, B = b) &= P(B = b|A = a) \times P(A = a) \\ &= P(A = a|B = b) \times P(B = b) \end{aligned}$$

OR

$$P(A, B) = P(B|A) \times P(A) = P(A|B) \times P(B)$$

# Bayes Rule or Bayes Theorem

$$P(A|B) = \frac{P(A, B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- This is induced from product rule

$$P(A, B) = P(A|B) \times P(B) = P(B|A) \times P(A)$$

- Generalized to the case when **A** and **B** are a set of variables

$$\begin{aligned} P(A_1 \dots A_k | B_1 \dots B_p) &= \frac{P(B_1 \dots B_p, A_1 \dots A_k)}{P(B_1 \dots B_p)} \\ &= \frac{P(B_1 \dots B_p | A_1 \dots A_k)P(A_1 \dots A_k)}{P(B_1 \dots B_p)} \end{aligned}$$

# An Example



V.S.



- Suppose the next match between the two teams will be hosted by Manchester United
- Head to Head Statistics:

Team	Played	Win	Draw	Lose
Manchester United	151	59	47	45
Manchester City	151	45	47	59

- Among the 59 victories for Manchester United, 32 of them come from playing at home
- Among the games won by Manchester City, 20 of them are obtained while playing on Manchester United home ground
- Among the draw games, 23 of them were played on Manchester United home ground

What result will likely be for the match?

# Define Variables

- Let  $Y$  be the random variable that represents the result of the match (0, 1, 2)
  - $Y = 0$ : Manchester United wins the match
  - $Y = 1$ : Manchester City wins the match
  - $Y = 2$ : Draw
- Let  $X$  be the random variable that represents the team hosting the match (0 or 1)
  - $X = 0$ : Manchester United hosts the match
  - $X = 1$ : Manchester City hosts the match
- To estimate  $P(Y = 0|X = 0)$ ,  $P(Y = 1|X = 0)$ , and  $P(Y = 2|X = 0)$

# Estimate Probabilities

Team	Played	Win	Draw	Lose
Manchester United	151	59	47	45
Manchester City	151	45	47	59

- To calculate  $P(Y = 0)$ ,  $P(Y = 1)$ , and  $P(Y = 2)$ 
  - $P(Y = 0) = \frac{59}{151} \approx 39\%$
  - $P(Y = 1) = \frac{45}{151} \approx 30\%$
  - $P(Y = 2) = 1 - P(Y = 0) - P(Y = 1) = 31\%$

$$\sum_{y_i} P(Y = y_i) = 1$$

# Estimate Probabilities (cont.)

- Among the 59 victories for Manchester United, 32 of them come from playing at home

$$P(X = 0|Y = 0) = \frac{32}{59} = 54\%$$

- Among the 45 games won by Manchester City, 20 of them are obtained while playing on Manchester United home ground

$$P(X = 0|Y = 1) = \frac{20}{45} = 44\%$$

- Among the 47 draw games, 23 of them were played on Manchester United home ground

$$P(X = 0|Y = 2) = \frac{23}{47} = 49\%$$

- However, the goal is to estimate

$$P(Y = 0|X = 0) \quad \text{v.s.} \quad P(Y = 1|X = 0) \quad \text{v.s.} \quad P(Y = 2|X = 0)$$



# Apply Bayes Rule

- Probability that Manchester United wins:  $P(Y = 0) = 0.39$
- Probability that Manchester City wins:  $P(Y = 1) = 0.3$
- Probability of a draw game:  $P(Y = 2) = 0.31$
- Probability that Manchester United hosted the match it won:  $P(X = 0|Y = 0) = 0.54$
- Probability that Manchester United hosted the match won by Manchester City:  $P(X = 0|Y = 1) = 0.44$
- Probability that Manchester United hosted the match that is a draw game:  $P(X = 0|Y = 2) = 0.49$
- To use Bayes rule to compute  
 $P(Y = 1|X = 0)$ ,  $P(Y = 0|X = 0)$  and  $P(Y = 2|X = 0)$

$$P(Y = 1|X = 0) \quad \text{Bayes rule}$$

$$= \frac{P(X = 0|Y = 1) \times P(Y = 1)}{P(X = 0)}$$

Sum rule:  $P(X) = \sum_Y P(X, Y)$

$$P(X = 0|Y = 1) \times P(Y = 1)$$

$$= P(X = 0, Y = 1) + P(X = 0, Y = 0) + P(X = 0, Y = 2)$$

$$P(X = 0|Y = 1) \times P(Y = 1)$$

$$= P(X = 0|Y = 1) \times P(Y = 1) + P(X = 0|Y = 0) \times P(Y = 0) + P(X = 0|Y = 2) \times P(Y = 2)$$

Product rule:  $P(X, Y) = P(X|Y)P(Y)$

$$= \frac{0.44 \times 0.3}{0.44 \times 0.3 + 0.54 \times 0.39 + 0.49 \times 0.31} = \frac{0.132}{0.4945} = 0.267$$

Similarly

$$P(Y = 0|X = 0) = 0.426 \quad P(Y = 2|X = 0) = 0.307$$

# Bayesian Classifiers (cont.)

- Bayesian classifiers aim to learn the mapping  $f: \mathbf{x} \rightarrow y$  for supervised learning in the form of conditional probability  $P(y|\mathbf{x})$  via Bayes rule

posterior

$$P(y|\mathbf{x}) = \frac{P(y, \mathbf{x})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

prior

- For a classification problem with  $C$  classes, given a test data instance  $\mathbf{x}^*$ , a Bayesian classifier computes

$$P(y = c|\mathbf{x}^*), c \in \{0, \dots, C - 1\}, \text{ and } \sum_c P(y = c|\mathbf{x}^*) = 1$$

- Make a prediction based on the maximum posterior

$$y^* = c^* \text{ if } c^* = \arg \max_c P(y = c|\mathbf{x}^*)$$

Return the value of  $c$  that  
maximizes  $P(y = c|\mathbf{x}^*)$

where  $c \in \{0, \dots, C - 1\}$

# Bayesian Classifiers (cont.)

- Based on Bayes rule

$$y^* = c^* \text{ if } c^* = \arg \max_c P(y = c | \mathbf{x}^*), c \in \{0, \dots, C - 1\}$$

$$= \arg \max_c \frac{P(\mathbf{x}^* | y = c)P(y = c)}{P(\mathbf{x}^*)}$$

Constant w.r.t.  
diff. values of  $c$

$$= \arg \max_c P(\mathbf{x}^* | y = c)P(y = c)$$

- Therefore, we make a prediction based on

$$y^* = c^* \text{ if } c^* = \arg \max_c P(\mathbf{x}^* | y = c)P(y = c), c \in \{0, \dots, C - 1\}$$

- Take binary classification as an example: 0 vs 1

$$P(y = 0 | \mathbf{x}) = \frac{P(\mathbf{x} | y = 0)P(y = 0)}{P(\mathbf{x})} \quad \text{vs} \quad P(y = 1 | \mathbf{x}) = \frac{P(\mathbf{x} | y = 1)P(y = 1)}{P(\mathbf{x})}$$

# Notes on Bayesian Classifiers

- Why not computing  $P(y = c|x)$  for each class directly from the training data, but using Bayes rule to estimate  $P(x|y = c)$  and  $P(y = c)$  instead?
  - To estimate  $P(y = c|x)$ , one needs to consider each combination of values of  $x$
  - E.g., suppose  $x$  is  $m$ -dimensional and each feature has binary values, then the total number of possible value combinations of  $x$  is  $2^m$  --- require a huge size of training dataset and time consuming, not practical!
  - The form of  $P(x|y = c)$  can be decomposed based on some assumptions and probability properties --- no need to consider all possible combinations

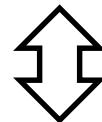
# Summary on Bayesian Classifiers

- Estimate  $P(y|\mathbf{x})$  via Bayes rule

$$P(y = c|\mathbf{x}) = \frac{P(\mathbf{x}|y = c)P(y = c)}{P(\mathbf{x})}$$

- Make predictions based on maximum posterior

$$y^* = c^* \text{ if } c^* = \arg \max_c \frac{P(\mathbf{x}|y = c)P(y = c)}{P(\mathbf{x})}$$



$$y^* = c^* \text{ if } c^* = \arg \max_c P(\mathbf{x}|y = c)P(y = c)$$

- Two implementations will be introduced

- Naïve Bayes Classifier (Lecture 3)
- Bayesian Brief Networks or Bayesian Networks (Lecture 4)

Advanced decision  
making: Bayesian  
Decision Theory  
(Lecture 2b)

How to estimate  
from training data?

# Naïve Bayes Classifiers (L3)

- How to estimate  $P(\mathbf{x}|y = c)$  from the training data
- Assume that the features are conditionally independent given the class label:

$$\boxed{P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)}$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_d]$

$$\rightarrow P(x_1, x_2, \dots, x_d | y = c) = \prod_{i=1}^d P(x_i | y = c)$$

# Bayesian Belief Networks (L4)

- A more general approach to modeling the independence and conditional independence among  $x$  and  $y$ , s.t. the computation of  $P(x, y) = P(x|y)P(y)$  is tractable
- Use a graphical representation of the probabilistic relationships among features ( $x$ ) and output class ( $y$ )

**Thank you!**

# **SC4000/CZ4041/CE4041:**

# **Machine Learning**

## **Lesson 2b: Bayesian Decision Theory**

Kelly KE

College of Computing and Data Science  
NTU, Singapore

# Decisions with Posteriors: Limitation

- By far, a decision (or prediction) is made based on the maximum posterior
  - Cost of misclassification on different classes is not taken into consideration
  - However, in some application domains, like the medical domain, the cost of misclassification on different classes may be different
  - In some applications, even correct classification incurs a cost (e.g., natural disasters)

# An Example

- To diagnose whether a patient  $A$  is with covid-19:  $y = 1$  (Yes) or  $y = 0$  (No). Suppose based on a trained Bayesian classifier, we know that  $P(y = 1|x_A) = 0.1$ . Should the doctor diagnose that  $A$  is with covid-19 or not?
  - Cost of misclassifying a healthy patient with covid-19:



- Cost of misclassifying a patient with covid-19 as healthy:

Community outbreak!

# Loss or Cost

- Actions:  $a_c$ , i.e., predict  $y = c$ , where  $c = 0, \dots, C - 1$
- Define  $\lambda_{ij}$  as the loss/cost of  $a_i$  when the optimal action is  $a_j$  (i.e., predict  $y = i$  while true class label is  $j$ )
- E.g., in the previous example,  $y = 0$ : healthy, and  $y = 1$ : with covid-19 (binary classification)
- We define two corresponding actions,  $a_0$ : predict  $y = 0$  and  $a_1$ : predict  $y = 1$ , and the losses as

$$\begin{cases} \lambda_{00} = 0 & \text{predict correctly} \\ \lambda_{11} = 0 & \text{predict correctly (sometimes this may also incur a cost)} \\ \lambda_{01} = 10 & \text{misclassify 1 as 0 (misclassify with covid-19 as healthy)} \\ \lambda_{10} = 1 & \text{misclassify 0 as 1 (misclassify healthy as with covid-19)} \end{cases}$$

# Expected Risk

- Expected risk for taking action  $a_i$ :

$$R(a_i | \mathbf{x}) = \sum_{c=0}^{C-1} \lambda_{ic} P(y = c | \mathbf{x})$$

- Explanation: to estimate a risk of taking an action, one needs to consider all the possible losses
  - Specifically, taking action  $a_i$  (predict  $\mathbf{x}$  belonging to class  $i$ ), as the ground-truth label of  $\mathbf{x}$  can be any class in the  $C$  classes, we need consider all the possible losses:  $\lambda_{i0}, \lambda_{i1}, \dots, \lambda_{i(C-1)}$
  - We can simply use the average  $\frac{\lambda_{i0} + \dots + \lambda_{i(C-1)}}{C}$  to estimate its risk
  - The possibilities of each loss occurring are different because the probabilities that  $\mathbf{x}$  belongs to each class are different
    - Use the  $P(y = c | \mathbf{x})$  as a weight for each loss  $\lambda_{ic}$ , and compute the weighted sum of all possible losses → expected risk

# An Example

- Expected risk for taking action  $a_i$ :  $R(a_i|\mathbf{x}) = \sum_{c=0}^{C-1} \lambda_{ic} P(y = c|\mathbf{x})$
- Consider the covid-19 example:
  - $P(y = 1|\mathbf{x}_A) = 0.1$  and  $P(y = 0|\mathbf{x}_A) = 0.9$
  - $a_0$ : predict  $y = 0$  (healthy), and  $a_1$ : predict  $y = 1$  (with covid-19)

$$\begin{cases} \lambda_{00} = 0 & \text{predict correctly} \\ \lambda_{11} = 0 & \text{predict correctly} \quad (\text{sometimes this may also incur a cost}) \\ \lambda_{01} = 10 & \text{misclassify 1 as 0} \quad (\text{misclassify with covid-19 as healthy}) \\ \lambda_{10} = 1 & \text{misclassify 0 as 1} \quad (\text{misclassify healthy as with covid-19}) \end{cases}$$

- Expected risk of taking action  $a_0$  (predict patient A as healthy)  
$$R(a_0|\mathbf{x}_A) = \lambda_{00}P(y = 0|\mathbf{x}_A) + \lambda_{01}P(y = 1|\mathbf{x}_A) = 1$$
- Expected risk of taking action  $a_1$   
$$R(a_1|\mathbf{x}_A) = \lambda_{10}P(y = 0|\mathbf{x}_A) + \lambda_{11}P(y = 1|\mathbf{x}_A) = 0.9$$

# Decision based on Expected Risk

- Choose the action with minimum risk:

Choose  $a^*$  if  $a^* = \arg \min_{a_c} R(a_c | \mathbf{x})$

- In the covid-19 example,
  - Expected risk of taking action  $a_0$  (predict as healthy)  
 $R(a_0 | \mathbf{x}_A) = 1$
  - Expected risk of taking action  $a_1$  (predict with covid-19)  
 $R(a_1 | \mathbf{x}_A) = 0.9$



- Thus, we choose action  $a_1$ : predict patient A is more likely with covid-19

# A Special Case

- Making predictions based on maximum posterior is a special case of making decisions based on minimum expected risk
- Define the losses as

$$\lambda_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

All correct decisions have no loss  
and all errors are equally costly

Known as the 0/1 loss

# A Special Case (cont.)

- With the 0/1 loss:  $\lambda_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$
- The expected risk of taking action  $a_i$ :

$$\begin{aligned} R(a_i | \mathbf{x}) &= \sum_{c=0}^{C-1} \lambda_{ic} P(y = c | \mathbf{x}) \\ &= \lambda_{i0} P(y = 0 | \mathbf{x}) + \cdots + \boxed{\lambda_{ii}} P(y = i | \mathbf{x}) + \cdots + \lambda_{i(C-1)} P(y = C-1 | \mathbf{x}) \\ &= P(y = 0 | \mathbf{x}) + \cdots + P(y = i-1 | \mathbf{x}) + P(y = i+1 | \mathbf{x}) \dots + P(y = C-1 | \mathbf{x}) \\ &= \sum_{j \neq i} P(y = j | \mathbf{x}) = \boxed{\sum_c P(y = c | \mathbf{x})} - P(y = i | \mathbf{x}) = 1 - P(y = i | \mathbf{x}) \\ &\quad \sum_c P(y = c | \mathbf{x}) = 1 \end{aligned}$$

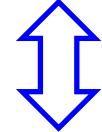
# A Special Case (cont.)

- The expected risk of taking action  $a_i$ :

$$R(a_i|\mathbf{x}) = 1 - P(y = i|\mathbf{x})$$

- Choose an action with minimum expected risk,

Choose  $a_i$  if  $R(a_i|\mathbf{x}) = \min_{a_c} R(a_c|\mathbf{x})$

 Equivalent to

Predict  $y = c^*$  if  $P(y = c^*|\mathbf{x}) = \max_c P(y = c|\mathbf{x})$

# Bayesian Decision Theory: Summary

- If cost of misclassification on different classes is available, rather than only using posterior probabilities (usually estimated by a Bayesian classifier), Bayesian decision theory provides a way to encode the cost information into decision making.

**Thank you!**

# SC4000/CZ4041/CE4041: Machine Learning

## Lesson 3: Naïve Bayes Classifiers

Kelly KE

College of Computing and Data Science  
NTU, Singapore

Acknowledgements: some tables are adapted from the lecture notes of the book  
“Introduction to Data Mining”

# Bayesian Classifiers: Recall

- To learn a prediction function via  $P(y|\mathbf{x})$  using Bayes rule

$$P(y = c|\mathbf{x}) = \frac{P(\mathbf{x}, y = c)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y = c)P(y = c)}{P(\mathbf{x})}$$

- Make predictions based on maximum posterior

$$y^* = c^* \text{ if } c^* = \arg \max_c \frac{P(\mathbf{x}|y = c)P(y = c)}{P(\mathbf{x})}$$

$$y^* = c^* \text{ if } c^* = \arg \max_c P(\mathbf{x}|y = c)P(y = c)$$

Easy to estimate

Still difficult to estimate as  $\mathbf{x}$  contains many input variables, some are discrete, and others are continuous

# Naïve Bayes Classifiers

- How to estimate  $P(\mathbf{x}|y)$  from the training data?
- Assume that the features are conditionally independent given the class label:

$$\boxed{P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)}$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_d]$

$$\rightarrow P(x_1, x_2, \dots, x_d | y = c) = \prod_{i=1}^d P(x_i | y = c)$$

# Independence

- Let  $A$  and  $B$  be two random variables.
- $A$  is said to be independent of  $B$ , if the following condition holds:

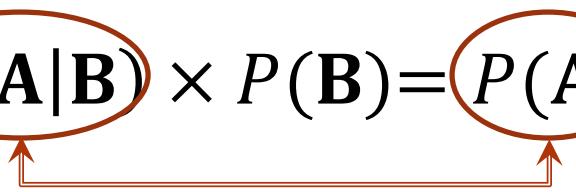
$$P(A, B) = P(A|B) \times P(B) = P(A) \times P(B)$$
$$P(A, B) = P(B|A) \times P(A) = P(A) \times P(B)$$

- E.g., let  $A$  and  $B$  denote the results of two matches in different leagues, knowing the result of one match (e.g., the value of  $A$ ) does not affect the possibility of result for the other match (i.e., the value of  $B$ ).

# Independence (cont.)

A more general case:

- Let **A** and **B** be two sets of random variables.
- The variables in **A** are said to be independent of the variables in **B**, if the following condition holds:

$$P(A, B) = P(A|B) \times P(B) = P(A) \times P(B)$$

$$P(A|B) = P(A)$$

# Conditional Independence

- Let  $A$ ,  $B$  and  $C$  be three random variables.
- $A$  is said to be conditionally independent of  $B$ , given  $C$ , if the following condition holds:

$$P(A|B, C) = P(A|C)$$

# Conditional Independence (cont.)

A more general case:

- Let **A**, **B**, and **C** be three sets of random variables.
- The variables in **A** are said to be conditionally independent of the variables in **B**, given that the variables in **C** are observed, if the following condition holds:

$$P(\mathbf{A}|\mathbf{B}, \mathbf{C}) = P(\mathbf{A}|\mathbf{C})$$

$$P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)$$



# Conditional Independence (cont.)

- The conditional independence between A and B given C can also be written as follows:

$$\begin{aligned} P(A, B|C) &= \frac{P(A, B, C)}{P(C)} && \text{Product rule} \\ &= \frac{P(A, B, C)}{P(B, C)} \times \frac{P(B, C)}{P(C)} \\ &= P(A|B, C) \times P(B|C) && \text{Product rule} \\ &= P(A|C) \times P(B|C) && \text{Conditional independence} \end{aligned}$$

$P(A, U) = P(A|U)$   
where  $U = \{B, C\}$

$P(V|C) = \frac{P(V, C)}{P(C)}$   
where  $V = \{A, B\}$

# Naïve Bayes – Induction

- The conditional independence between A and B given C can also be written as follows:

$$P(A, B|C) = P(A|C) \times P(B|C)$$



Naïve Bayes Classifier:  $P(x|y = c) = \prod_{i=1}^d P(x_i|y = c)$

Assume that the features are conditionally independent given the class label

# Naïve Bayes – Induction (cont.)

$$P(x|y = c) = P(x_1, x_2, \dots, x_d | y = c)$$

Define  $\mathbf{X}^{(d-1)} = [x_1, \dots, x_{d-1}]$

$$= P(\mathbf{X}^{(d-1)}, x_d | y = c) \quad \begin{matrix} \text{Features are conditionally} \\ \text{independent given class label} \end{matrix}$$

$$= P(\mathbf{X}^{(d-1)} | y = c)P(x_d | y = c)$$

Define  $\mathbf{X}^{(d-2)} = [x_1, \dots, x_{d-2}]$

$$= P(\mathbf{X}^{(d-2)}, x_{d-1} | y = c)P(x_d | y = c)$$

$$= P(\mathbf{X}^{(d-2)} | y = c)P(x_{d-1} | y = c)P(x_d | y = c)$$

Features are  
conditionally  
independent given  
class label

Recursively apply conditional independence

$$= P(x_1 | y = c)P(x_2 | y = c) \cdots P(x_d | y = c)$$

$$= \prod_{i=1}^d P(x_i | y = c)$$

# How Naïve Bayes Classifier Works

Naïve Bayes Classifier:  $P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)$

Easier to estimate

- To classify a test record  $\mathbf{x}^*$ , we need to compute the posteriors for each class by using

$$P(y = c|\mathbf{x}^*) = \frac{(\prod_{i=1}^d P(x_i^*|y = c))P(y = c)}{P(\mathbf{x}^*)}$$

- $P(\mathbf{x}^*)$  is constant for each class  $c$ , it is sufficient to choose the class that maximizes the numerator

$$\left( \prod_{i=1}^d P(x_i^*|y = c) \right) P(y = c)$$

# Illustrative Example

- Consider the problem of predicting whether  
→ a loan applicant will repay his/her loan obligation  
(no cheat) or become delinquent (cheat).

Example

Predefined categories



# Example (cont.)

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

Training data



$$P(y|x)$$

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

Test application  $x^*$

- To classify the application, we need to compute the posterior probabilities  $P(\text{Yes}|x^*)$  and  $P(\text{No}|x^*)$
- If  $P(\text{Yes}|x^*) > P(\text{No}|x^*)$  then classified as Yes
- Otherwise classified as No

# Estimate Priors

- Class: #instances in class  $c$

$$P(y = c) = \frac{|y = c|}{N}$$

- e.g.,

$$P(\text{No}) = 0.7$$

$$P(\text{Yes}) = 0.3$$

#training instances

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

# Estimate Conditional Probabilities for Discrete Features

#instances in class  $c$ , whose values of feature  $x_i$  are  $k$

$$P(x_i = k | y = c) = \frac{|(x_i = k) \wedge (y = c)|}{|y = c|}$$

A specific value  $k$  of the feature  $x_i$

$$P(\text{Status} = \text{Married} | \text{Cheat} = \text{No})$$

$$= \frac{\#(\text{Status} = \text{Married} \wedge \text{Cheat} = \text{No})}{\#(\text{Cheat} = \text{No})} = \frac{4}{7}$$

$$P(\text{Home Owner} = \text{Yes} | \text{Cheat} = \text{Yes})$$

$$= \frac{\#(\text{Home Owner} = \text{Yes} \wedge \text{Cheat} = \text{Yes})}{\#(\text{Cheat} = \text{Yes})} = \frac{0}{3} = 0$$

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

# Estimate Conditional Probabilities for Continuous Features

- For continuous features:
  - Probability density estimation (more details will be introduced in the 2<sup>nd</sup> half of the semester):
    - Assume the values of a feature given a class label follow a Gaussian distribution, i.e., assume  $P(x_i|y = c)$  is a Gaussian distribution
    - Use training data in the class  $c$  to estimate parameters of distribution (e.g., mean  $\mu$  and variance  $\sigma^2$ )
    - Once probability density function is known, we can use it to estimate the conditional probability

# Estimate Conditional Probabilities for Continuous Features (cont.)

- For each class  $c$ , assume values of the feature  $x_i$  follow a Gaussian distribution:

$$P(x_i|y = c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}}$$

The variance of  $x_i$  of the training data in class  $c$

The mean of  $x_i$  of the training data in class  $c$

The diagram shows the Gaussian probability formula. A red oval highlights the term  $(x_i - \mu_{ic})^2$ . A red arrow points from this oval to the text "The mean of  $x_i$  of the training data in class  $c$ ". A blue circle highlights the term  $\sqrt{2\pi\sigma_{ic}^2}$ . A blue arrow points from this circle to the text "The variance of  $x_i$  of the training data in class  $c$ ".

- Suppose there are  $N_c$  instances in class  $c$ , then

$$\mu_{ic} = \frac{1}{N_c} \sum_{j=1}^{N_c} x_{ij}$$
$$\sigma_{ic}^2 = \frac{1}{N_c - 1} \sum_{j=1}^{N_c} (x_{ij} - \mu_{ic})^2$$

Value of feature  $x_i$  of the  $j$ -th training data in class  $c$

# Estimate Conditional Probabilities for Continuous Features (cont.)

$$P(x_i|y = c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}}$$

- For {Income, Cheat = No}:
  - sample mean = 110
  - sample variance = 2975  
(standard deviation = 54.54)

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

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

Income

# Estimate Conditional Probabilities for Continuous Features (cont.)

- The estimated Gaussian distribution for {Income, Cheat = No}:

A function of  $x_i$

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

Tid	Home Owner	Marital Status	Taxable Income	Cheat
4	Yes	Married	120K	No

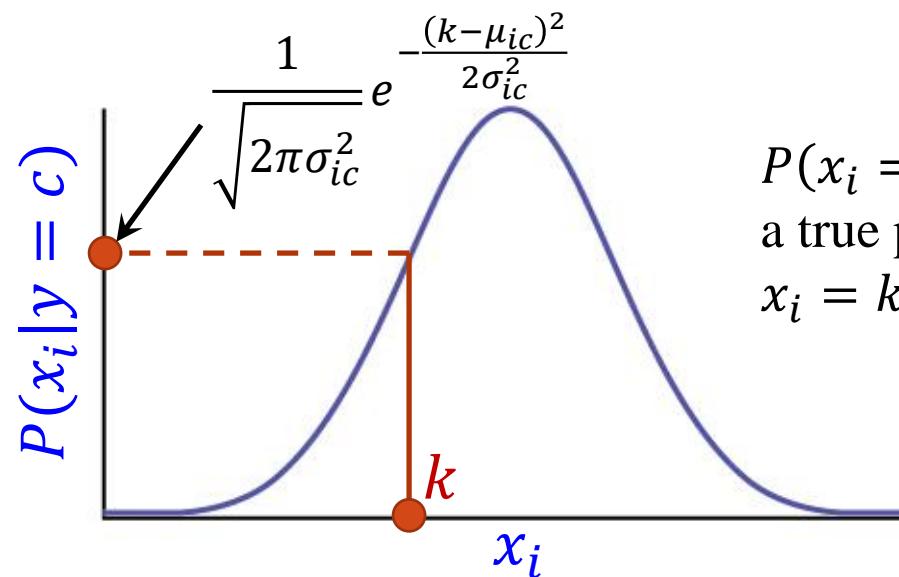
$$P(\text{Income} = 120|\text{No}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(120 - 110)^2}{2 \times 2975}} = 0.0072$$

Note: in practice, 0.0072 can be used as to approximate the conditional probability, but in theory it is not a true probability

# Additional Notes

Probability density function  $P(x_i|y = c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}}$

- The probability density function is continuous, the probability is defined as the area under the curve of the probability density function



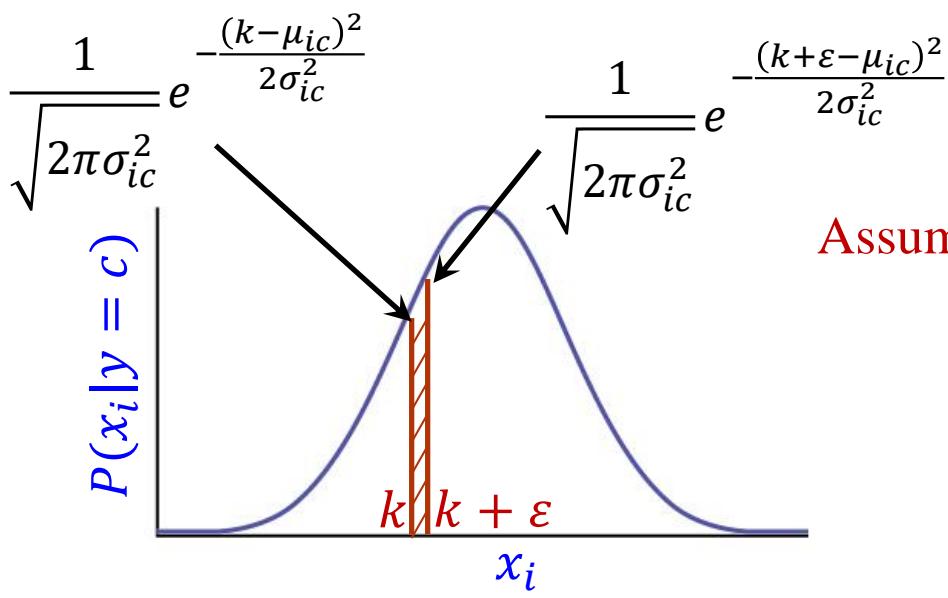
$P(x_i = k|y = c)$  is not a true probability that  $x_i = k$  for class  $c$

# Additional Notes (cont.)

- Instead, we should compute

$$P(k \leq x_i \leq k + \varepsilon | y = c) = \int_k^{k+\varepsilon} \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i - \mu_{ic})^2}{2\sigma_{ic}^2}} dx_i$$

Small positive constant



Assume  $\frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k+\varepsilon-\mu_{ic})^2}{2\sigma_{ic}^2}}$

$$\approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \times \varepsilon$$

# Additional Notes (cont.)

- Since  $\varepsilon$  appears as a constant multiplicative factor for each class, it cancels out when comparing posterior probabilities  $P(y = c|x)$  for each class
- E.g., consider binary classification and instance is represented by a single feature of continuous values

$$P(y = 0|x = k) \quad vs. \quad P(y = 1|x = k)$$



$$P(x = k|y = 0)P(y = 0) \quad vs. \quad P(x = k|y = 1)P(y = 1)$$



$$\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(k-\mu_0)^2}{2\sigma_0^2}} \times \cancel{\varepsilon} \times P(y = 0) \quad vs. \quad \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(k-\mu_1)^2}{2\sigma_1^2}} \times \cancel{\varepsilon} \times P(y = 1)$$

# Additional Notes (cont.)

- Therefore, we can still apply the following equation to approximate the probability of  $x_i = k$  for class  $c$

$$P(x_i = k | y = c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

# Example of Naïve Bayes Classifier

## Naïve Bayes Classifier:

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

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

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

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

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

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

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

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

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

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

For taxable income:

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

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

$$P(\text{Class} = \text{No}) = 7/10$$

$$P(\text{Class} = \text{Yes}) = 3/10$$

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

$$P(x|y = c) = \prod_{i=1}^d P(x_i|y = c)$$

Test example:

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

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

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

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

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

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

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

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

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

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

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

For taxable income:

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

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

$$P(\text{Class} = \text{No}) = 7/10$$

$$P(\text{Class} = \text{Yes}) = 3/10$$

$$P(\mathbf{x}^*|y=c) = \prod_{i=1}^d P(x_i^*|y=c)$$

$$\begin{aligned} P(\mathbf{x}^*|\text{Class}=\text{No}) &= P(\text{HomO}=\text{No}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Status}=\text{Married}|\text{Class}=\text{No}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{No}) \end{aligned}$$

$$= \frac{4}{7} \times \frac{4}{7} \times 0.0072 = 0.0024$$

$$\begin{aligned} P(\mathbf{x}^*|\text{Class}=\text{Yes}) &= P(\text{HomO}=\text{No}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Status}=\text{Married}|\text{Class}=\text{Yes}) \\ &\quad \times P(\text{Income}=120\text{K}|\text{Class}=\text{Yes}) \end{aligned}$$

$$= 1 \times 0 \times (1.2 \times 10^{-9}) = 0$$

$$\begin{aligned} P(\mathbf{x}^*|\text{No}) \times P(\text{No}) &= 0.0024 \times 0.7 = 0.00168 \\ &> P(\mathbf{x}^*|\text{Yes}) \times P(\text{Yes}) = 0 \times 0.3 = 0 \end{aligned}$$

Therefore  $P(\text{No}|\mathbf{x}^*) > P(\text{Yes}|\mathbf{x}^*)$  **Class = No**

# Laplace Estimate

- Alternative probability estimation (discrete features):

Original:  $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)|}{|y = c|}$

Laplace:  $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + 1}{|y = c| + n_i}$

#distinct  
values of  $x_i$

$$P(\text{Married}|\text{Yes}) = \frac{\#(\text{Married} \wedge \text{Yes})}{\#(\text{Yes})} = \frac{0}{3}$$

$$P(\text{Married}|\text{Yes}) = \frac{\#(\text{Married} \wedge \text{Yes}) + 1}{\#(\text{Yes}) + 3} = \frac{1}{6}$$

The same to  $P(\text{Single}|\text{Yes})$  and  $P(\text{Divorced}|\text{Yes})$

Extreme case - no training data:

$$P(\text{Single}|\text{Yes}) = P(\text{Married}|\text{Yes}) = P(\text{Divorced}|\text{Yes}) = \frac{1}{3}$$

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

# M-estimate

- A more general estimation:

$$\text{Original: } P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)|}{|y = c|}$$

$$\text{M-estimate: } P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + m \times p}{|y = c| + m}$$

e.g., if prior information of  $P(x_i = k|y = c)$  is available, then we can set  $p$  as the prior

User-specified parameters

For example, based on domain knowledge, you have the prior information:

Domain knowledge,  
not learned from data

$$\tilde{P}(\text{Single|Yes}) = \frac{1}{2} \quad \tilde{P}(\text{Divorced|Yes}) = \frac{1}{3} \quad \tilde{P}(\text{Married|Yes}) = \frac{1}{6}$$

Extreme case - no training data:

$$P(\text{Single|Yes}) = \frac{\#(\text{Single} \wedge \text{Yes}) + m \times \tilde{P}(\text{Single|Yes})}{\#(\text{Yes}) + m} = \frac{m \times \tilde{P}(\text{Single|Yes})}{m} = \tilde{P}(\text{Single|Yes})$$

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

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

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

$$P(\text{HomO}=\text{Yes}|\text{Yes}) = 0/3$$

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

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

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

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

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

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

$$P(\text{Marital Status} = \text{Married}|\text{Yes}) = 0/3$$

For taxable income:

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

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

$$P(\text{Class} = \text{No}) = 7/10$$

$$P(\text{Class} = \text{Yes}) = 3/10$$

$$m = 3$$

$p = 1/3$  for all discrete features of class **Yes**

$p = 2/3$  for all discrete features of class **No**

$$P(\text{HomO}=\text{Yes}|\text{No}) = ?$$

$$P(\text{HomO}=\text{No}|\text{No}) = ?$$

$$P(\text{HomO}=\text{Yes}|\text{Yes}) = ?$$

$$P(\text{HomO}=\text{No}|\text{Yes}) = ?$$

$$P(\text{Marital Status} = \text{Single}|\text{No}) = ?$$

$$P(\text{Marital Status} = \text{Divorced}|\text{No}) = ?$$

$$P(\text{Marital Status} = \text{Married}|\text{No}) = ?$$

$$P(\text{Marital Status} = \text{Single}|\text{Yes}) = ?$$

$$P(\text{Marital Status} = \text{Divorced}|\text{Yes}) = ?$$

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

**M-estimate**

$$P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + m \times p}{|y = c| + m}$$

$$P(\mathbf{x}^*|\text{Class} = \text{No}) = ? \quad P(\mathbf{x}^*|\text{Class} = \text{Yes}) = ?$$



Tutorial

# Implementation using scikit-learn

- API: `sklearn.naive_bayes`: Naive Bayes

[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive\\_bayes](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes)

## `sklearn.naive_bayes`: Naive Bayes

The `sklearn.naive_bayes` module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

**User guide:** See the [Naive Bayes](#) section for further details.

`naive_bayes.BernoulliNB(*  
[, alpha, ...])` Naive Bayes classifier for multivariate Bernoulli models.

`naive_bayes.CategoricalNB(*  
[, alpha, ...])` Naive Bayes classifier for categorical features

`naive_bayes.ComplementNB(*  
[, alpha, ...])` The Complement Naive Bayes classifier described in Rennie et al.

`naive_bayes.GaussianNB(*  
[, priors, ...])` Gaussian Naive Bayes (GaussianNB)

`naive_bayes.MultinomialNB(*  
[, alpha, ...])` Naive Bayes classifier for multinomial models

Documentation: [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

# Mixed Naïve Bayes Implementation

<https://pypi.org/project/mixed-naive-bayes/#installation>

The screenshot shows the PyPI project page for 'mixed-naive-bayes' version 0.0.1. The header includes the project name, a 'Latest version' button (which is green with a checkmark), and a release date of 'Oct 13, 2019'. Below the header, there's a brief description: 'Categorical and Gaussian Naïve Bayes'. The left sidebar has a 'Navigation' section with 'Project description' (selected), 'Release history', and 'Download files'. The main content area is titled 'Project description' and contains the following text:  
**Mixed Naïve Bayes**  
Naïve Bayes classifiers are a set of supervised learning algorithms based on applying Bayes' theorem, but with strong independence assumptions between the features given the value of the class variable (hence naïve).  
This module implements **Categorical** (Multinoulli) and **Gaussian** naïve Bayes algorithms (hence *mixed naïve Bayes*). This means that we are not confined to the assumption that features (given their respective *y*'s) follow the Gaussian distribution, but also the categorical distribution. Hence it is natural that the continuous data be attributed to the Gaussian and the categorical data (nominal or ordinal) be attributed to the categorical distribution.  
The motivation for writing this library is that [scikit-learn](#) does not have an implementation for mixed type of naïve bayes. They have one for [CategoricalNB](#) [here](#) but it's still pending.

```
>>> from mixed_naive_bayes import MixedNB
```

```
>>> nbC = MixedNB(categorical_features=[0,1,3])
```

```
>>> nbC.fit(X, y)
```

```
>>> nbC.predict(X)
```

Specify which columns  
are categorical features

# Naïve Bayes Classifier: Summary

- Based on a very strong assumption on conditional independence: all the input features are independent to each other given a class label

$$P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)$$

- Computationally efficient
- Independence assumption may not hold in practice (for most of time), that is why it is called “naïve”
  - Correlated features can degrade the performance
  - To be continued ...

**Thank you!**

# **SC4000/CZ4041/CE4041:**

# **Machine Learning**

## **Lesson 4: Bayesian Belief Networks**

Kelly KE

College of Computing and Data Science  
NTU, Singapore

Acknowledgements: some figures are adapted from the lecture notes of the books  
“Introduction to Data Mining” (Chap. 5).

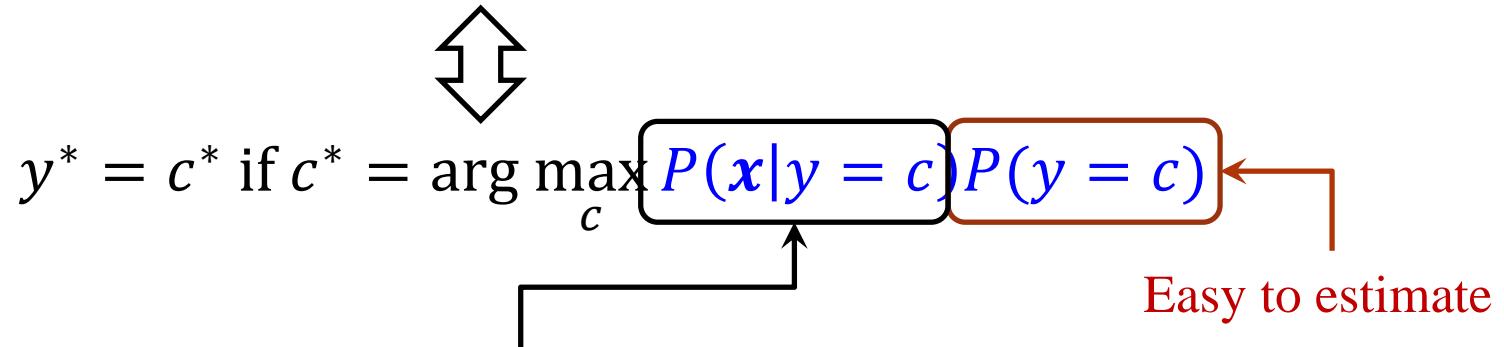
# Bayesian Classifiers: Recall

- Estimate  $P(y|\mathbf{x})$  via Bayes rule:

$$P(y = c|\mathbf{x}) = \frac{P(\mathbf{x}, y = c)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y = c)P(y = c)}{P(\mathbf{x})}$$

- Make predictions based on maximum posterior:

$$y^* = c^* \text{ if } c^* = \arg \max_c P(y = c|\mathbf{x}) \quad \text{the 0/1 loss}$$



Still difficult to estimate.  $\mathbf{x}$  contains many input variables. Some are discrete, and others are continuous.

# Naïve Bayes Classifier

- To make estimation of  $P(\mathbf{x}|y)$  from training data tractable, Naïve Bayes classifiers assume features are conditionally independent given the class label:

$$P(\mathbf{x}|y = c) = \prod_{i=1}^d P(x_i|y = c)$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_d]$

- The conditional independence assumption may not hold in practice.
  - Correlated features can degrade the performance.

# An Example

- Suppose the probability of a person having a specific disease  $D$  is 50%.
- There are two medical tests,  $T_1$  and  $T_2$ , of binary values (positive or negative). The outcomes of  $T_2$  are perfectly positively correlated with  $T_1$  if a person has the disease, but are independent of  $T_1$  if the person does not have the disease.



When a person has the disease, the outcomes of  $T_1$  and  $T_2$  are both positive (or negative).

- If a person has the disease, the probabilities of tests  $T_1$  and  $T_2$  being negative are 40%, respectively.
- If a person does not have the disease, the probabilities of  $T_1$  and  $T_2$  being negative are 60% and 65%, respectively.

# An Example (cont.)

- If the two tests  $T_1$  and  $T_2$  are both negative for a particular patient, diagnose whether the patient has the disease?

# Variables Definition

- Let  $X_1$  denote the outcome of  $T_1$ 
  - $X_1 = 1$ : positive
  - $X_1 = 0$ : negative
- Let  $X_2$  denote the outcome of  $T_2$ 
  - $X_2 = 1$ : positive
  - $X_2 = 0$ : negative
- Let  $Y$  denote whether a person has the disease  $D$ 
  - $Y = 1$ : yes
  - $Y = 0$ : no

# Probabilities

- Suppose that the probability of a person having a specific disease  $D$  is 50%.

$$P(Y = 0) = 50\% \text{ and } P(Y = 1) = 50\%$$

- The outcomes of  $T_2$  are perfectly positively correlated with  $T_1$  if a person has the disease.
  - Given that a person has the disease, if the outcome of  $T_1$  is positive then the outcome of  $T_2$  is always positive, and if the outcome of  $T_1$  is negative then the outcome of  $T_2$  is always negative.

$$\begin{aligned} P(X_1 = 1, X_2 = 1 | Y = 1) &= \frac{|(X_1 = 1) \wedge (X_2 = 1) \wedge (Y = 1)|}{|Y = 1|} \\ &= \frac{|(X_1 = 1) \wedge (Y = 1)|}{|Y = 1|} = P(X_1 = 1 | Y = 1) \end{aligned}$$

$$P(X_1 = 0, X_2 = 0 | Y = 1) = P(X_1 = 0 | Y = 1)$$

# Probabilities (cont.)

- The outcomes of  $T_2$  are perfectly positively correlated with  $T_1$  if a person has the disease, **but are independent of  $T_1$  if the person does not have the disease.**
  - Given that a person does not have the disease, the outcomes of  $T_1$  and  $T_2$  are independent.

$$P(X_1 = 1, X_2 = 1 | Y = 0) = P(X_1 = 1 | Y = 0)P(X_2 = 1 | Y = 0)$$

$$P(X_1 = 1, X_2 = 0 | Y = 0) = P(X_1 = 1 | Y = 0)P(X_2 = 0 | Y = 0)$$

$$P(X_1 = 0, X_2 = 1 | Y = 0) = P(X_1 = 0 | Y = 0)P(X_2 = 1 | Y = 0)$$

$$P(X_1 = 0, X_2 = 0 | Y = 0) = P(X_1 = 0 | Y = 0)P(X_2 = 0 | Y = 0)$$

- The conditional independence assumption in Naïve Bayes Classifiers holds when a person does not have the disease.

# Probabilities (cont.)

- If a person has the disease, the probabilities of tests  $T_1$  and  $T_2$  being negative are 40%, respectively.

$$P(X_1 = 0|Y = 1) = 0.4 \implies P(X_1 = 1|Y = 1) = 0.6$$

$$P(X_2 = 0|Y = 1) = 0.4 \implies P(X_2 = 1|Y = 1) = 0.6$$

- If a person does not have the disease, the probabilities of  $T_1$  and  $T_2$  being negative are 60% and 65%, respectively.

$$P(X_1 = 0|Y = 0) = 0.6 \implies P(X_1 = 1|Y = 0) = 0.4$$

$$P(X_2 = 0|Y = 0) = 0.65 \implies P(X_2 = 1|Y = 0) = 0.35$$

- Given a patient with  $X_1 = 0$ , and  $X_2 = 0$ , to estimate

$$P(Y = 0|X_1 = 0, X_2 = 0) \text{ and } P(Y = 1|X_1 = 0, X_2 = 0)$$

# Using a Naïve Bayes Classifier

$$P(Y = 0|X_1 = 0, X_2 = 0)$$

Using Naïve  
Bayes assumption

$$= \frac{P(X_1 = 0, X_2 = 0|Y = 0)P(Y = 0)}{P(X_1 = 0, X_2 = 0)}$$

Bayes rule

$$= \frac{P(X_1 = 0|Y = 0)P(X_2 = 0|Y = 0)P(Y = 0)}{P(X_1 = 0, X_2 = 0)}$$

Prediction:  $Y = 0$

$$= \frac{0.6 \times 0.65 \times 0.5}{P(X_1 = 0, X_2 = 0)} = \frac{0.195}{P(X_1 = 0, X_2 = 0)}$$



$$P(Y = 1|X_1 = 0, X_2 = 0)$$

Using Naïve  
Bayes assumption

$$= \frac{P(X_1 = 0, X_2 = 0|Y = 1)P(Y = 1)}{P(X_1 = 0, X_2 = 0)}$$

Bayes rule

$$= \frac{P(X_1 = 0|Y = 1)P(X_2 = 0|Y = 1)P(Y = 1)}{P(X_1 = 0, X_2 = 0)}$$

# Features are Correlated When $Y = 1$

- However, because  $X_1$  and  $X_2$  are perfectly positively correlated when  $Y = 1$ ,

$$P(X_1 = 0, X_2 = 0 | Y = 1) = P(X_1 = 0 | Y = 1) = 0.4$$

$$\begin{aligned} P(Y = 1 | X_1 = 0, X_2 = 0) &= \frac{P(X_1 = 0, X_2 = 0 | Y = 1)P(Y = 1)}{P(X_1 = 0, X_2 = 0)} \\ \text{Perfectly correlated} &= \frac{P(X_1 = 0 | Y = 1)P(Y = 1)}{P(X_1 = 0, X_2 = 0)} = \frac{0.4 \times 0.5}{P(X_1 = 0, X_2 = 0)} \end{aligned}$$

- $X_1$  and  $X_2$  are independent if the person does not have the disease.

$$\begin{aligned} P(Y = 0 | X_1 = 0, X_2 = 0) &= \frac{P(X_1 = 0 | Y = 0)P(X_2 = 0 | Y = 0)P(Y = 0)}{P(X_1 = 0, X_2 = 0)} \\ &= \frac{0.195}{P(X_1 = 0, X_2 = 0)} \end{aligned}$$

Prediction:  $Y = 1$

# Bayesian Belief Networks

- A more general approach to modeling the independence and conditional independence among  $x$  and  $y$ , s.t. the computation of  $P(x, y) = P(x|y)P(y)$  is tractable.
  - Suppose all features are discrete (if there are both continuous and discrete features, the estimation is much more difficult).
- Representation: a Bayesian network provides a graphical representation of the probabilistic relationships among a set of random variables including features and output class.
- Two key elements:
  - A directed acyclic graph (DAG) encoding the dependence relationships among a set of variables
  - A probability table associating each node to its immediate parent nodes

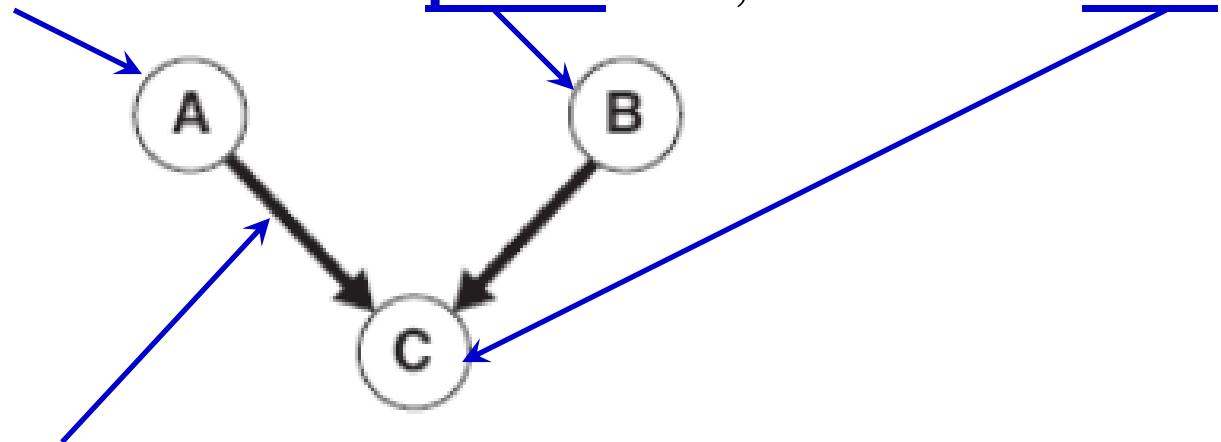
# A DAG Example

- Consider three random variables  $A$ ,  $B$  and  $C$ , where  $A$  and  $B$  are independent variables and each has a direct influence on a third variable,  $C$

$$P(A, B) = P(A)P(B)$$

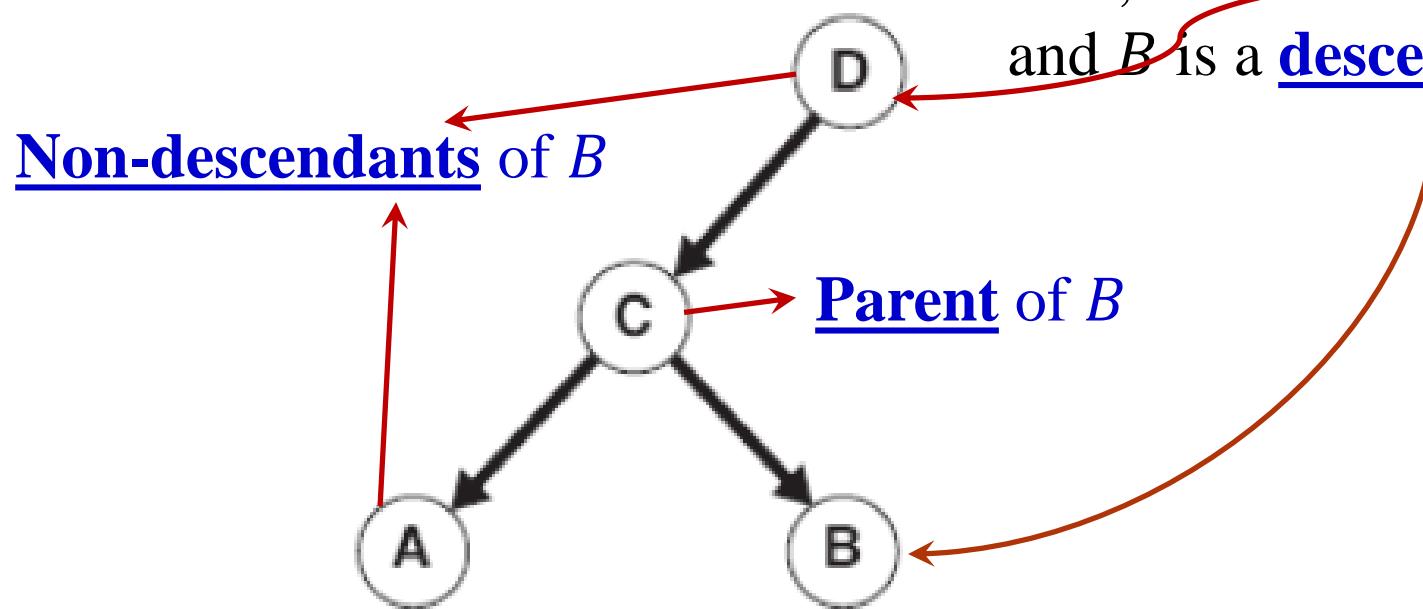
If there is a directed arc from  $B$  to  $C$ , then  $B$  is the **parent** of  $C$ , and  $C$  is the **child** of  $B$

node  $\rightarrow$  variable



Directed arc  $\rightarrow$  dependence relationship ( $C$  depends on  $A$  and  $B$ )

# Another DAG Example

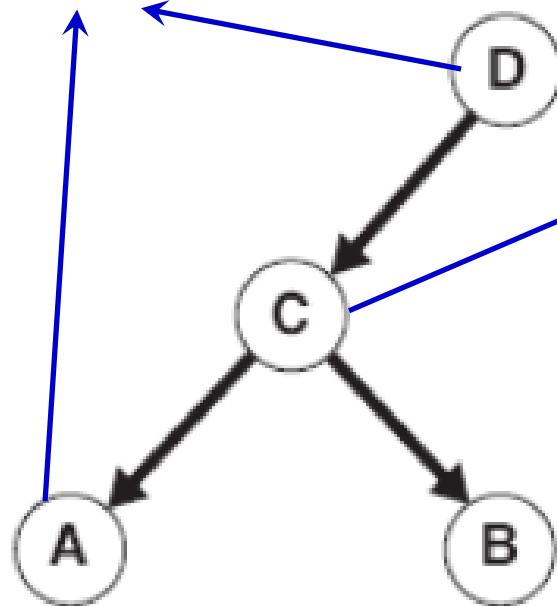


If there is a directed path from *D* to *B*, then *D* is an ancestor of *B*, and *B* is a descendant of *D*

# DAG: Conditional Independence

- Property (conditional independence): a node in a Bayesian network is conditionally independent of its non-descendants, if its parents are known.

Non-descendants of  $B$



Parent of  $B$

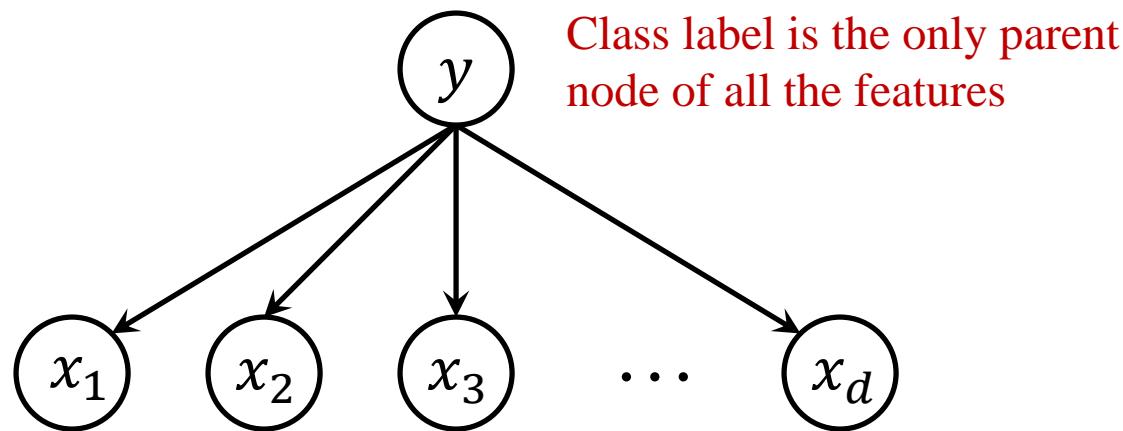
If  $C$  is observed, then  $B$  is conditionally independent of  $A$  and  $D$

$$P(B|C, A, D) = P(B|C)$$

$$P(B, A, D|C) = P(B|C)P(D|C) P(A|C)$$

# A Special Case: Naïve Bayes

- A Naïve Bayes classifier can be represented using a special DAG:



Once a class label is given, all the features are conditionally independent to each other, because each feature is a non-descendant to any other features

$$P(x_1, x_2, \dots, x_d | y = c) = \prod_{i=1}^d P(x_i | y = c)$$

# BBN Representation

- Besides the conditional independence conditions imposed by the network topology, each node is also associated with a probability table.
  - If a node  $X$  does not have any parents, then the table contains only the prior probability  $P(X)$
  - If a node  $X$  has only one parent,  $Z$ , then the table contains the conditional probability  $P(X|Z)$
  - If a node  $X$  has multiple parents,  $\{Z_1, Z_2, \dots, Z_k\}$ , then the table contains the conditional probability  $P(X|Z_1, Z_2, \dots, Z_k)$

# BBN Representation: Example

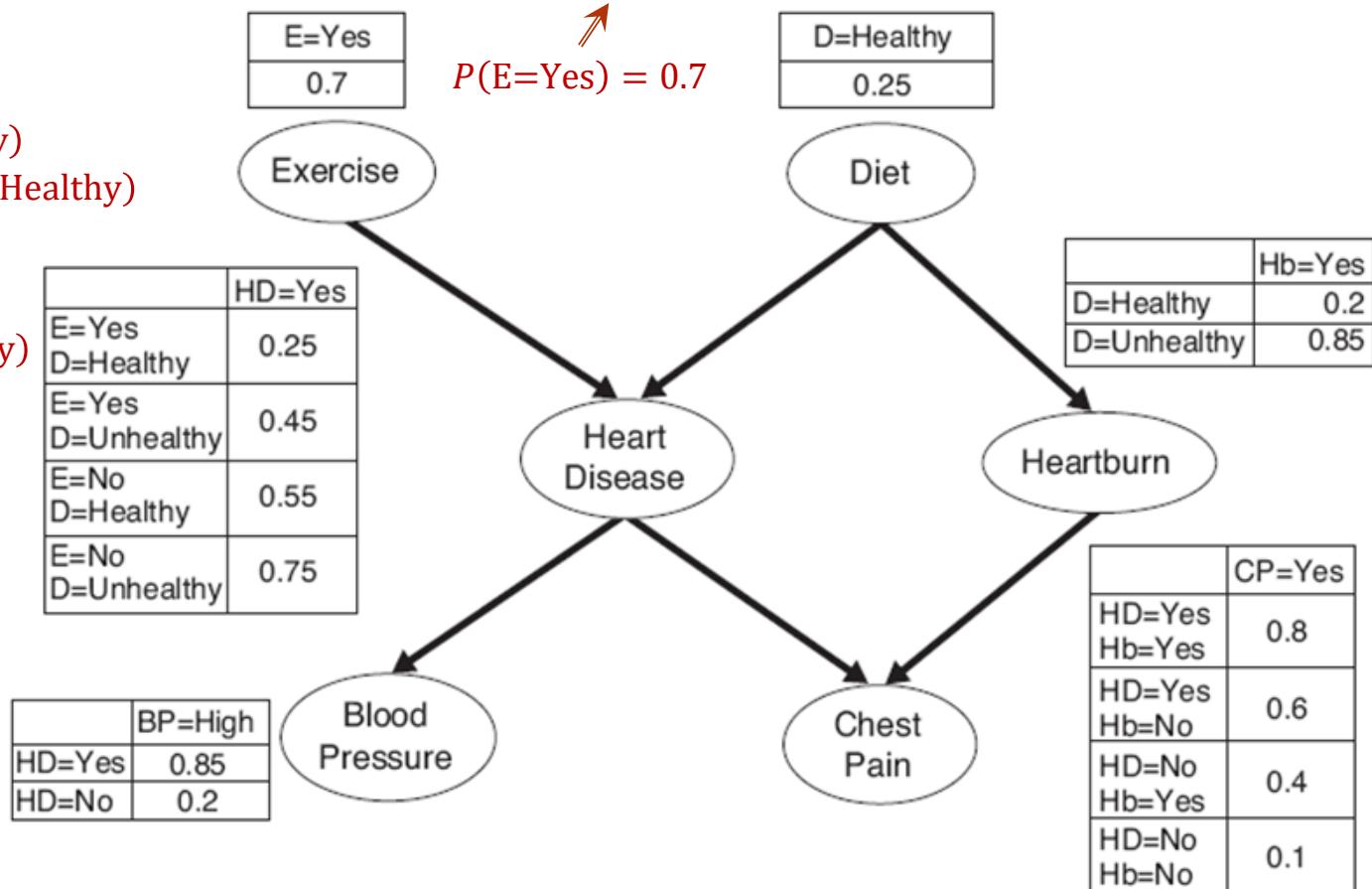
A Bayesian network for modeling patients with heart disease or heartburn problems

$$P(E=\text{No}) = 1 - P(E=\text{Yes}) = 0.3$$

$$P(E=\text{Yes}) = 0.7$$

$$\begin{aligned} P(\text{HD}=\text{No}|E=\text{Yes}, D=\text{Healthy}) \\ = 1 - P(\text{HD}=\text{Yes}|E=\text{Yes}, D=\text{Healthy}) \\ = 0.75 \end{aligned}$$

$$\begin{aligned} P(\text{HD}=\text{Yes}|E=\text{Yes}, D=\text{Healthy}) \\ = 0.25 \end{aligned}$$



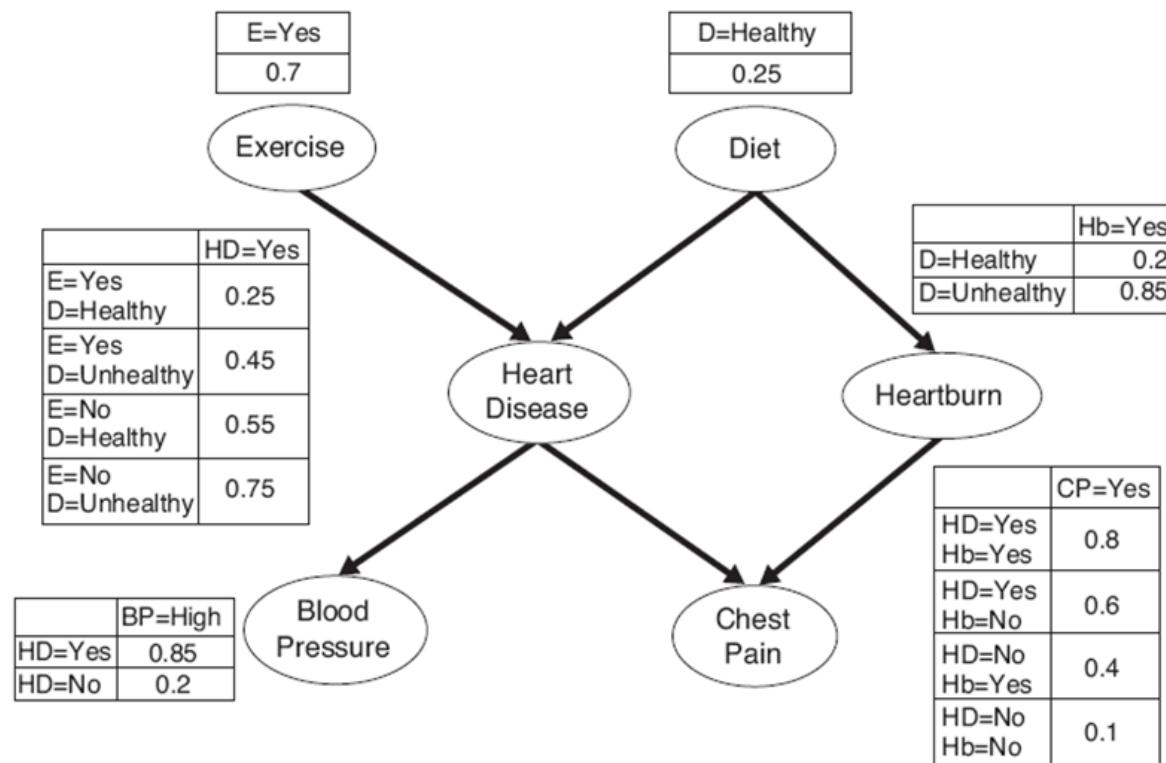
# BBN Model Building

- Two steps in the training phase:
  - Creating the structure of the network, i.e., DAG
    - Network topology can be obtained by encoding the subjective knowledge of domain experts
    - Or can be learned from data (structure learning) --- still an open problem
  - Estimating the probability values in the table associated with each node
    - Counting based on the definition of the corresponding probabilities
- Note: in this module we only focus on how to use a BBN to make predictions (or inference)

Suppose Heart Disease is our target variable to make prediction on (i.e., output), the other variables are input features, whose values can be observed or missing.

# Inference: Example 1

- Without any additional information, to determine whether the person is likely to have heart disease.



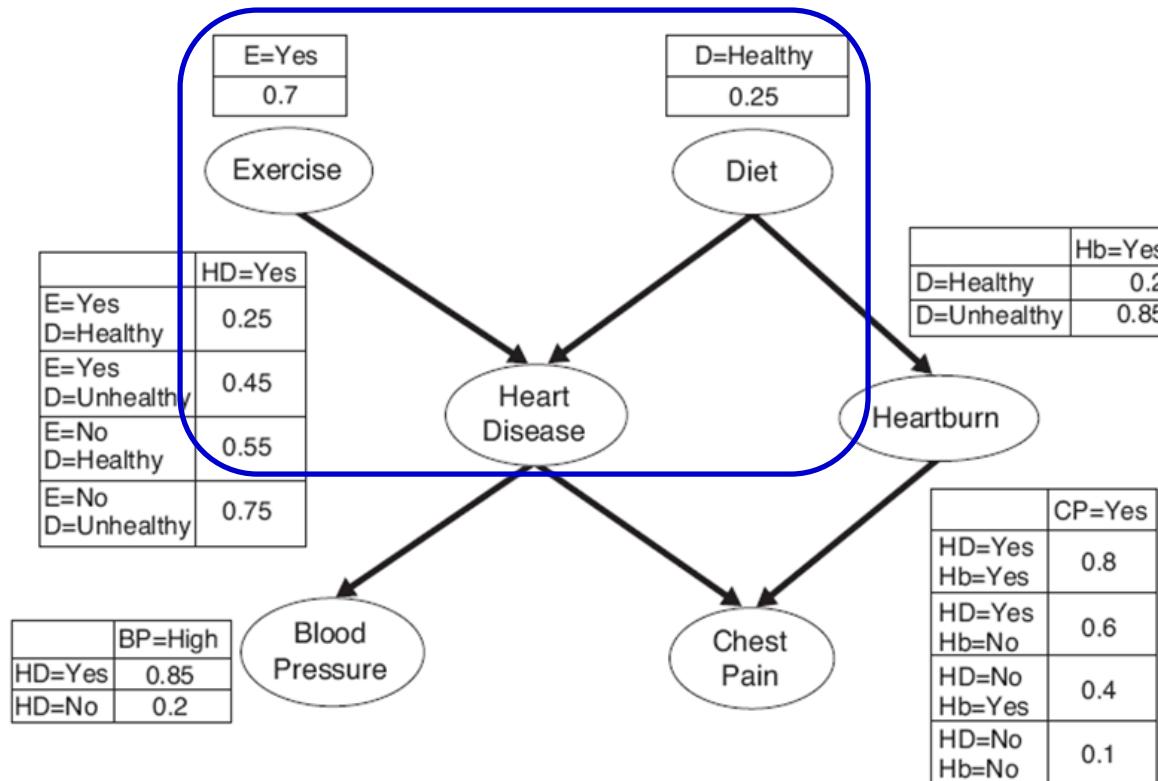
# Inference: Example 1 (cont.)

$P(\text{HD}=\text{Yes}) ?$

$$P(\text{HD}=\text{Yes}) = \sum_{\alpha} \sum_{\beta} P(\text{HD}=\text{Yes}, E=\alpha, D=\beta)$$

Sum Rule

$$\alpha = \{\text{Yes}, \text{No}\} \quad \beta = \{\text{Healthy}, \text{Unhealthy}\}$$



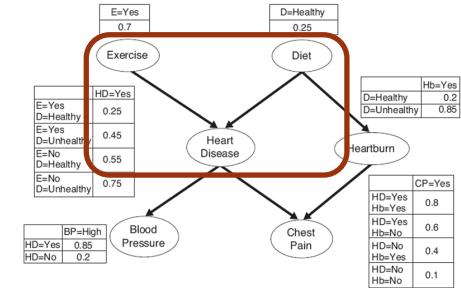
# Inference: Example 1 (cont.)

$$P(\text{HD=Yes}) = \sum_{\alpha} \sum_{\beta} P(\text{HD=Yes}, E=\alpha, D=\beta)$$

Product Rule

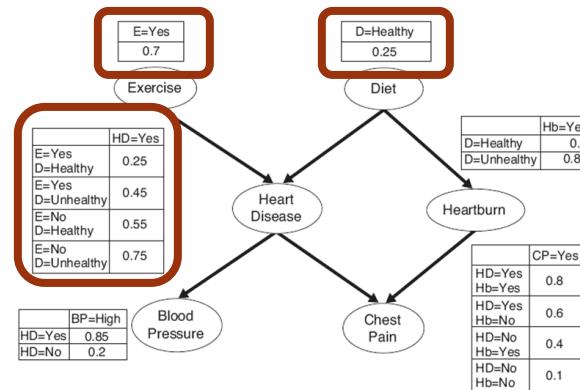
$$\begin{aligned}
 &= \sum_{\alpha} \sum_{\beta} P(\text{HD=Yes}|E=\alpha, D=\beta) P(E=\alpha, D=\beta) \\
 &= \sum_{\alpha} \sum_{\beta} P(\text{HD=Yes}|E=\alpha, D=\beta) P(E=\alpha) P(D=\beta)
 \end{aligned}$$

Independence



$$\alpha = \{\text{Yes}, \text{No}\} \quad \beta = \{\text{Healthy}, \text{Unhealthy}\}$$

$$\begin{aligned}
 P(\text{HD=Yes}) &= \sum_{\alpha} \sum_{\beta} P(\text{HD=Yes}|\text{E}=\alpha, \text{D}=\beta)P(\text{E}=\alpha)P(\text{D}=\beta) \\
 &= P(\text{HD=Yes}|\text{E=Yes}, \text{D=Healthy})P(\text{E=Yes})P(\text{D=Healthy}) \\
 &\quad + P(\text{HD=Yes}|\text{E=Yes}, \text{D=Unhealthy})P(\text{E=Yes})P(\text{D=Unhealthy}) \\
 &\quad + P(\text{HD=Yes}|\text{E=No}, \text{D=Healthy})P(\text{E=No})P(\text{D=Healthy}) \\
 &\quad + P(\text{HD=Yes}|\text{E=No}, \text{D=Unhealthy})P(\text{E=No})P(\text{D=Unhealthy})
 \end{aligned}$$



Look up  
probability tables

$$\begin{aligned}
 &= 0.25 \times 0.7 \times 0.25 + 0.45 \times 0.7 \times 0.75 + 0.55 \times 0.3 \times 0.25 + 0.75 \times 0.3 \times 0.75 \\
 &= 0.49
 \end{aligned}$$

# Inference: Example 1 (cont.)

$$P(\text{HD}=\text{Yes}) = 0.49$$

$$P(\text{HD}=\text{No}) = 1 - P(\text{HD}=\text{Yes}) = 0.51$$

- Therefore, the person has a slightly higher chance of not getting the heart disease.

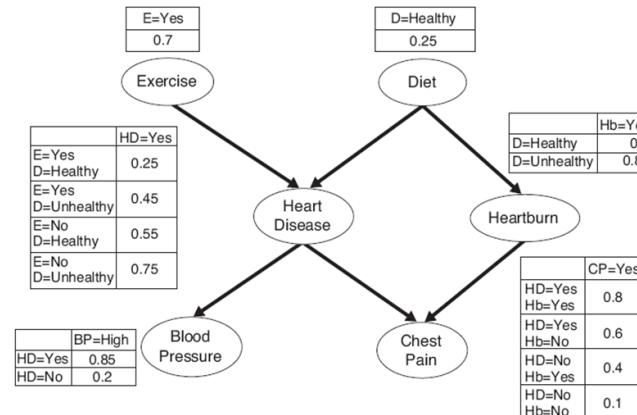
# Inference: Example 2

- If the person has high blood pressure, chest pain and heartburn, but does regular exercise and eats a healthy diet, to diagnose whether the patient has heart disease:

$$P(\text{HD=Yes} | \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

*vs.*

$$P(\text{HD=No} | \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$



$$P(HD=Yes|BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes)$$

$$P(HD=Yes, BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes)$$

$$P(BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes)$$



How to estimate the joint probability in the numerator?

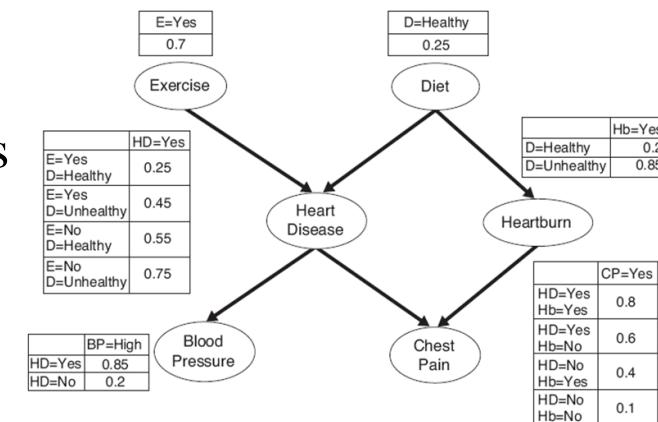
Keep in mind that the goal is to rewrite the joint probability into an equivalent form, such that for all the probabilities in the rewritten equivalent form, their values can be found from the tables in the BBN.

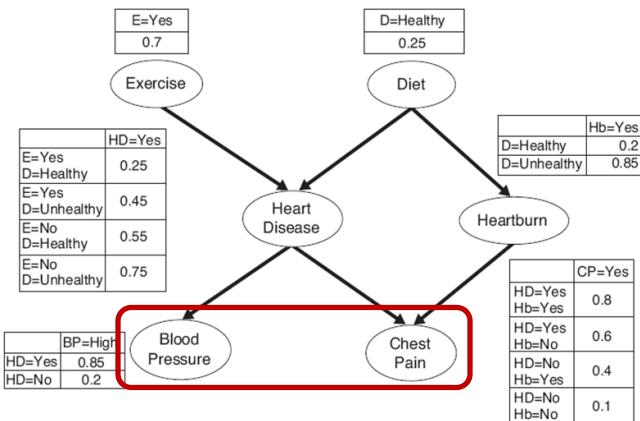
There are many ways to rewrite the above joint probability, e.g.,

$$P(D, E | HD, BP, Hb, CP)P(HD, BP, Hb, CP), \text{ or}$$

$$P(D, BP, CP | E, Hb, HD)P(E, Hb, HD), \text{ many others}$$

Which one is useful for the joint probability simplification?





From the BBN, we found that the variables BP and CP are child nodes of HD and Hb, but not a parent node for any other variables. That means we could not find any conditional probabilities in the tables, which involves BP and CP in the condition, like  $P(D, E | HD, BP, Hb, CP)$ . In other words, if in the rewritten equivalent form, there is a probability where BP or (and) CP is (are) in the condition, we still have to further rewrite it such that BP and CP are not in the condition of any conditional probability term.

The above analysis motivates us to transform the joint probability (numerator) using the following form based on the product rule:

$$P(HD=Yes, BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes)$$

$$= P(BP=High, CP=Yes | HD=Yes, Hb=Yes, E=Yes, D=Healthy) P(HD=Yes, Hb=Yes, E=Yes, D=Healthy)$$

Not in the condition of the conditional probability

Denote by  $\mathbf{U} = \{BP, CP\}$  and  $\mathbf{V} = \{HD, Hb, E, D\}$

$$P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U}|\mathbf{V})P(\mathbf{V})$$

$$P(\text{HD=Yes} | \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

$$\frac{P(\text{HD=Yes}, \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}$$

$$P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

Denote by  $\mathbf{U} = \{\text{BP}, \text{CP}\}$  and  $\mathbf{V} = \{\text{HD}, \text{Hb}, \text{E}, \text{D}\}$

$$P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U} | \mathbf{V})P(\mathbf{V})$$

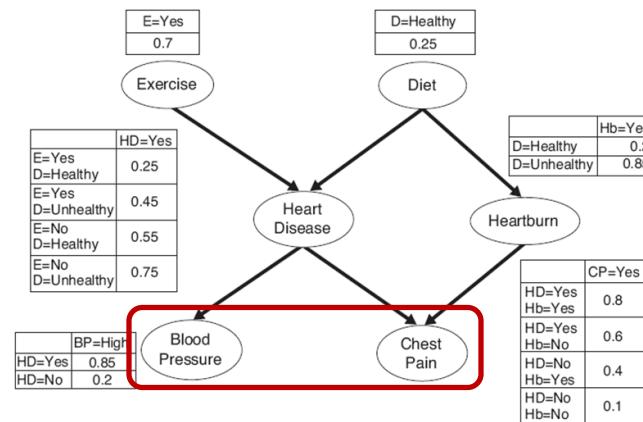
1

2

$$P(\text{BP=High}, \text{CP=Yes} | \text{HD=Yes}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy}) / P(\text{HD=Yes}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy})$$

Slide 29

Slide 30



Recall: if **A** and **B** are conditionally independent given **C**, we have

$$P(A|B, C) = P(A|C) \text{ or } P(A, B|C) = P(A|C)P(B|C)$$

1

$P(\text{BP}=\text{High}, \text{CP}=\text{Yes} | \boxed{\text{HD}=\text{Yes}, \text{Hb}=\text{Yes}}, \text{E}=\text{Yes}, \text{D}=\text{Healthy})$

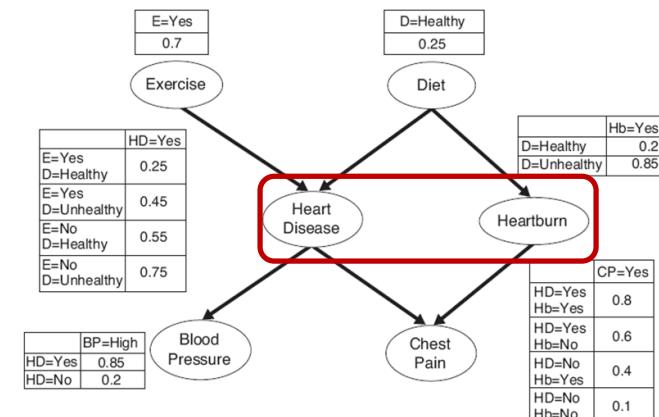
- HD and Hb are parents of BP and CP
- E and D are non-descendants of BP and CP
- Therefore, BP and CP are conditionally independent of E and D given HD and Hb

$\rightarrow P(\text{BP}=\text{High}, \text{CP}=\text{Yes} | \text{HD}=\text{Yes}, \text{Hb}=\text{Yes})$

- HD and Hb are parents of CP, and BP is a non-descendant of CP
- Therefore, BP and CP are conditionally independent given HD and Hb

$\rightarrow \boxed{P(\text{BP}=\text{High} | \text{HD}=\text{Yes}, \text{Hb}=\text{Yes})} P(\text{CP}=\text{Yes} | \text{HD}=\text{Yes}, \text{Hb}=\text{Yes})$

- HD is parent of BP
- Hb is a non-descendant of BP
- Therefore, BP and Hb are conditionally independent given HD



$P(\text{BP}=\text{High} | \text{HD}=\text{Yes}) P(\text{CP}=\text{Yes} | \text{HD}=\text{Yes}, \text{Hb}=\text{Yes})$

$$= 0.85 \times 0.8 = 0.68$$

$$P(HD=Yes, Hb=Yes, E=Yes, D=Healthy)$$

Denote by  $\mathbf{U} = \{HD, Hb\}$  and  $\mathbf{V} = \{E, D\}$ ,  $P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U}|\mathbf{V})P(\mathbf{V})$

$$= P(HD=Yes, Hb=Yes | E=Yes, D=Healthy) P(E=Yes, D=Healthy)$$

Independence

Given E and D, HD and Hb are conditionally independent

$$P(HD=Yes | E=Yes, D=Healthy) P(Hb=Yes | E=Yes, D=Healthy)$$

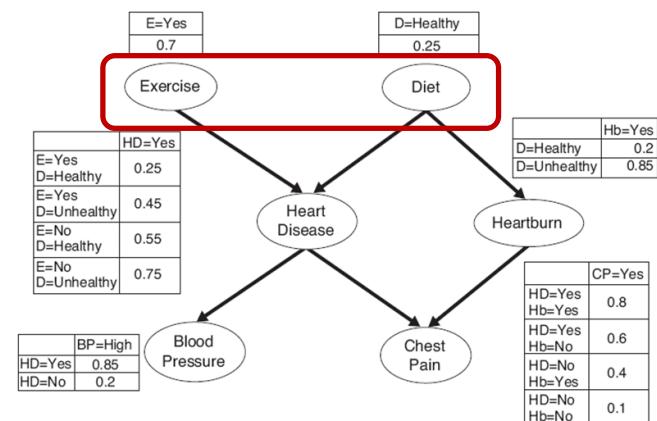
Given D, Hb and E are conditionally independent

$$P(HD=Yes | E=Yes, D=Healthy) P(Hb=Yes | D=Healthy)$$



$$= P(HD=Yes | E=Yes, D=Healthy) P(Hb=Yes | D=Healthy) P(E=Yes) P(D=Healthy)$$

$$= 0.25 \times 0.2 \times 0.7 \times 0.25 = 0.00875$$



$$P(\text{HD=Yes} \mid \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

$$= \frac{P(\text{BP=High}, \text{CP=Yes} \mid \text{HD=Yes}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy}) P(\text{HD=Yes}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy})}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}$$

$$= \frac{0.68 \times 0.00875}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})} = \frac{0.00595}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}$$

$$P(\text{HD=No} \mid \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

$$= \frac{P(\text{BP=High}, \text{CP=Yes} \mid \text{HD=No}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy}) P(\text{HD=No}, \text{Hb=Yes}, \text{E=Yes}, \text{D=Healthy})}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}$$

$$= \frac{0.08 \times 0.02625}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})} = \frac{0.0021}{P(\text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})}$$

$$P(\text{HD=Yes} \mid \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

>

$$P(\text{HD=No} \mid \text{BP=High}, \text{Hb=Yes}, \text{CP=Yes}, \text{D=Healthy}, \text{E=Yes})$$

The person has a higher chance of getting the heart disease.

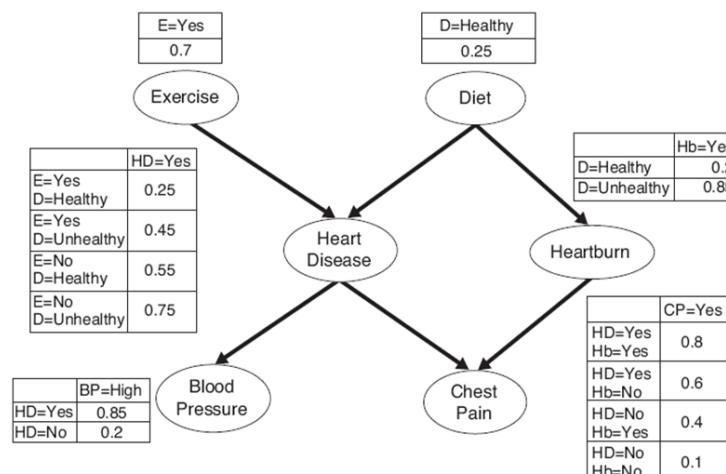
# Inference: Example 3

- If the person has high blood pressure, but exercises regularly and eats a healthy diet, to diagnose about heart disease (estimate the probabilities).

$$P(\text{HD}=\text{Yes} | \text{BP}=\text{High}, \text{D}=\text{Healthy}, \text{E}=\text{Yes})$$

vs.

$$P(\text{HD}=\text{No} | \text{BP}=\text{High}, \text{D}=\text{Healthy}, \text{E}=\text{Yes})$$



Tutorial



# Hint: Using BBNs for Inference

- Given a BBN, and an inference (prediction) task:
  1. Translate the problem into a probabilistic language, i.e., what probabilities to be estimated?
  2. If the probabilities to be estimated cannot be obtained from the probability tables of the BBN, then
    - A. Identify a subgraph which captures the dependence between input variables (features) and the output variable (class)
    - B. Based on the network topology, apply product rule, sum rule and the properties of conditional independence and independence to induce equivalent forms of the probabilities until all probabilities can be found from the probability tables

# Bayesian Belief Networks: Summary

- BBNs use a (directed) graphical model to model dependence among variables
  - Other directed graphical models: Hidden Markov Models, Dynamic Bayesian Networks, etc.
  - Undirected graphical models: e.g., Markov Random Fields, Conditional Random Fields, etc.
- Network structure construction is difficult
  - Use domain knowledge – not complete, may not accurate
  - Learn structure from data – computationally expensive, greedy algorithm → not optimal

**Thank you!**

# **SC4000/CZ4041/CE4041:**

# **Machine Learning**

## **Lesson 5: Decision Tree**

Kelly KE

College of Computing and Data Science  
NTU, Singapore

Acknowledgements: slides are adapted from the lecture notes of the books “Introduction to Machine Learning” (Chap. 9) and “Introduction to Data Mining” (Chap. 4).

# An Illustrative Example

- Consider the problem of predicting whether  
→ a loan applicant will repay his/her loan obligation  
(no cheat) or become delinquent (cheat).

Example

Predefined categories



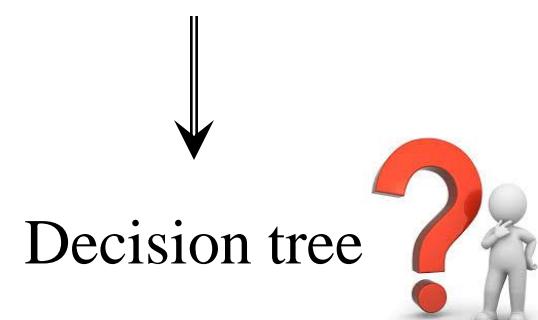
# An Illustrative Example (cont.)

- Training set: constructed by examining the records of previous borrowers.

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

Bayesian classifiers:  $P(y|x)$

$$\xrightarrow{\hspace{1cm}} f(x) = y$$



# Motivation of Decision Trees

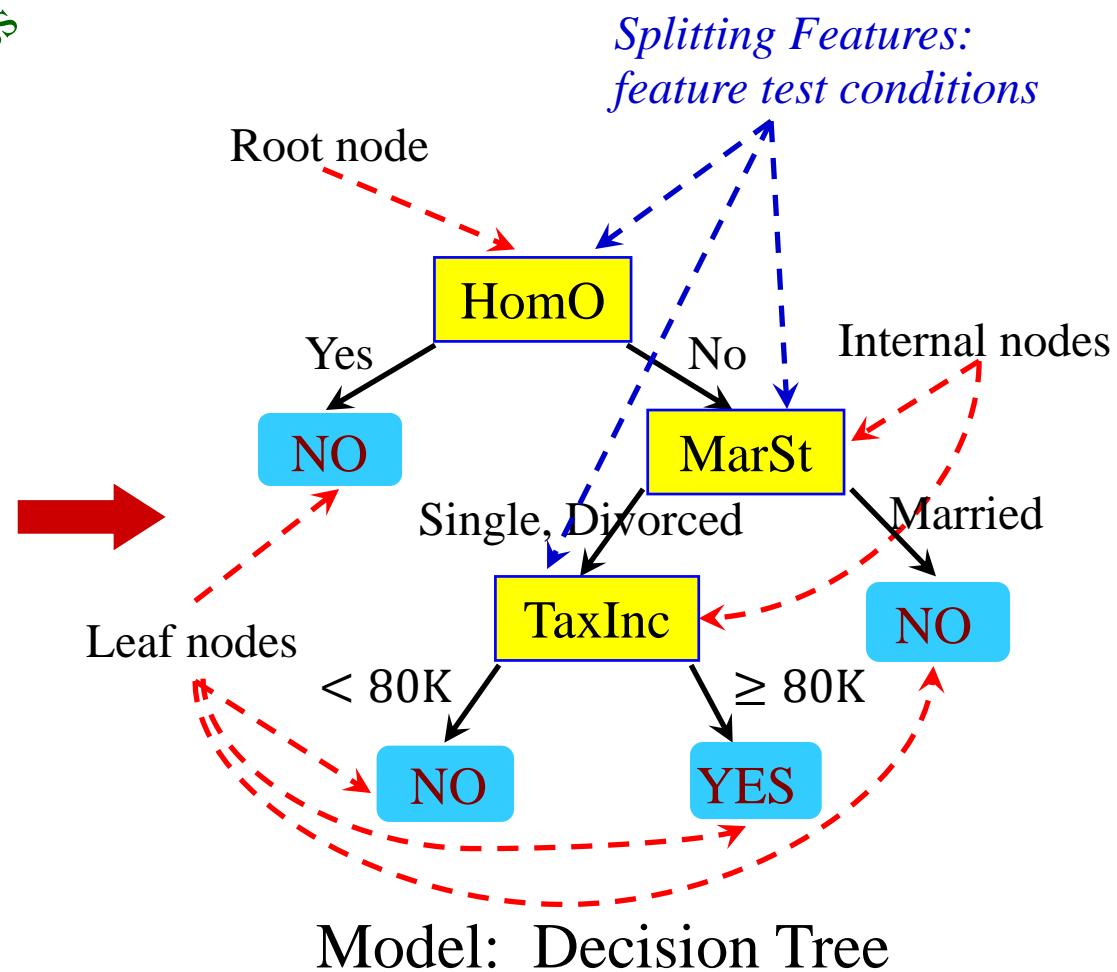
- Suppose a new applicant submits a loan application. How do we decide whether to approve or reject the application?

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?

- To pose a series of questions about the profile of the applicant:
  - Whether the applicant is a home owner? If yes, then he/she may repay the loan obligation with a high probability.
  - After that, we may ask a follow-up question: what is the applicant's marital status? ...

# Example of a Decision Tree

Tid	binary discrete continuous class				Cheat
	Home Owner	Marital Status	Taxable Income		
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	



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

Induction  
(induce a tree)

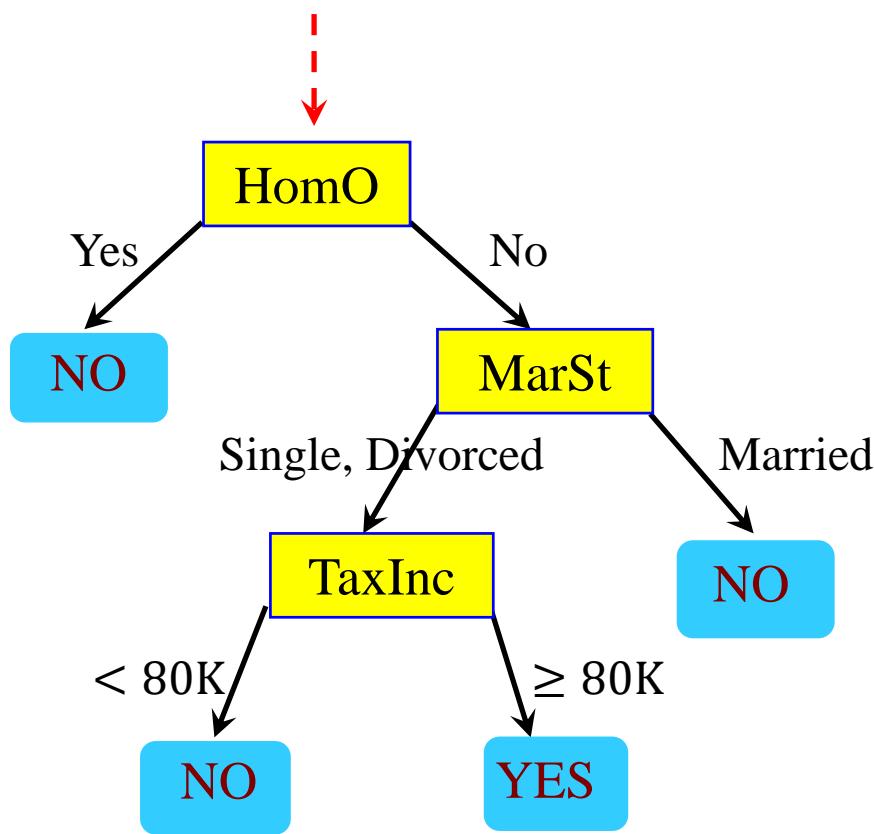
Decision Tree

<i>Tid</i>	Home Owner	Marital Status	Taxable Income	Cheat
11	No	Single	55K	?
12	Yes	Divorce	80K	?
13	Yes	Married	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

Deduction  
(apply the tree)

# Apply Decision Tree to Test Data

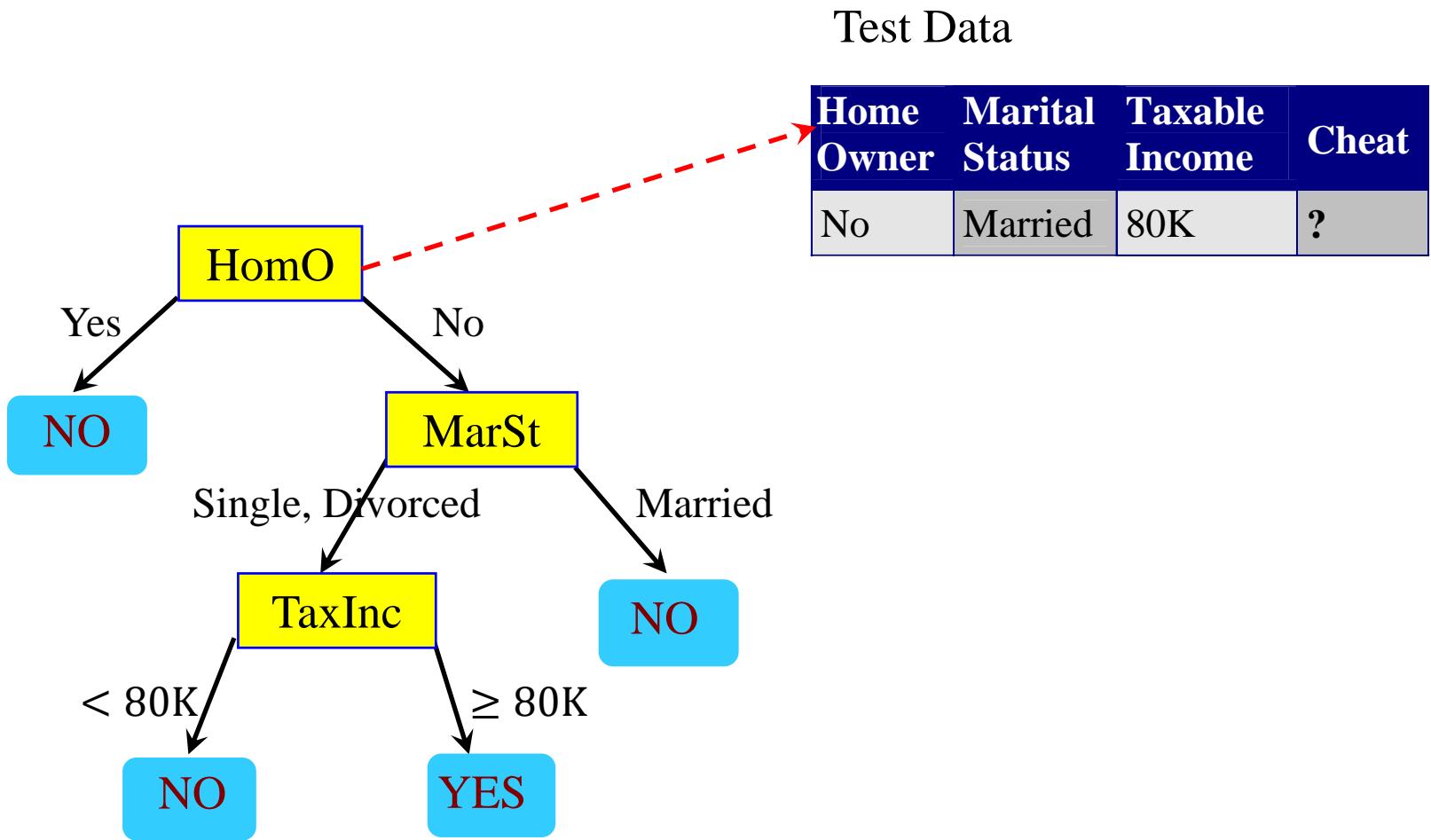
Start from the root of tree.



Test Data

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?

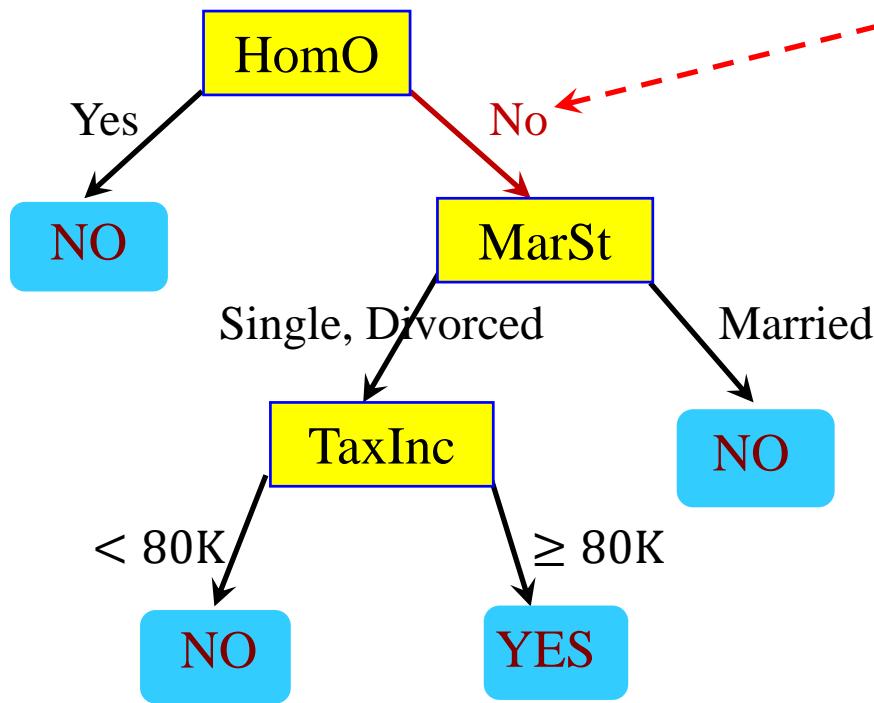
# Apply Model to Test Data



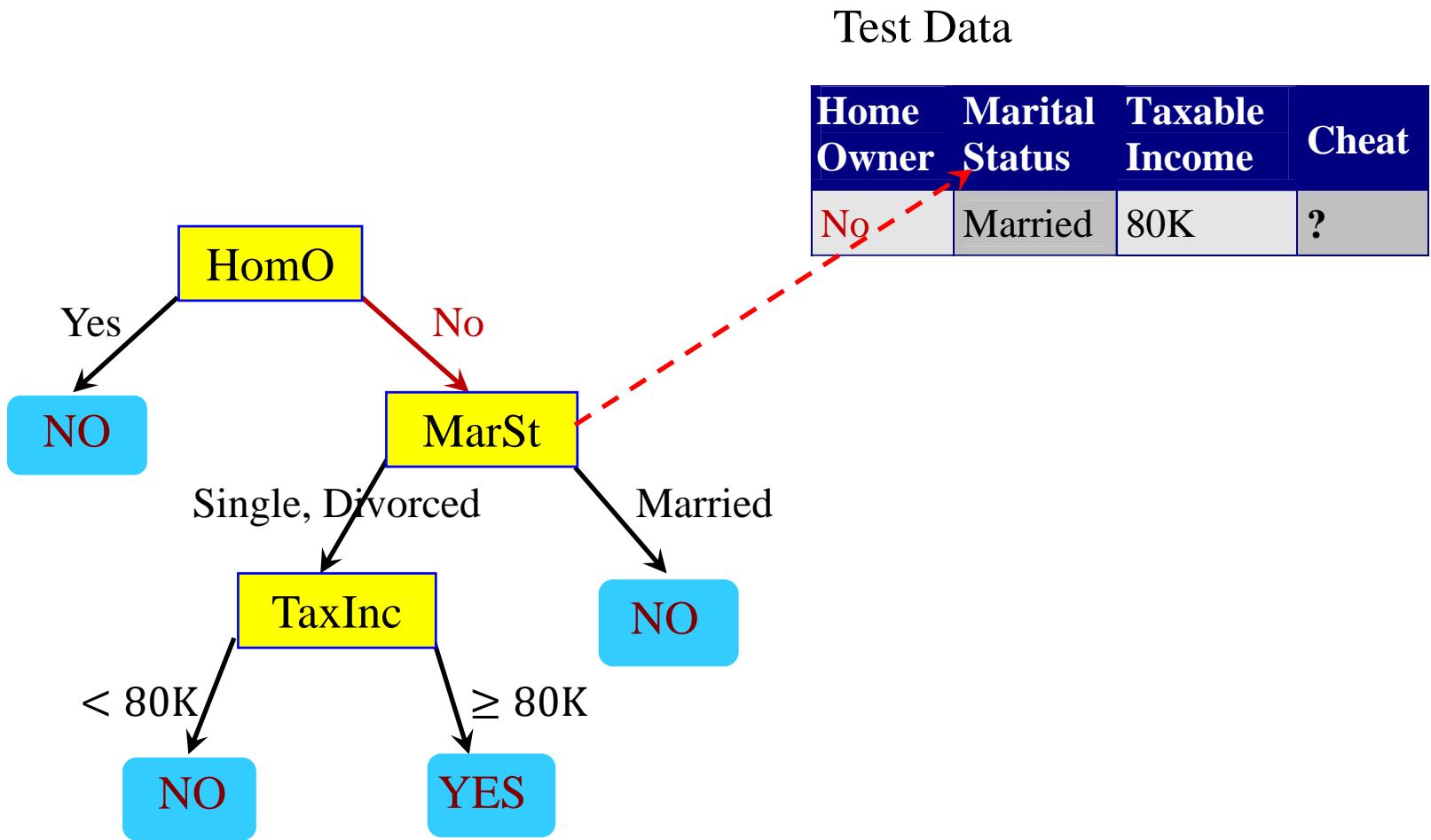
# Apply Model to Test Data

Test Data

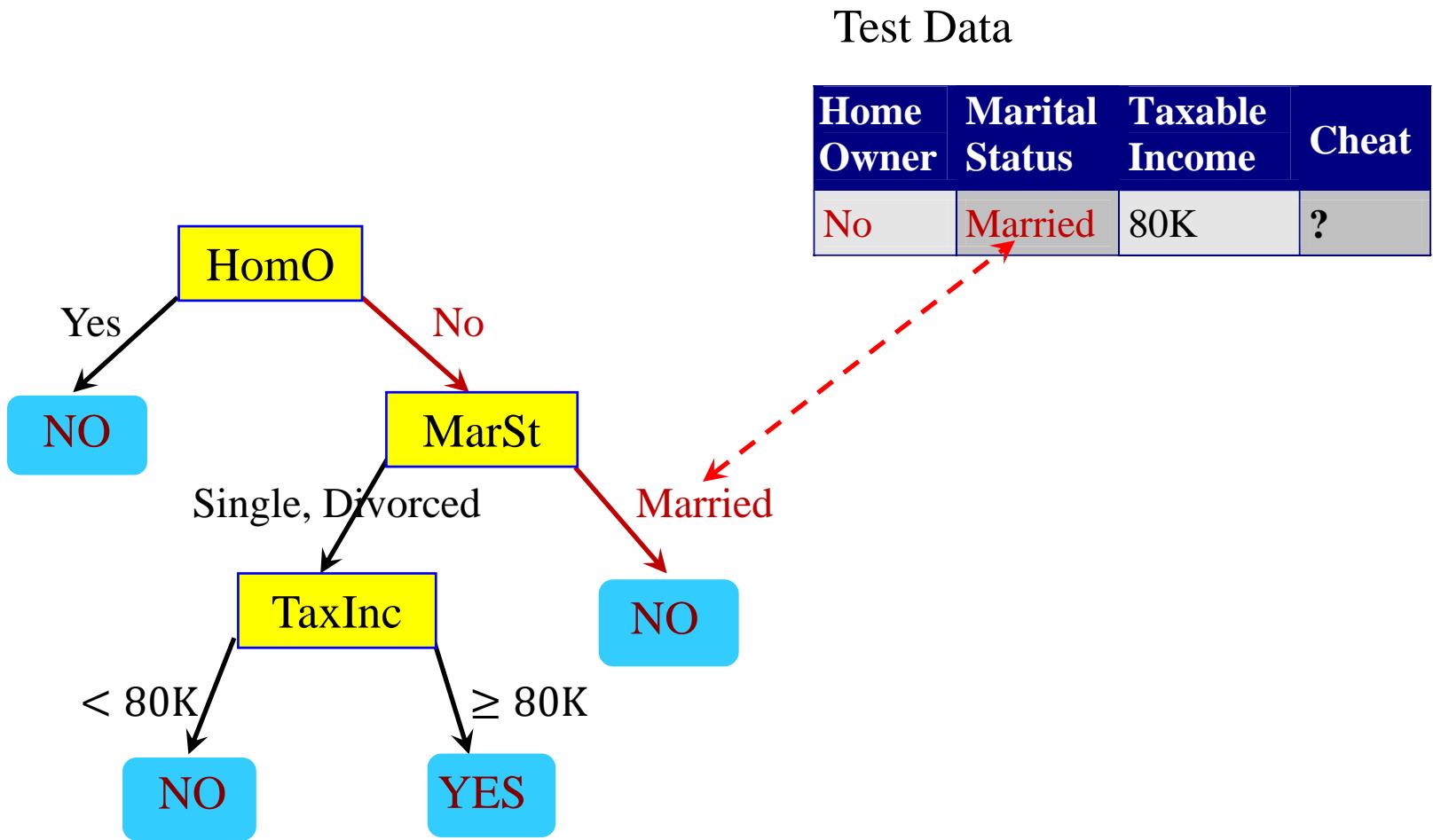
Home Owner	Marital Status	Taxable Income	Cheat
No	Married	80K	?



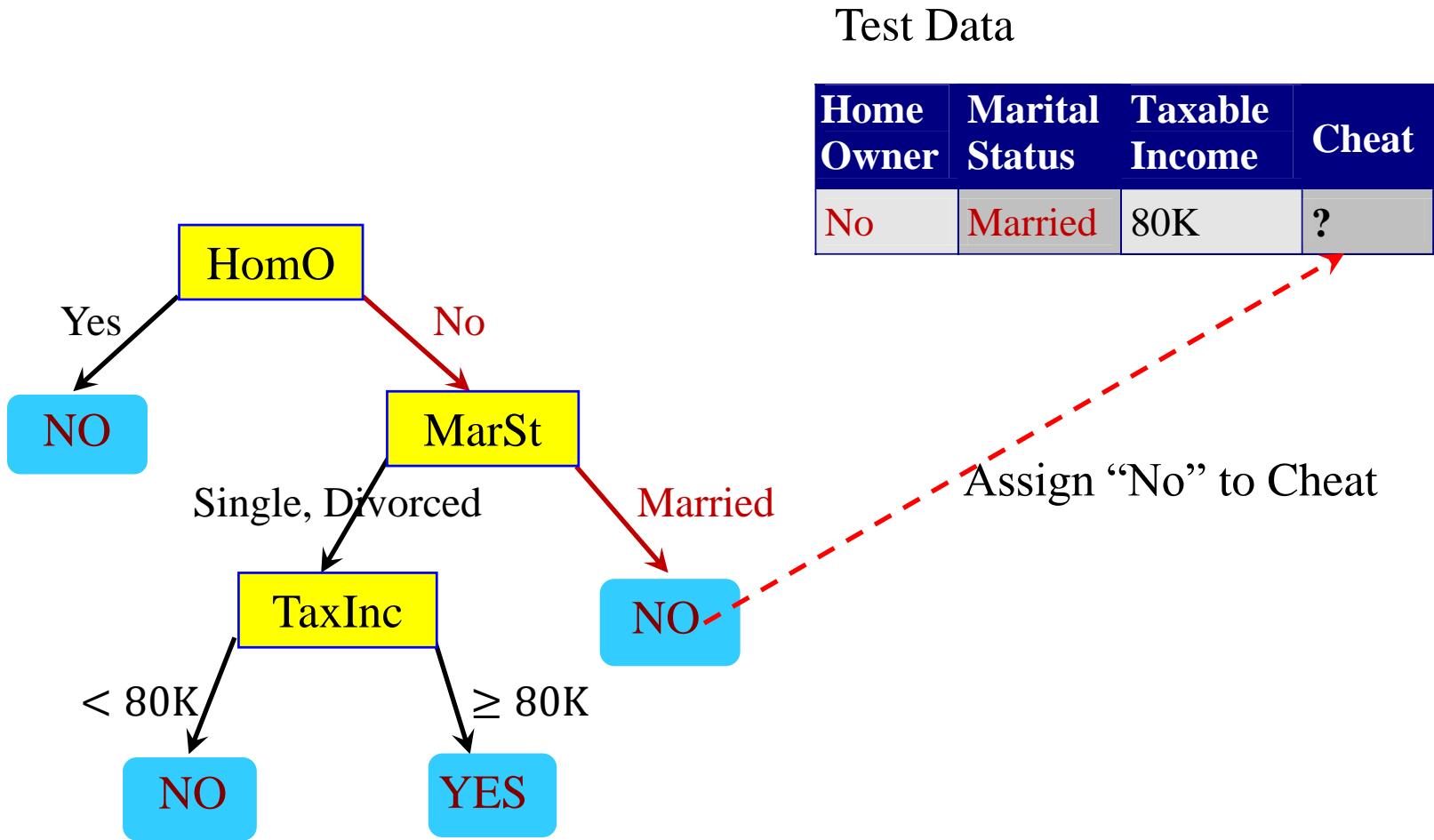
# Apply Model to Test Data



# Apply Model to Test Data



# Apply Model to Test Data



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

Tree induction algorithm

Induction  
(induce a tree)

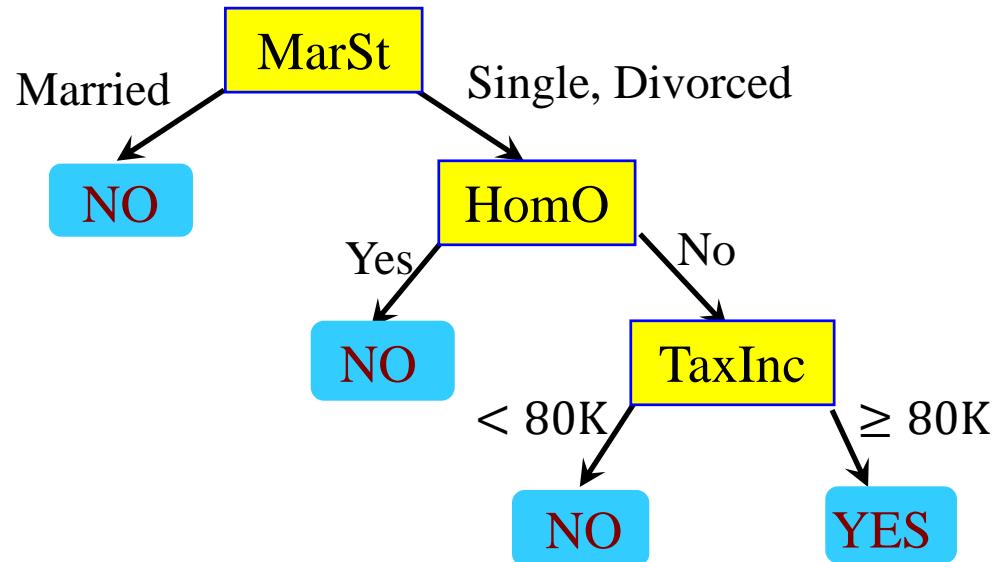
Decision  
Tree

<i>Tid</i>	Home Owner	Marital Status	Taxable Income	Cheat
11	No	Single	55K	?
12	Yes	Divorce	80K	?
13	Yes	Married	110K	?
14	No	Single	95K	?
15	No	Married	67K	?

Deduction  
(apply the tree)

# Another Example of Decision Tree

Tid	Home Owner	Marital Status	Taxable Income	class	
				binary	discrete
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	

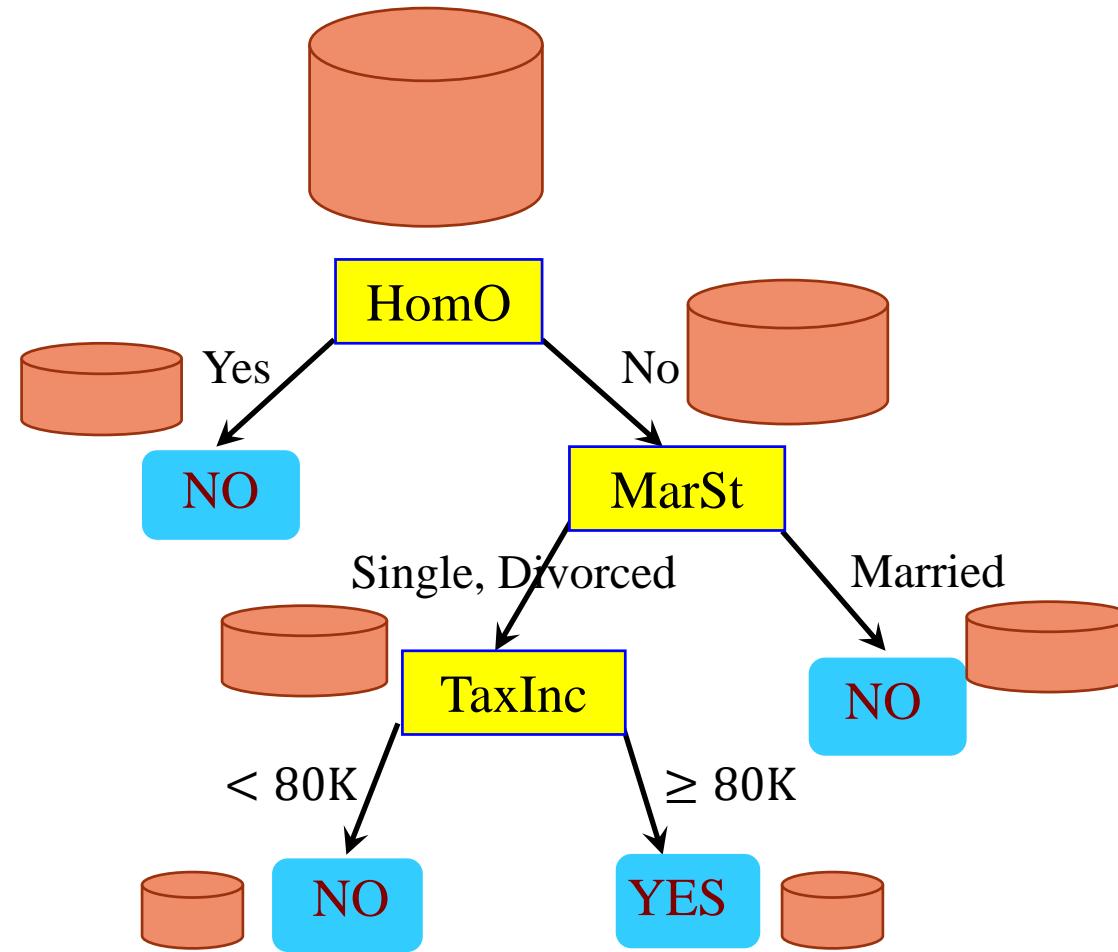


There could be more than one tree that fits the same data!

# Decision Tree Induction Algorithms

- CART, ID3, C4.5, etc.
- High-level basic idea:
  - Given a new application, the goal of asking a series of questions (checking properties of the profile) one by one is to find similar profiles (applicants) in the past until it is confident to make a decision based on the labels (whether cheat or no cheat) of the similar applicants.
  - A tree can be learned by splitting training data into subsets based on outcomes of a feature test.
  - This process is recursively applied on each derived subset until the subset at a node has all the same class or there is no improvement for prediction.

# Illustration



# High-level Algorithm

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

Initial tree

Don't Cheat

Home Owner test condition is selected

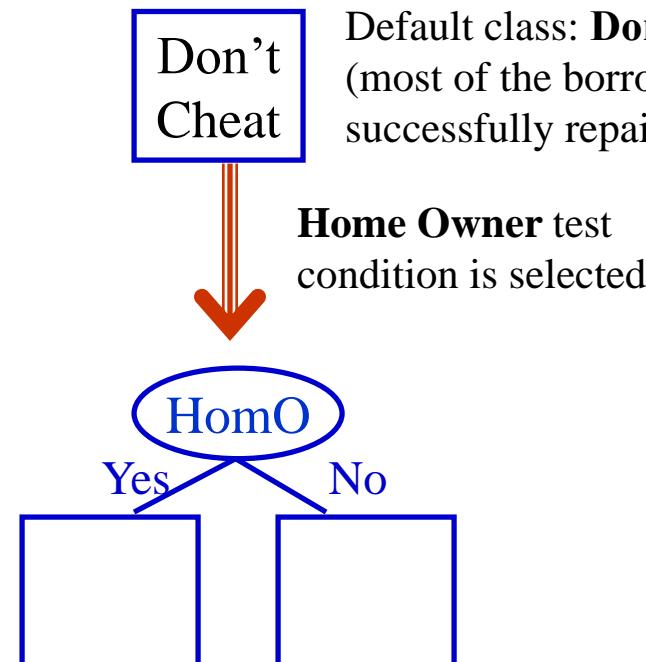
Default class: **Don't Cheat**  
(most of the borrowers successfully repaid their loans )

HomO

# High-level Algorithm (cont.)

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

Initial tree

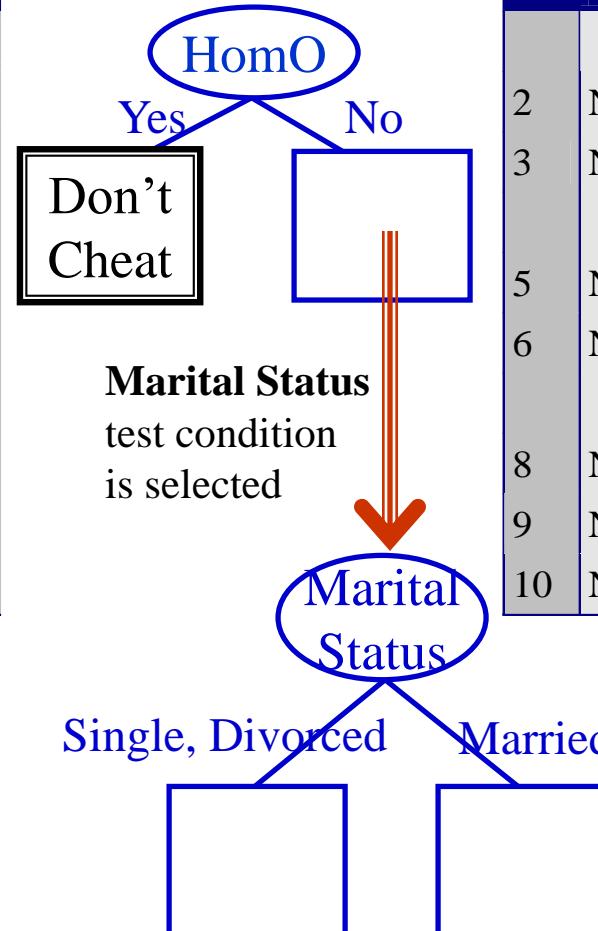


Default class: **Don't Cheat**  
 (most of the borrowers successfully repaid their loans )

**Home Owner** test condition is selected

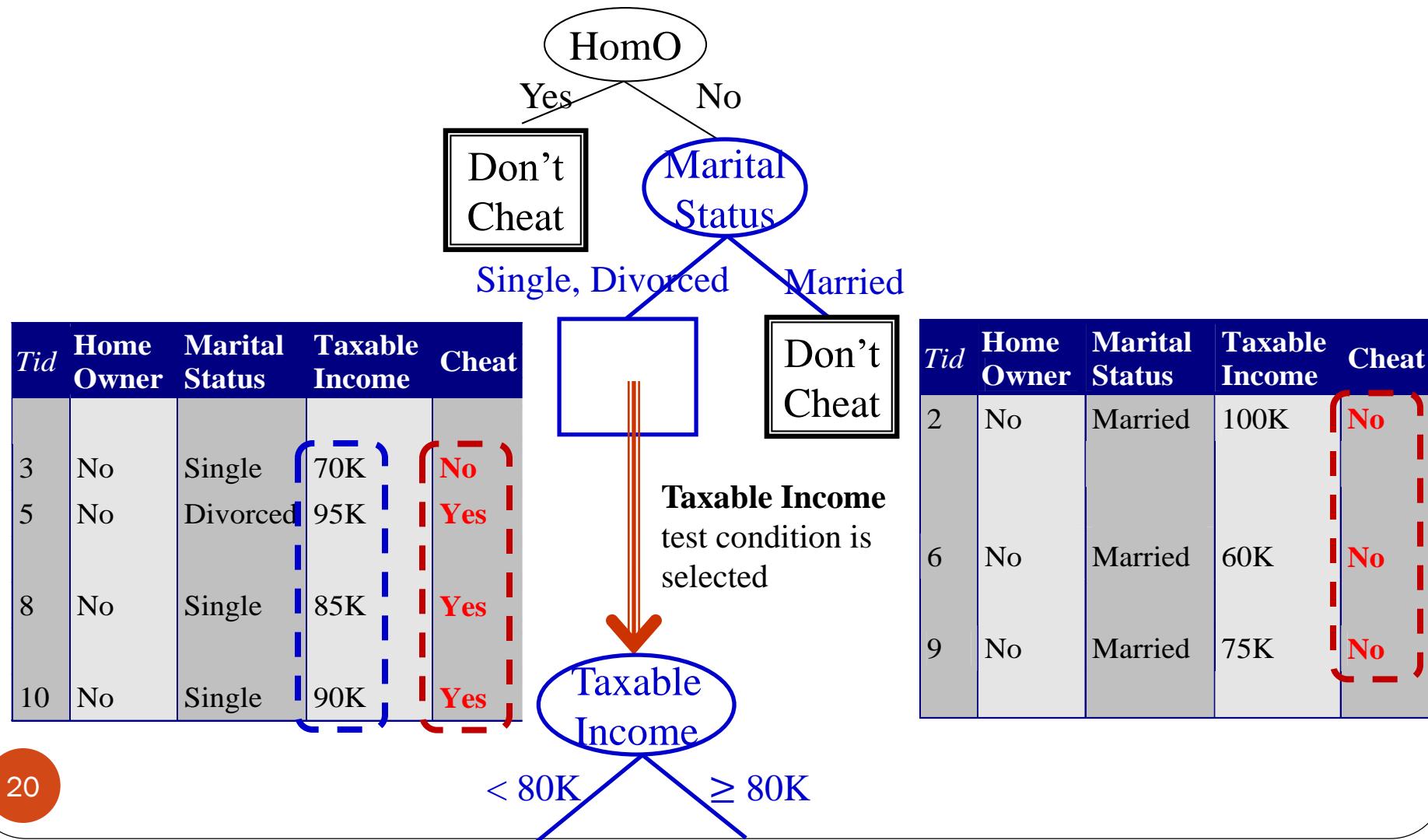
# High-level Algorithm (cont.)

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No

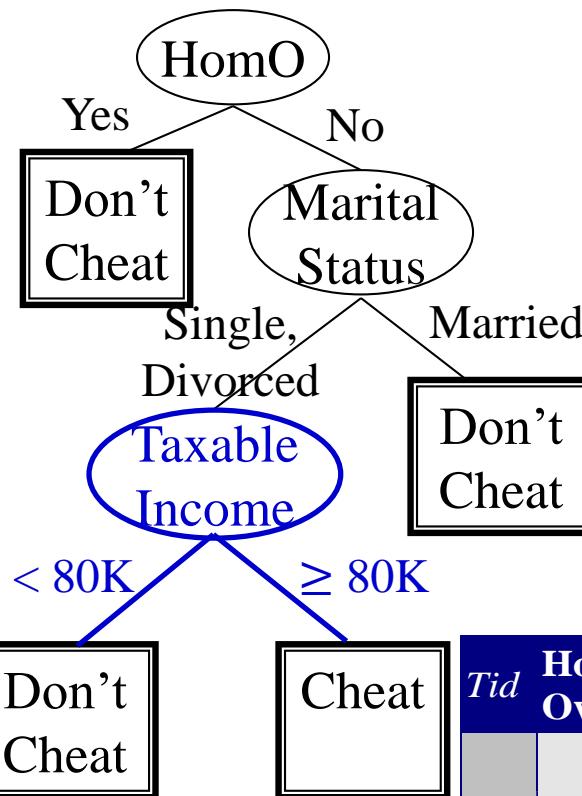


Tid	Home Owner	Marital Status	Taxable Income	Cheat
2	No	Married	100K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# High-level Algorithm (cont.)



# High-level Algorithm (cont.)

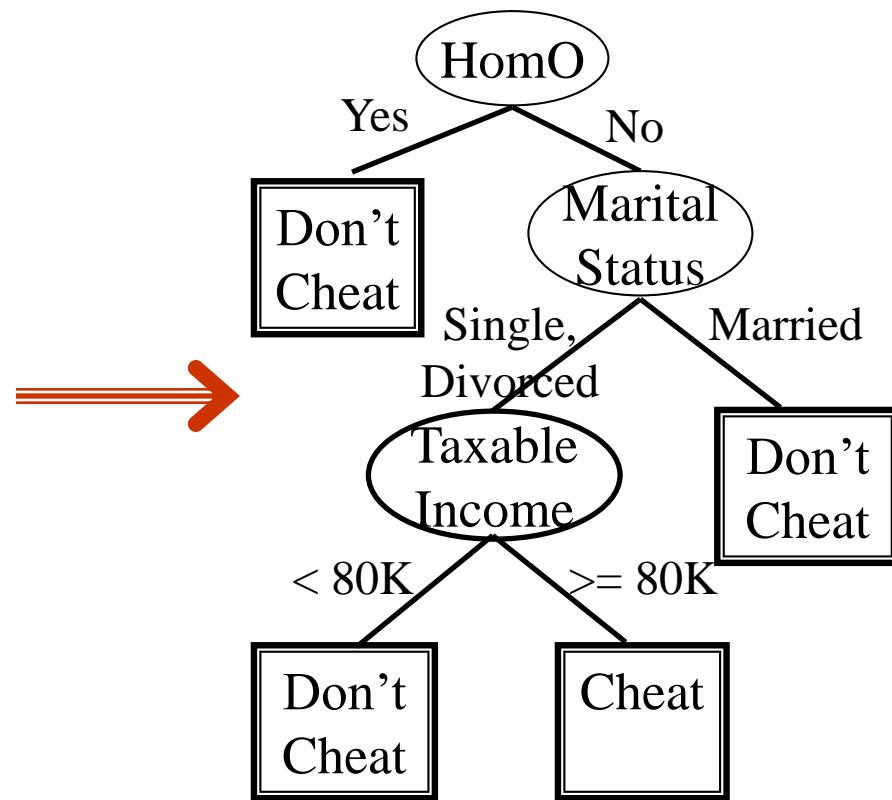


Tid	Home Owner	Marital Status	Taxable Income	Cheat
3	No	Single	70K	No

Tid	Home Owner	Marital Status	Taxable Income	Cheat
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

# High-level Algorithm (cont.)

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



# High-level Algorithm: Summary

- Let  $D_t$  be the set of training records that reach a node  $t$
- General Procedure:
  - If  $D_t$  contains records that belong to the same class  $y_t$ , then  $t$  is a leaf node labeled as  $y_t$  (e.g., Yes or No)
  - Otherwise if  $D_t$  is an empty set, then  $t$  is a leaf node labeled by the default class,  $y_d$  (e.g., No)
  - Otherwise if  $D_t$  contains records that belong to more than one class, then a feature is selected to conduct condition test to split the data into smaller subsets.
    - A child node is created for each outcome of the test condition and the records in  $D_t$  are distributed to the children based on the outcomes
    - Recursively apply the procedure to each subset

# Tree Induction

- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the feature test condition?
    - How to determine the best split?
  - Determine when to stop splitting

# Tree Induction

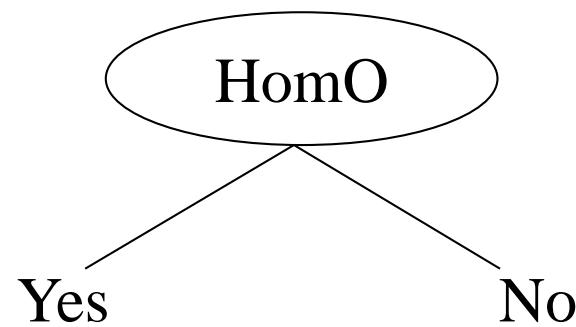
- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the feature test condition?
    - How to determine the best split?
  - Determine when to stop splitting

# How to Specify Test Condition?

- Depends on feature types
  - Discrete
  - Continuous
- Depends on number of ways to split
  - Binary split
  - Multi-way split

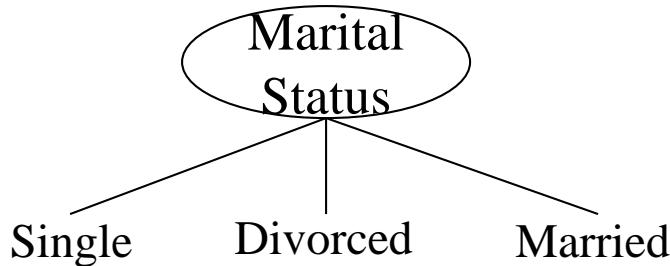
# Splitting Based on Binary Features

- Generate two potential outcomes

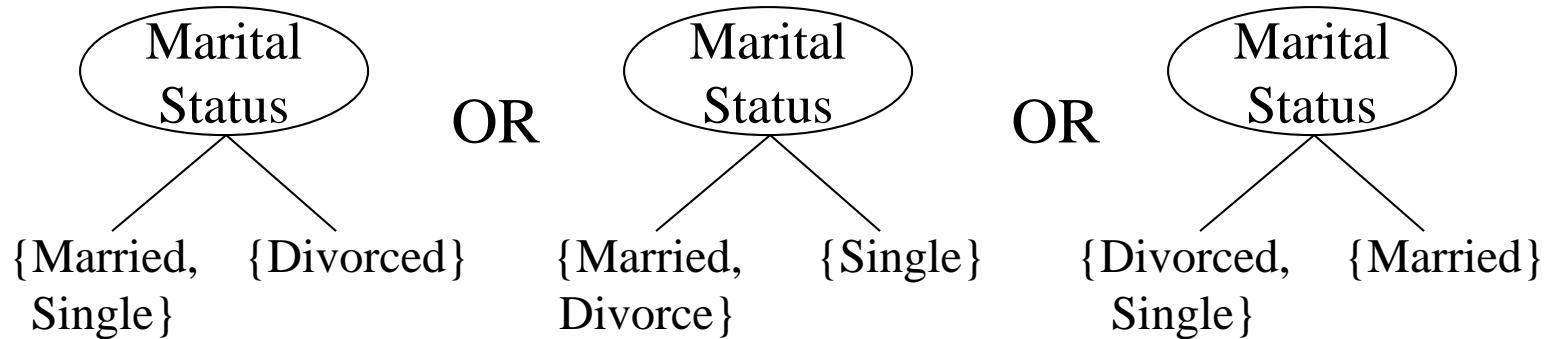


# Splitting Based on Discrete Features (more than two distinct values)

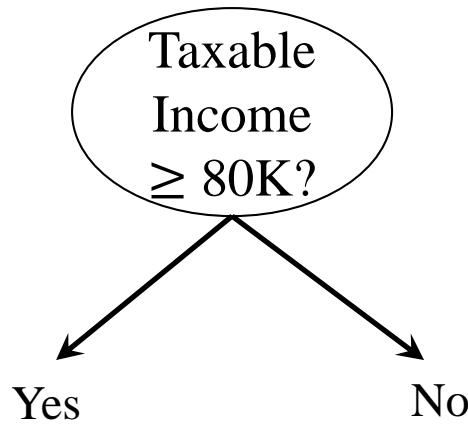
- Multi-way split: Use as many partitions as distinct values



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

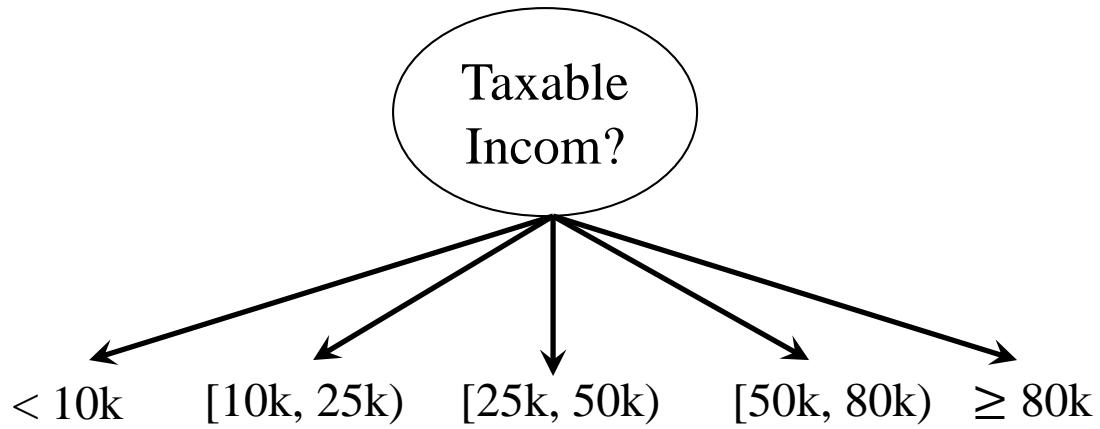


# Splitting Based on Continuous Features



(b) Binary split

$$(x_j < v) \text{ or } (x_j \geq v)$$



(a) Multi-way split

Discretization

- Consider all possible splits and find the best cut
- Can be very computationally intensive

# Tree Induction

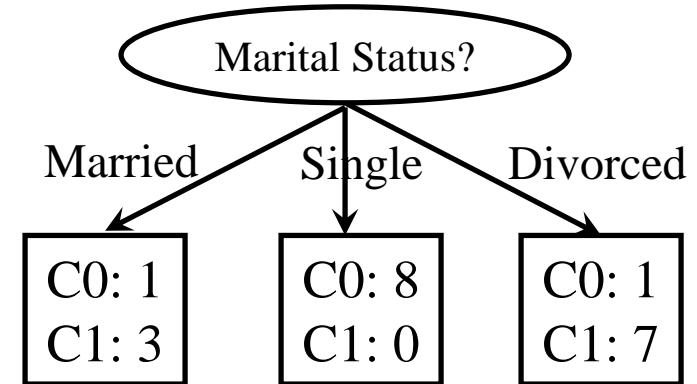
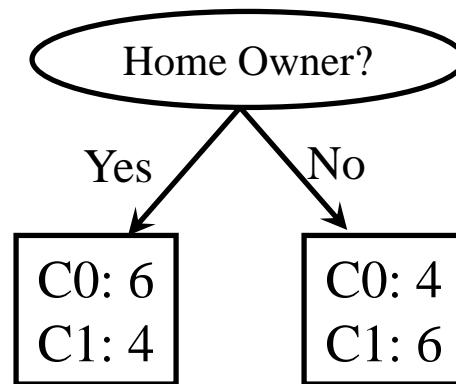
- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

# How to Determine the Best Split

Before Splitting:

10 records of class 0  
10 records of class 1

After Splitting:



Which condition test is the best?

# How to Determine the Best Split (cont.)

- An intuitive idea:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity

C0: 5
C1: 5

Non-homogeneous,  
High degree of impurity

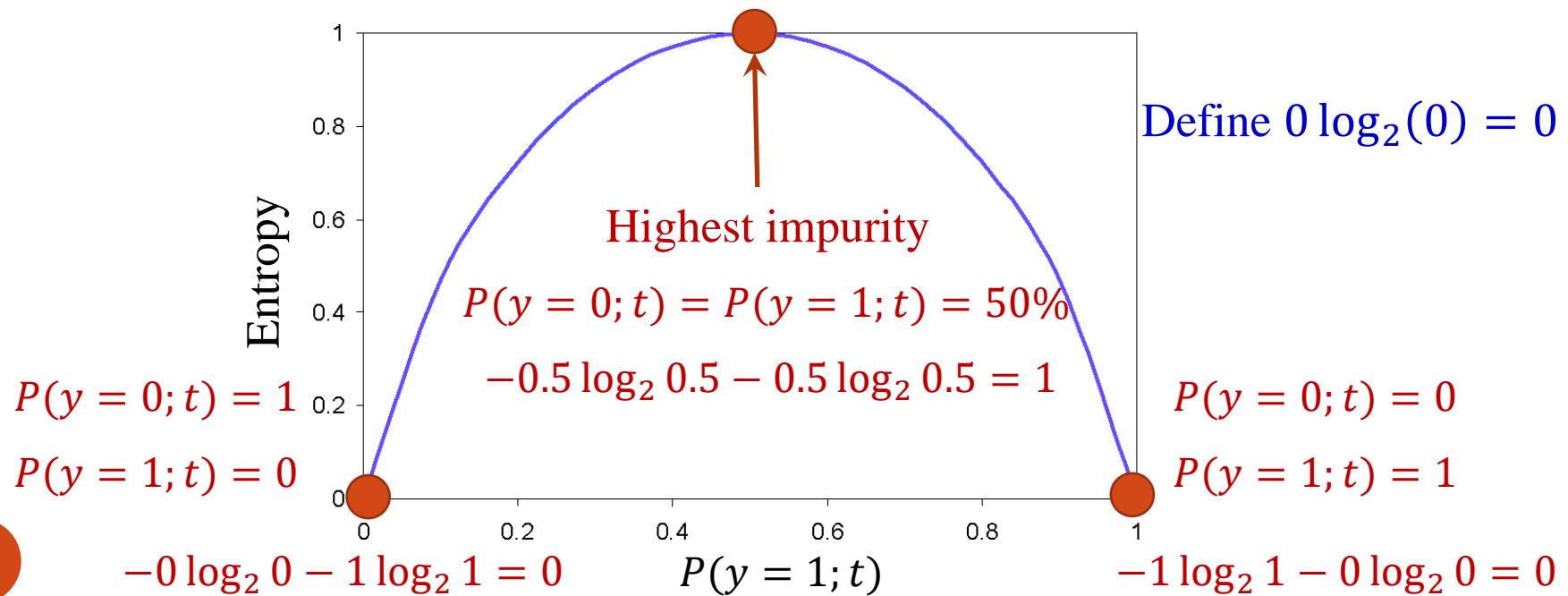
C0: 9
C1: 1

Homogeneous,  
Low degree of impurity

- A split criterion is defined in terms of the difference in degree of node impurity before and after splitting

# Measure of Impurity: Entropy

- Entropy at a given node  $t$ :  
The probability of class  $c$  at the node  $t$   
$$\text{Entropy}(t) = - \sum_c P(y = c; t) \log_2 P(y = c; t)$$
- Entropy for a given node  $t$ , with binary classes:



# Entropy Properties

- Entropy at a given node  $t$ :

$$\text{Entropy}(t) = - \sum_c P(y = c; t) \log_2 P(y = c; t)$$

- Maximum:  $\log_2 C$  Total number of all possible values of  $y$ , i.e., #classes
  - when records are equally distributed among all classes
- Minimum: 0
  - when all records belong to one class

# Examples of Computing Entropy

$$\text{Entropy}(t) = - \sum_c P(y = c; t) \log_2 P(y = c; t)$$

$\mathbf{Y}_1$	<b>0</b>
$\mathbf{Y}_2$	<b>6</b>

$$P(Y_1) = \frac{0}{6} = 0 \quad P(Y_2) = \frac{6}{6} = 1$$

$$\text{Entropy} = -0 \log_2(0) - 1 \log_2(1) = -0 - 0 = 0$$

$\mathbf{Y}_1$	<b>1</b>
$\mathbf{Y}_2$	<b>5</b>

$$P(Y_1) = \frac{1}{6} \quad P(Y_2) = \frac{5}{6}$$

$$\text{Entropy} = -\left(\frac{1}{6}\right) \log_2 \left(\frac{1}{6}\right) - \left(\frac{5}{6}\right) \log_2 \left(\frac{5}{6}\right) = 0.65$$

$\mathbf{Y}_1$	<b>2</b>
$\mathbf{Y}_2$	<b>4</b>

$$P(Y_1) = \frac{2}{6} \quad P(Y_2) = \frac{4}{6}$$

$$\text{Entropy} = -\left(\frac{2}{6}\right) \log_2 \left(\frac{2}{6}\right) - \left(\frac{4}{6}\right) \log_2 \left(\frac{4}{6}\right) = 0.92$$

# Information Gain: Motivation

- Recall: a split criterion should be defined in terms of the difference in degree of node impurity before and after splitting
- Information Gain:

$$\Delta_{\text{info}} = \text{Entropy}(\text{parent node}) - \text{Entropy}(\text{children nodes})$$

- Choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the child nodes)

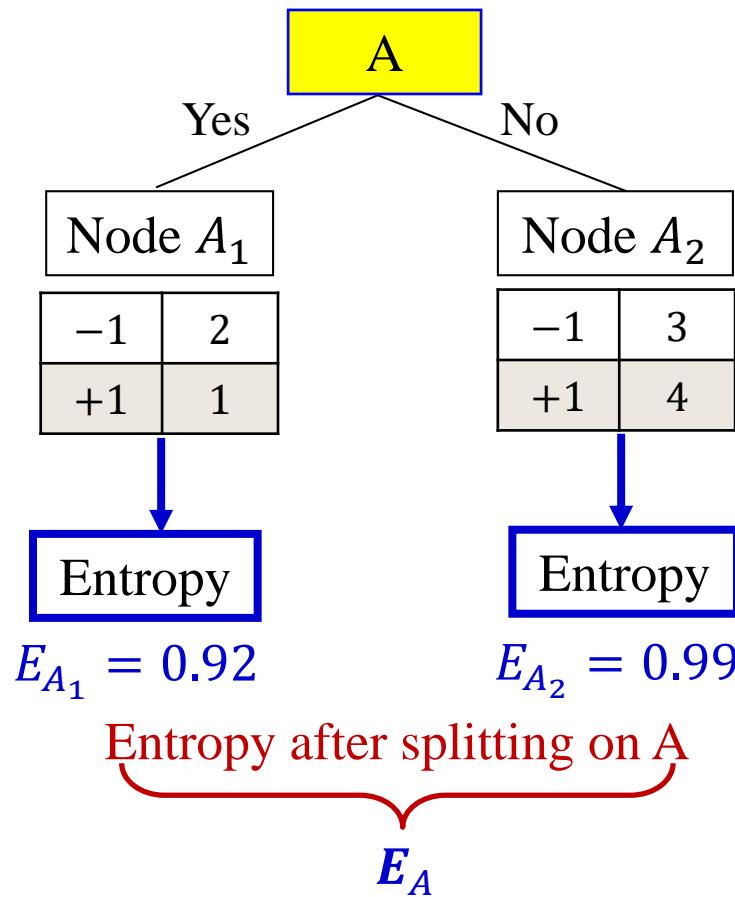
## A motivating example of Information Gain

Node  $P$

-1	5
+1	5

Entropy before splitting

$$\text{Entropy} \quad E_p = 1$$



#instances  
in node  $A_1$

$$E_A = \frac{\#A_1}{\#P} E_{A_1} + \frac{\#A_2}{\#P} E_{A_2}$$

#instances  
in node  $A_2$

$$\begin{aligned}
 &= \frac{3}{10} 0.92 + \frac{7}{10} 0.99 \\
 &= 0.97
 \end{aligned}$$

Use the **Information Gain**:  $E_p - E_A$   
to measure the difference of impurity  
before and after splitting on  $A$

$$E_p - E_A = 0.03$$

Goal: find a feature with maximum information gain to conduct condition test

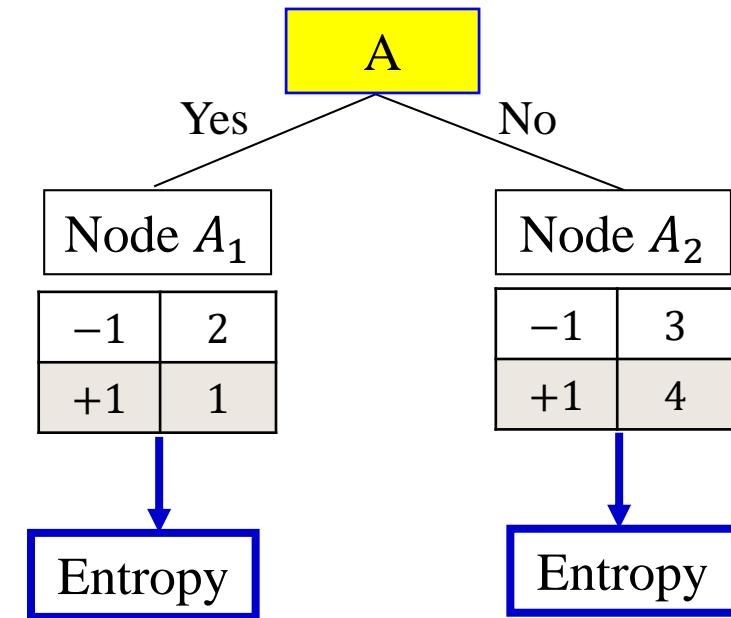
A motivating example  
of Information Gain

Node  $P$

-1	5
+1	5

Entropy before splitting

$$\text{Entropy} \quad E_p = 1$$



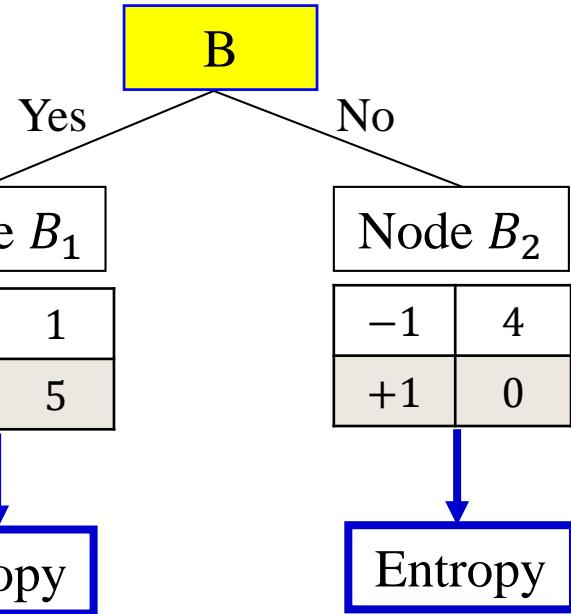
$$E_{A_1} = 0.92$$

$$E_{A_2} = 0.99$$

Entropy after splitting on A

$$E_A = \frac{3}{10} \times 0.92 + \frac{7}{10} \times 0.99 = 0.97$$

Information Gain:  $E_p - E_A = 0.03$



$$E_{B_1} = 0.65$$

$$E_{B_2} = 0$$

Entropy after splitting on B

$$E_B = \frac{6}{10} \times 0.65 + \frac{4}{10} \times 0 = 0.39$$

Information Gain:  $E_p - E_B = 0.61$



# Information Gain: Definition

- Suppose a parent node  $t$  is split into  $P$  partitions (children)
- Information Gain:

Number of examples at node  $t$

$$\Delta_{\text{info}} = \text{Entropy}(t) - \sum_{j=1}^P \frac{n_j}{n} \text{Entropy}(j)$$

Number of examples at child  $j$

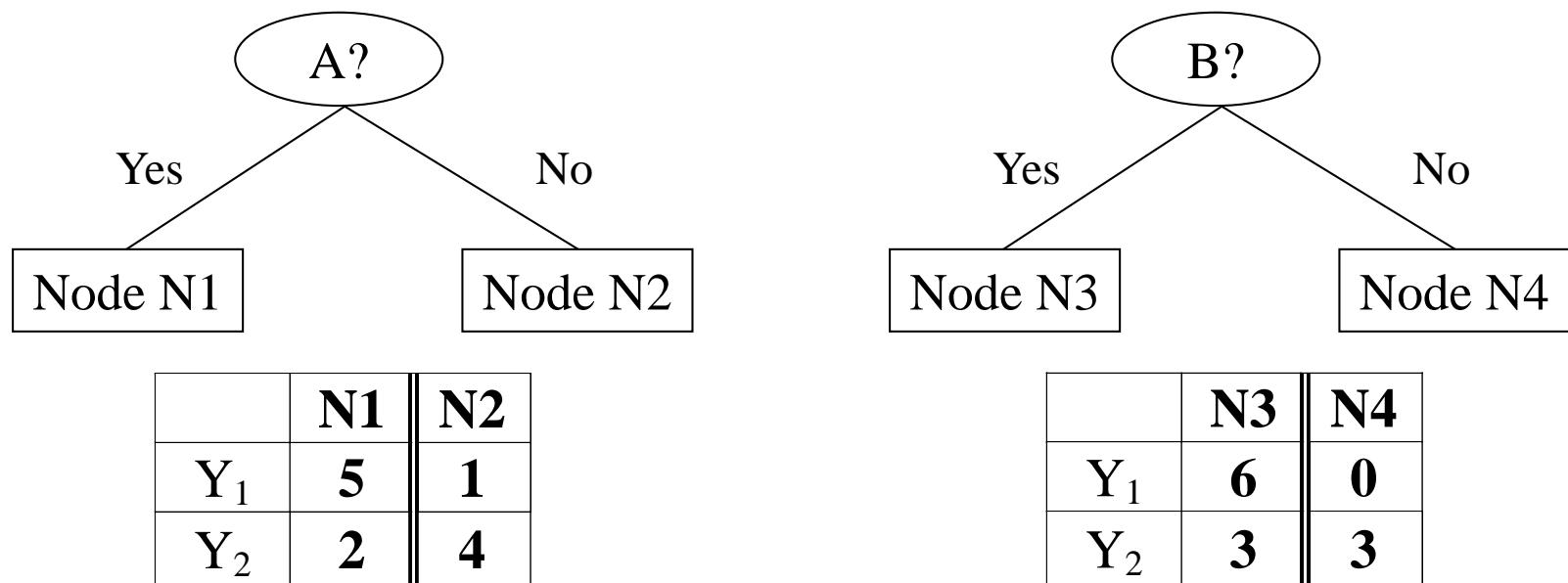
The diagram illustrates the components of the information gain formula. On the left, a large orange bracket groups the term "Number of examples at node  $t$ " with the first term of the equation,  $\text{Entropy}(t)$ . On the right, another orange bracket groups the term "Number of examples at child  $j$ " with the fraction  $n_j/n$ .

- To choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the children nodes)

# Information Gain: Practice

- Suppose features  $A$  and  $B$  are binary
- Based on Information Gain, which feature should be selected to split data?

	<b>Parent</b>
$Y_1$	<b>6</b>
$Y_2$	<b>6</b>



Entropy(Parent)

$$= -\left(\frac{6}{12}\right) \log_2 \left(\frac{6}{12}\right) - \left(\frac{6}{12}\right) \log_2 \left(\frac{6}{12}\right) = 1$$

	<b>Parent</b>
$Y_1$	6
$Y_2$	6

$$\text{Entropy}(N1) = -\left(\frac{5}{7}\right) \log_2 \left(\frac{5}{7}\right) - \left(\frac{2}{7}\right) \log_2 \left(\frac{2}{7}\right) = 0.8631$$

$$\text{Entropy}(N2) = -\left(\frac{1}{5}\right) \log_2 \left(\frac{1}{5}\right) - \left(\frac{4}{5}\right) \log_2 \left(\frac{4}{5}\right) = 0.7219$$

$$\text{Entropy}(\text{Split}_A) = \left(\frac{7}{12}\right) \times 0.8631 + \left(\frac{5}{12}\right) \times 0.7219 = 0.8043$$

	<b>N1</b>	<b>N2</b>
$Y_1$	5	1
$Y_2$	2	4

**Split on A**

$$\text{Entropy}(N3) = -\left(\frac{3}{9}\right) \log_2 \left(\frac{3}{9}\right) - \left(\frac{6}{9}\right) \log_2 \left(\frac{6}{9}\right) = 0.9183$$

$$\text{Entropy}(N4) = -\left(\frac{0}{3}\right) \log_2 \left(\frac{0}{3}\right) - \left(\frac{3}{3}\right) \log_2 \left(\frac{3}{3}\right) = 0$$

	<b>N3</b>	<b>N4</b>
$Y_1$	6	0
$Y_2$	3	3

$$\text{Entropy}(\text{Split}_B) = \left(\frac{9}{12}\right) \times 0.9183 + \left(\frac{3}{12}\right) \times 0 = 0.6887$$

**Split on B**

$$\text{Entropy(Parent)} = -\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) - \left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right) = 1$$

	<b>Parent</b>
$Y_1$	6
$Y_2$	6

## Split on *A*

$$\text{Entropy}(\text{Split}_A) = \left(\frac{7}{12}\right) \times 0.8631 + \left(\frac{5}{12}\right) \times 0.7219 = 0.8043$$

	<b>N1</b>	<b>N2</b>
$Y_1$	5	1
$Y_2$	2	4

## Split on *B*

$$\text{Entropy}(\text{Split}_B) = \left(\frac{9}{12}\right) \times 0.9183 + \left(\frac{3}{12}\right) \times 0 = 0.6887$$

	<b>N3</b>	<b>N4</b>
$Y_1$	6	0
$Y_2$	3	3

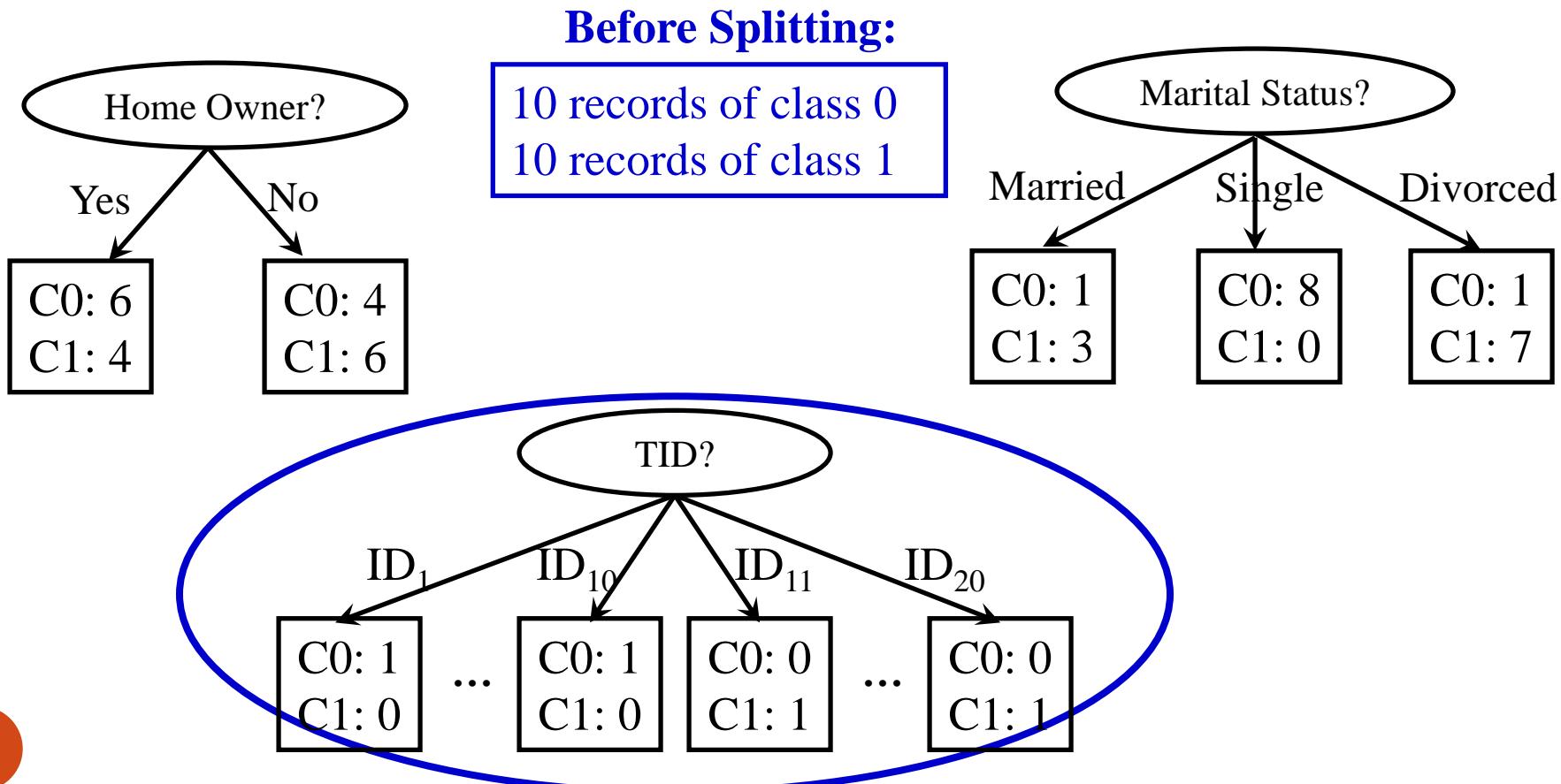
$$\Delta_{\text{info}}(A) = 1 - 0.8043 = 0.1957$$

$$\Delta_{\text{info}}(B) = 1 - 0.6887 = 0.3113$$

Choose *B* to conduct condition test to split data

# Information Gain: Limitation

- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure



# Splitting Based on Entropy (cont.)

- Gain Ratio:

Maximum:  $\log_2 P$

$$\Delta_{\text{InfoR}} = \frac{\Delta_{\text{info}}}{\text{SplitINFO}}$$

Penalty on large number  
of small partitions

where 
$$\text{SplitINFO} = - \sum_{i=1}^P \frac{n_i}{n} \log_2 \left( \frac{n_i}{n} \right)$$

- Parent node  $t$  of  $n$  instances is split into  $P$  partitions (children),  $n_i$  is the number of instances in partition  $i$
- Higher entropy partitioning (large number of small partitions) is penalized!

Refer to Question 1 in tutorial for practice

# Tree Induction

- Determine how to split the data
  - How to specify the feature test condition?
  - How to determine the best split?
- Determine when to stop splitting

# Stopping Criteria for Tree Induction

- Stop expanding a node when all the data instances belong to the same class
  - Ideal case but not always possible
- Stop expanding a node when all the data instances have similar feature values
- Early termination
  - Useful to avoid the overfitting issue (will be introduced next week)

# Decision Tree Classifiers: Summary

- Easy to interpret
- Efficient in both training and testing
- Effective for datasets with a lot of categorical features
- Used as a base classifier in many ensemble learning approaches (will be introduced in the 2<sup>nd</sup> half)

# Implementation --- scikit-learn

- API: `sklearn.tree.DecisionTreeClassifier`

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree>

## sklearn.tree: Decision Trees

The `sklearn.tree` module includes decision tree-based models for classification and regression.

**User guide:** See the [Decision Trees](#) section for further details.

`tree.DecisionTreeClassifier(*  
[, criterion, ...])` A decision tree classifier.

`tree.DecisionTreeRegressor(*  
[, criterion, ...])` A decision tree regressor.

`tree.ExtraTreeClassifier(*  
[, criterion, ...])` An extremely randomized tree classifier.

`tree.ExtraTreeRegressor(*  
[, criterion, ...])` An extremely randomized tree regressor.

`tree.export_graphviz(decision_tree[, ...])` Export a decision tree in DOT format.

`tree.export_text(decision_tree, *[..., ...])` Build a text report showing the rules of a decision tree.

## Plotting

`tree.plot_tree(decision_tree, *  
[, ...])` Plot a decision tree.

# An Example

```
>>> from sklearn import tree
```

```
>>> ...
```

Data preprocessing

```
>>> dtC = tree.DecisionTreeClassifier()
```

```
>>> dtC.fit(X, y)
```

```
>>> pred= dtC.predict(X_t)
```

**Thank you!**

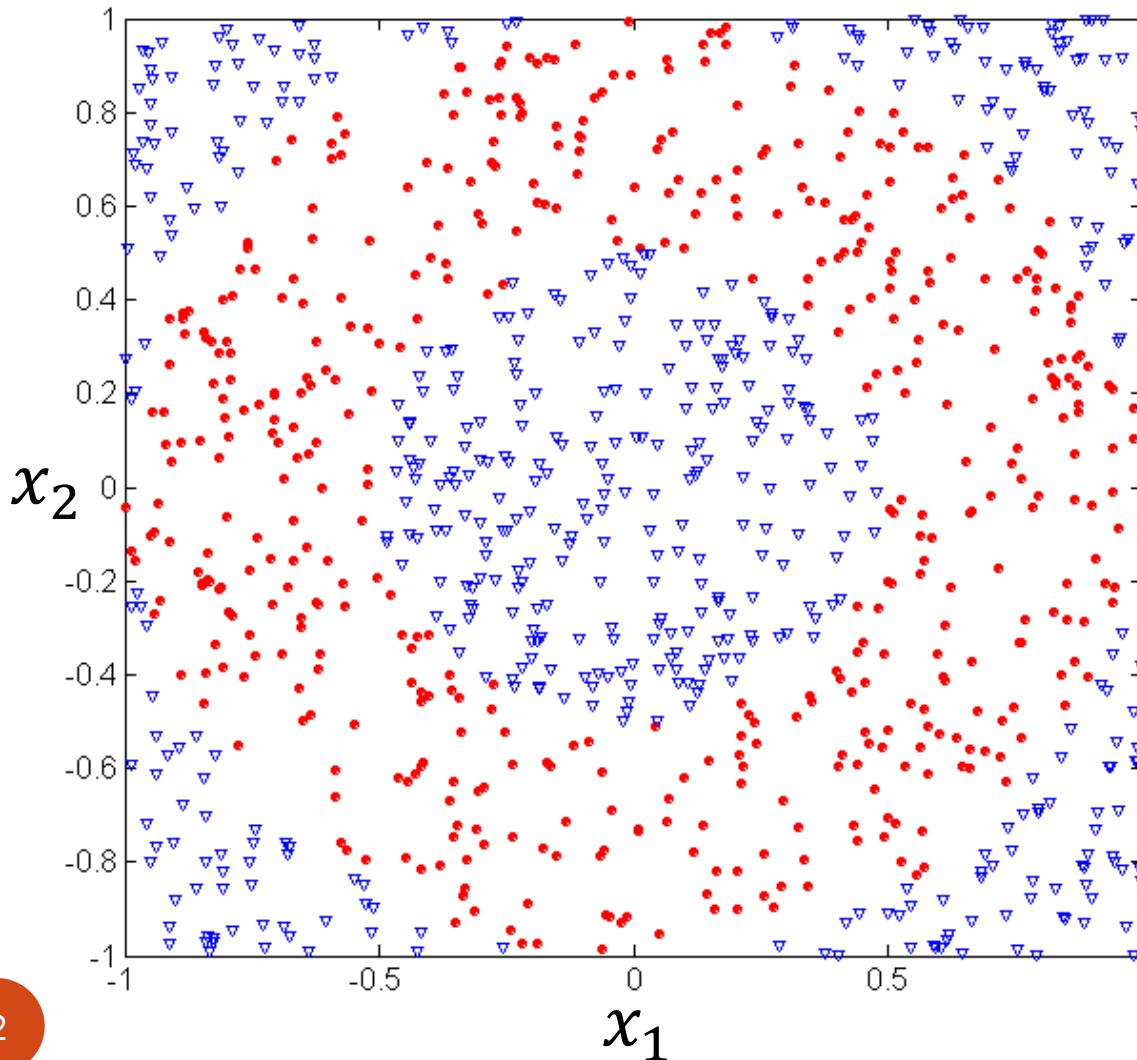
# SC4000/CZ4041/CE4041: Machine Learning

## Lesson 6a: Generalization

Kelly KE

College of Computing and Data Science  
NTU, Singapore

# Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

$$0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1$$

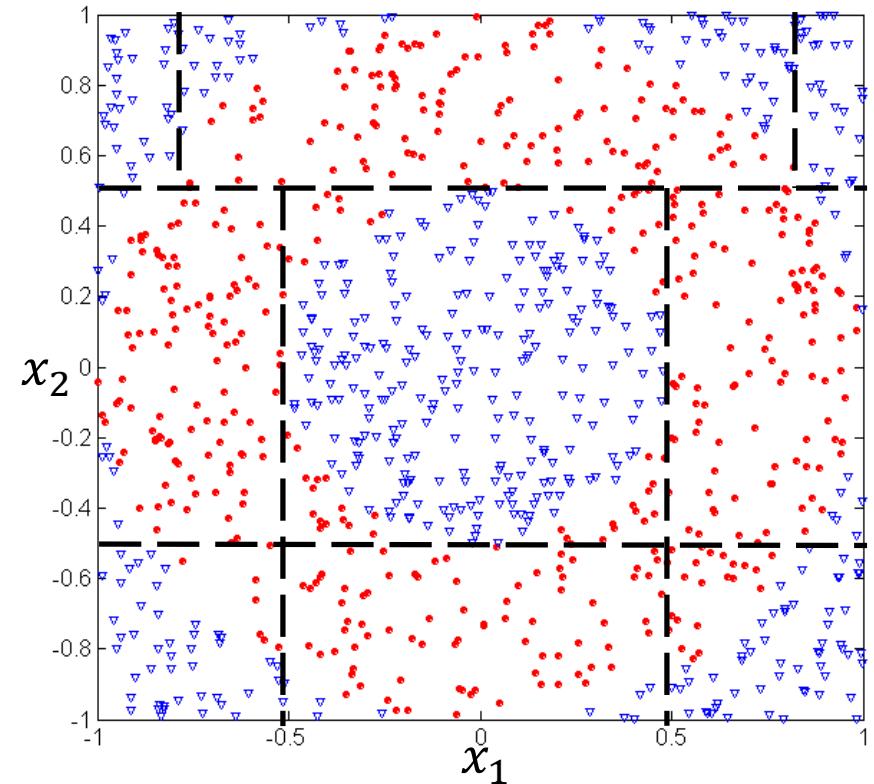
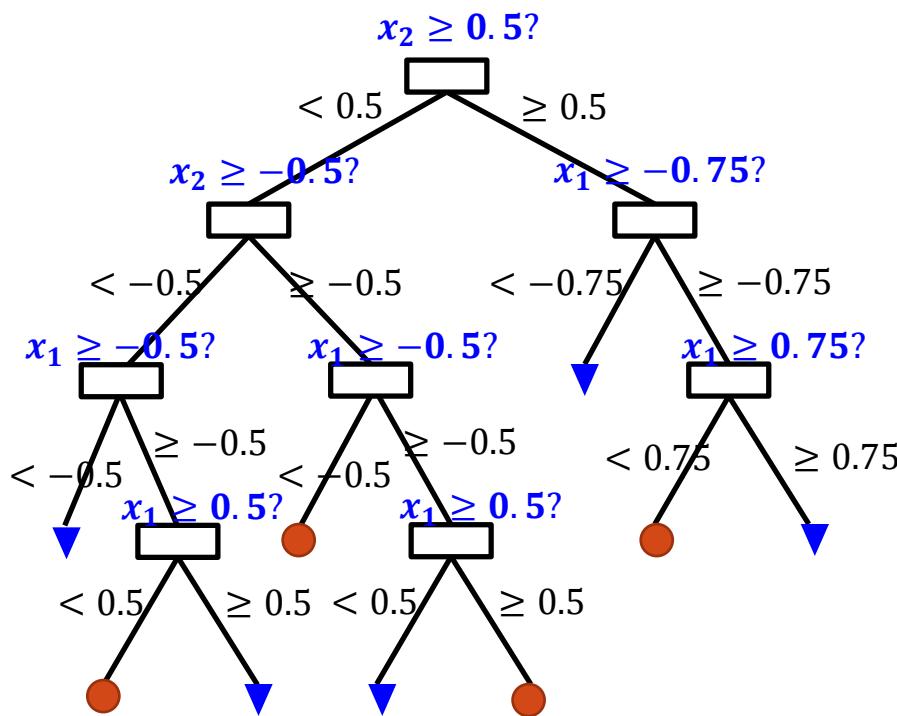
Triangular points:

$$\sqrt{x_1^2 + x_2^2} > 1$$

or

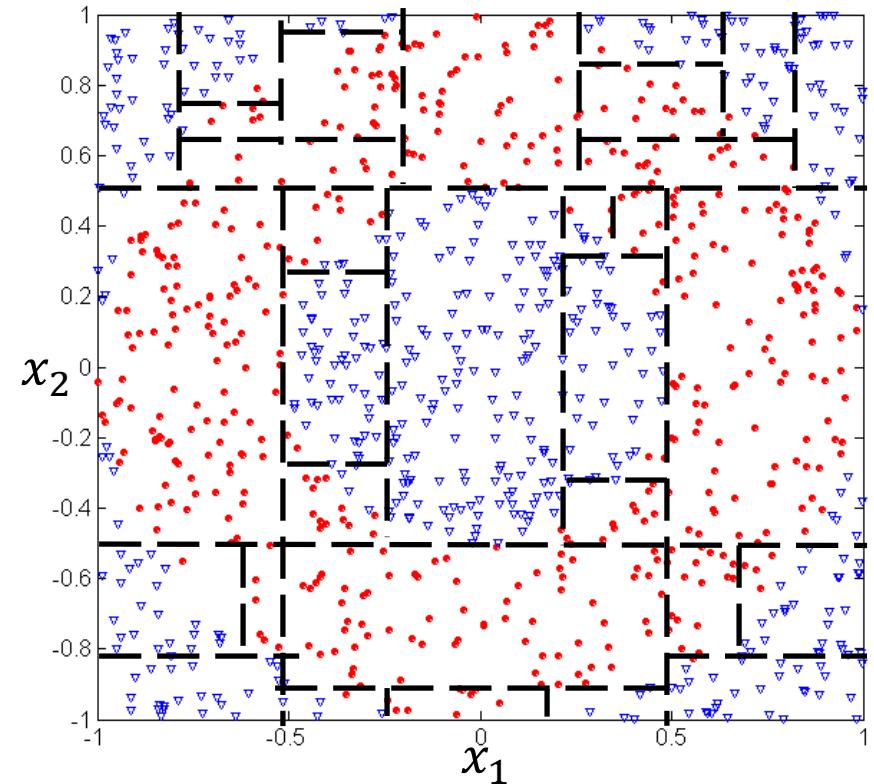
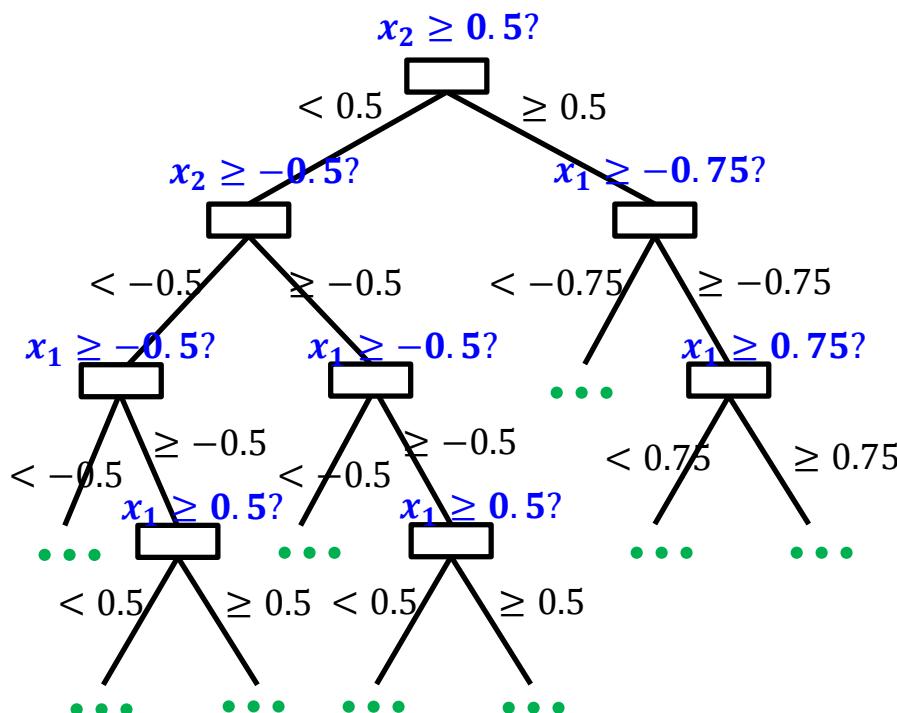
$$\sqrt{x_1^2 + x_2^2} < 0.5$$

# Overfitting v.s. Model Complexity



Training errors (#misclassified training data) = 100 +  
Decision tree with 9 leaf nodes

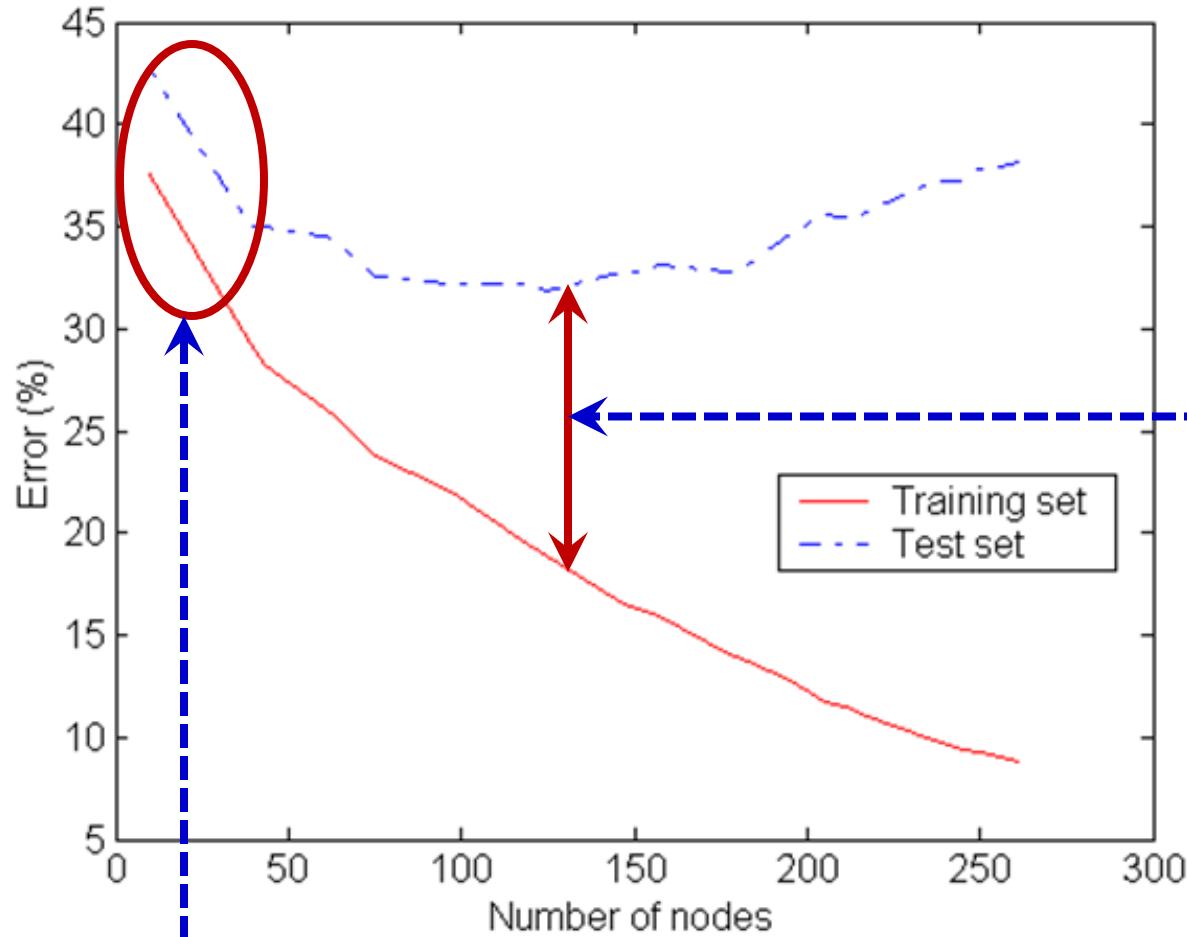
# Overfitting v.s. Model Complexity



Training errors = 20

Decision tree with 30 + leaf nodes

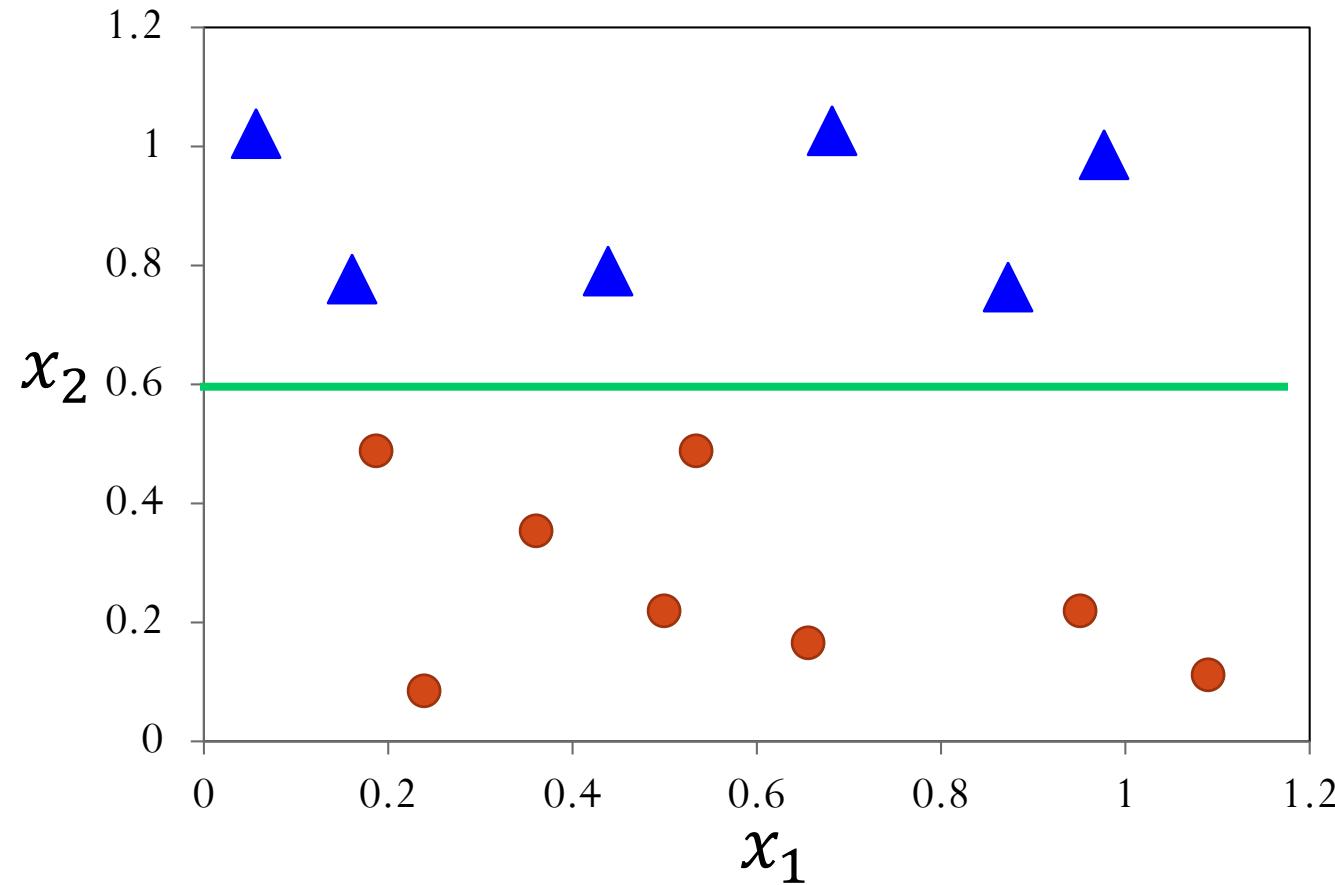
# Underfitting and Overfitting



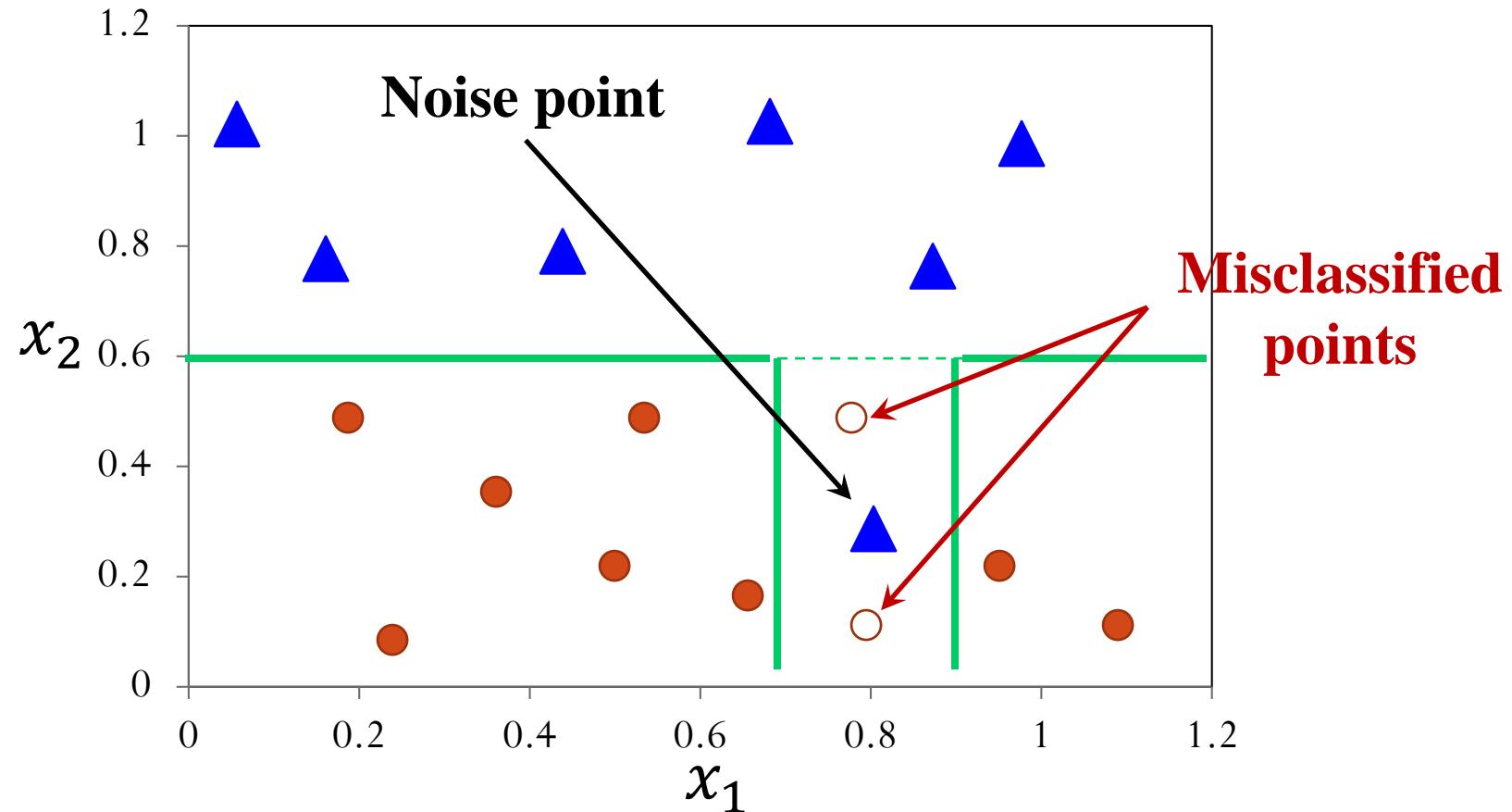
Overfitting: when test error rate begins to increase even though training error rate continues to decrease

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

# Overfitting due to Noise



# Overfitting due to Noise (cont.)



Decision boundary is distorted by noise point

# Notes on Overfitting

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

## Overfitting v.s. Model Complexity (cont.)

- How do we determine the right model complexity?
- A model with ideal complexity is the one that produces the lowest generalization error
- No knowledge of the test data and how well the model will perform on unseen data
- The best it can do is to estimate the generalization error of the induced model

# Estimation of Generalization Errors

- Training errors: error on the training set:  
 $e(T)$
- Generalization errors: error on previously  
unseen testing set:  $e'(T)$

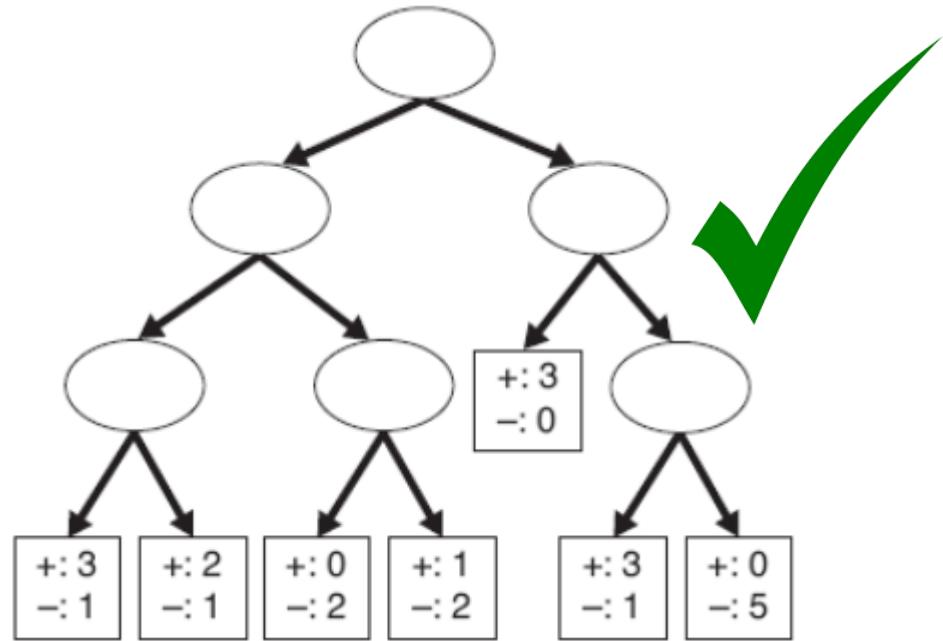
# Estimation of Generalization Errors

- Approaches to estimating generalization errors:
  - Optimistic Estimate:  $e'(T) = e(T)$
  - Incorporating model complexity
    - Occam's Razor
      - Pessimistic Error Estimate
    - Using Validation Set

# Optimistic Estimate

- Assume that the training set is a good representation of the overall data
- The training error can be used to provide an optimistic estimate for the generalization error
  - $e'(T) = e(T)$
- A decision tree induction algorithm selects the model that produces the lowest training error rate

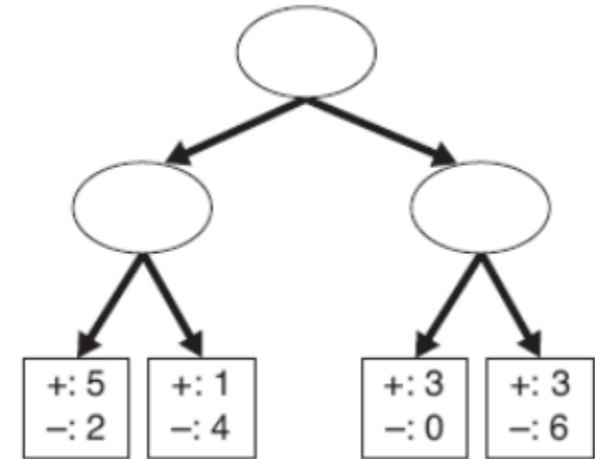
# An Example of Optimistic Estimate



Decision Tree,  $T_L$

$$e(T_L) = 4$$

$$e(T_L) \text{ rate} = \frac{4}{24} = 0.167$$



Decision Tree,  $T_R$

$$e(T_R) = 6$$

$$e(T_R) \text{ rate} = \frac{6}{24} = 0.25$$

# Estimation of Generalization Errors

- Approaches to estimating generalization errors:
  - Optimistic Estimate:  $e'(T) = e(T)$
  - Incorporating model complexity
    - Occam's Razor
      - Pessimistic Error Estimate
    - Using Validation Set

# Occam's Razor

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

*“Everything should be made as simple as possible, but no simpler.” – Albert Einstein*

# Pessimistic Error Estimate

- Idea: explicitly computes generalization error as the sum of training error and a penalty term for model complexity

$$e'(T) = e(T) + \Omega(T)$$

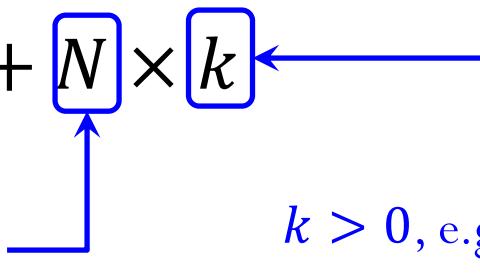
- In a decision tree, we can define a penalty term of  $k > 0$  on each leaf node, i.e.,

$$e'(t) = e(t) + \Omega(t) = e(t) + k$$

- Then,

$$e'(T) = e(T) + N \times k$$

Total number of leaf nodes       $k > 0$ , e.g.,  $k = 0.5$



# Example

- For a tree with 30 leaf nodes and 10 errors on training (out of 1,000 instances),  $k = 0.5$ :
  - Training errors = 10
  - Training error rate =  $\frac{10}{1000} = 1\%$
  - Generalization errors =  $10 + 30 \times 0.5 = 25$
  - Generalization error rate =  $\frac{25}{1000} = 2.5\%$

# Estimation of Generalization Errors

- Approaches to estimating generalization errors:
  - Optimistic Estimate:  $e'(T) = e(T)$
  - Incorporating model complexity
    - Occam's Razor
      - Pessimistic Error Estimate
  - Using Validation Set

# Using a Validation Set

- Divide the original training data set into two smaller subsets
- One is for training, the other (known as the validation set) is for estimating the generalization error
- The complexity of the best model can be estimated based on the performance of the model on the validation set

# How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the feature values are the same

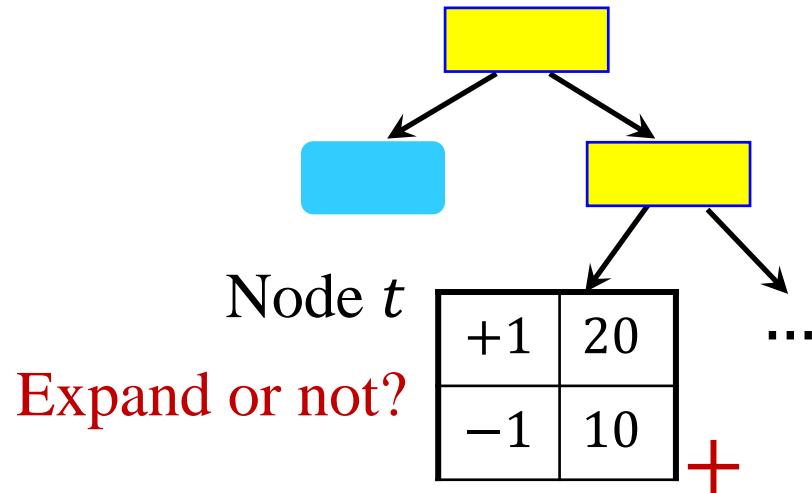
# How to Address Overfitting (cont.)

- Pre-Pruning (Early Stopping Rule)
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if expanding the current node does not improve generalization errors

# Pre-Pruning Example

Generalization errors:  $e'(T) = e(T) + \Omega(T) = e(T) + N \times k$

e.g.,  $k = 0.5$



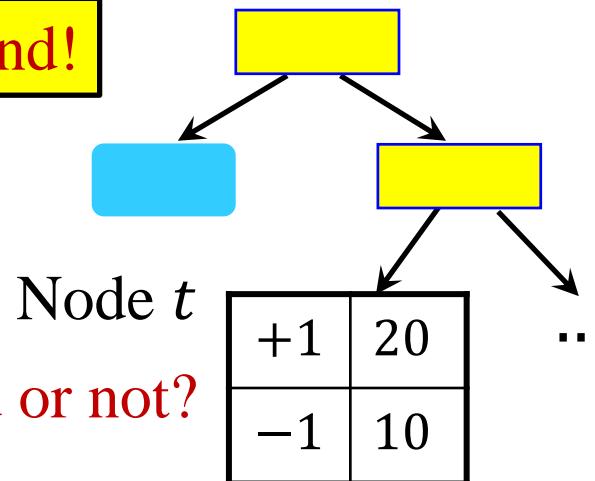
$$\begin{aligned}e'(t) &= e(t) + \Omega(t) \\&= 10 + 1 \times 0.5 \\&= 10.5\end{aligned}$$

A subtree with node  $t$  as its root

# Pre-Pruning Example (cont.)

Generalization errors:  $e'(T) = e(T) + N \times 0.5$

Not expand!

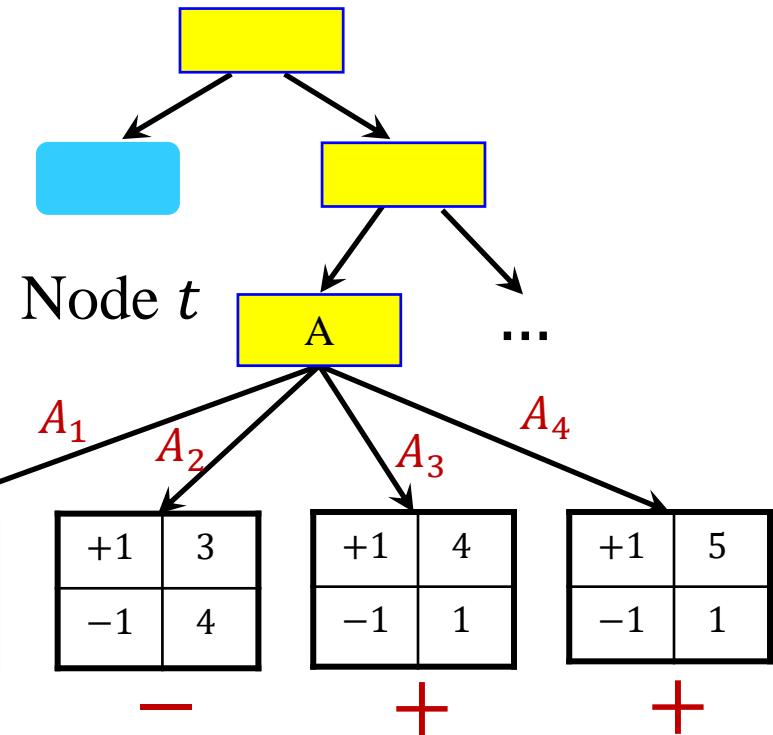


Expand or not?

A subtree with node  $t$  as its root

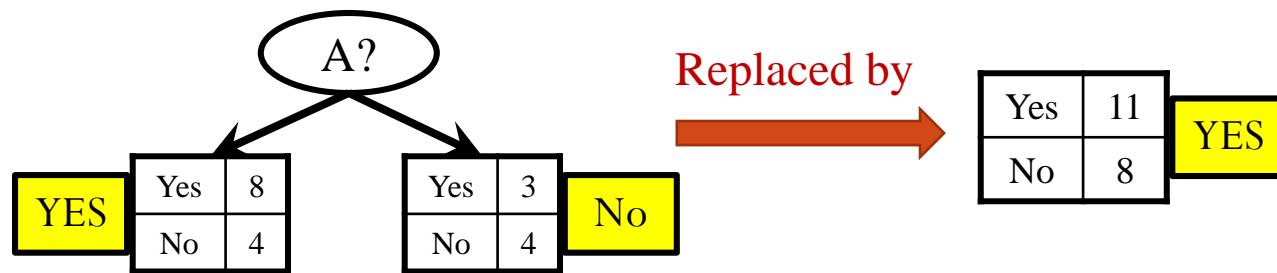
$$\begin{aligned} e'(t) &= e(t) + \Omega(t) \\ &= 9 + 0.5 \times 4 \\ &= 11 \end{aligned}$$

If yes, suppose  $A$  is best feature to conduct condition test



# How to Address Overfitting (cont.)

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

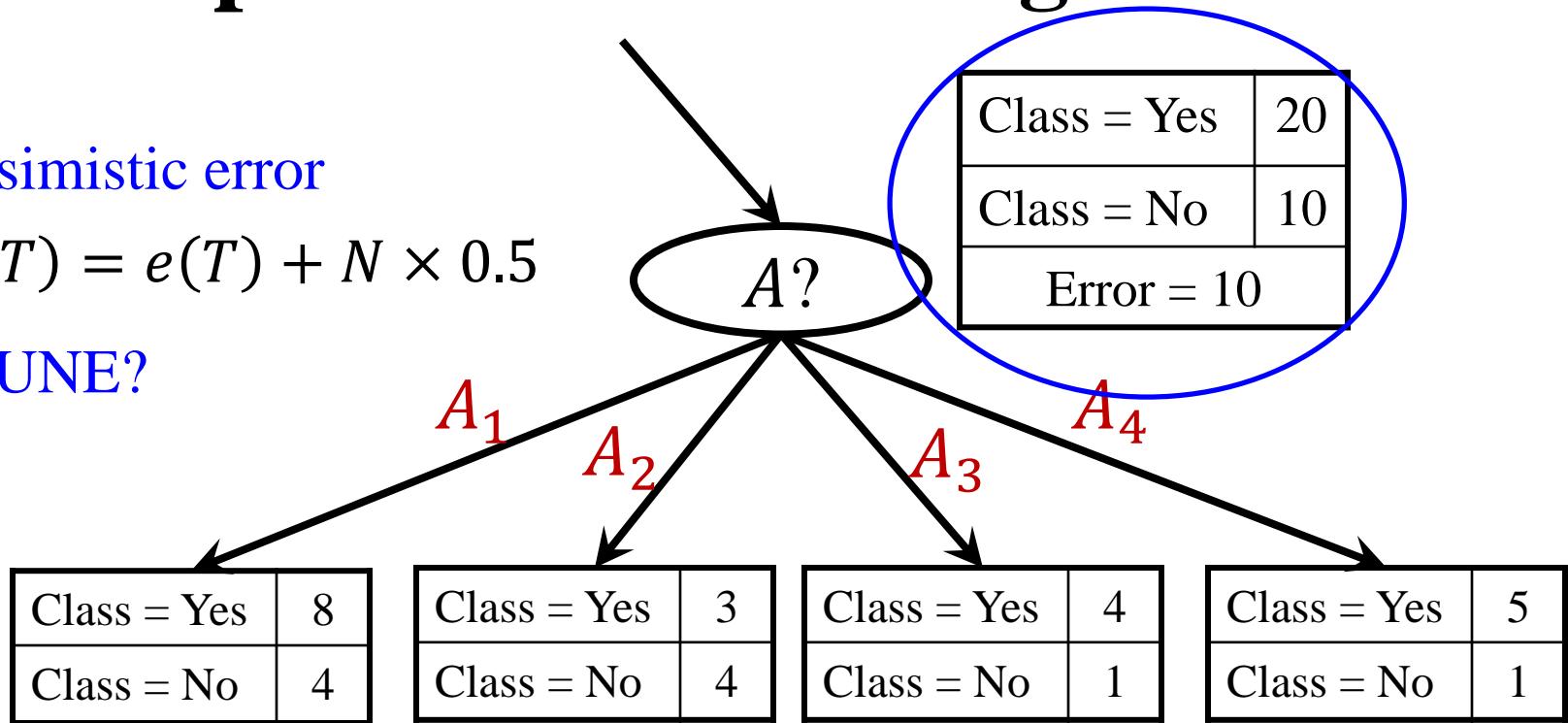


# Example of Post-Pruning

Pessimistic error

$$e'(T) = e(T) + N \times 0.5$$

PRUNE?



$$\text{Training errors (before pruning)} = 4 + 3 + 1 + 1 = 9$$

$$\text{Pessimistic errors (before pruning)} = 9 + 4 \times 0.5 = 11$$

$$\text{Training errors (after pruning)} = 10$$

$$\text{Pessimistic errors (after pruning)} = 10 + 0.5 = 10.5$$

**PRUNE!**

# Examples of Post-pruning

- Pessimistic error?

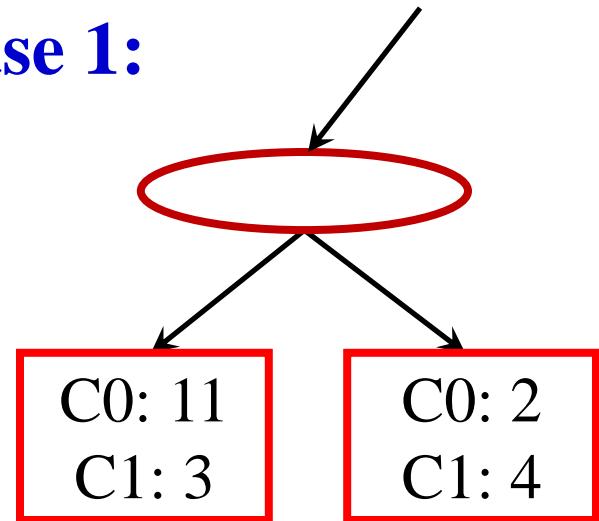
$$e'(T) = e(T) + N \times 0.5$$

PRUNE?

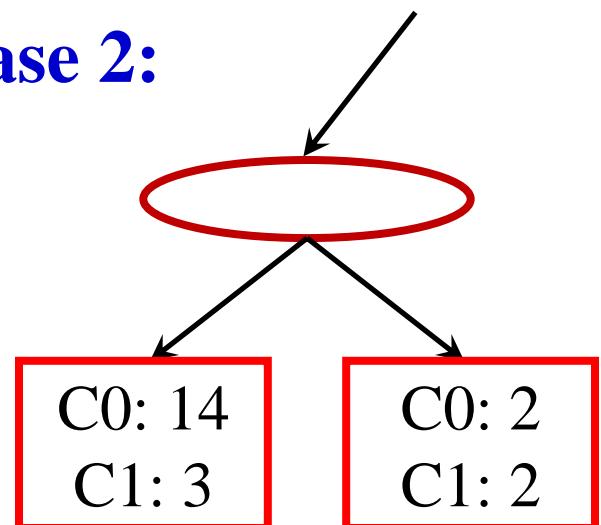
Tutorial



Case 1:



Case 2:



# In Practice

## **criterion : {"gini", "entropy"}, default="gini"**

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

## **max\_depth : int, default=None**

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

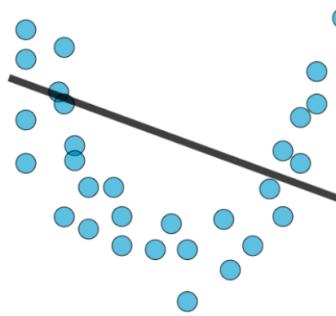
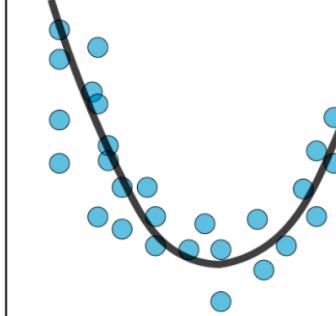
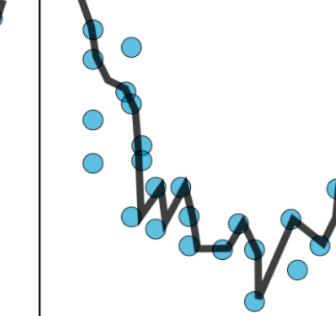
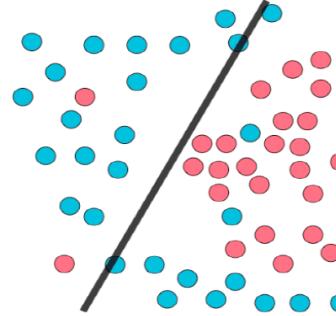
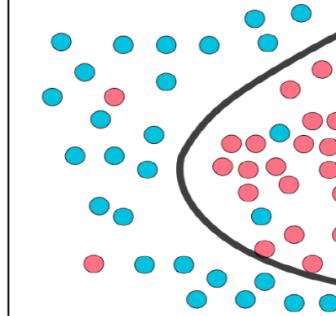
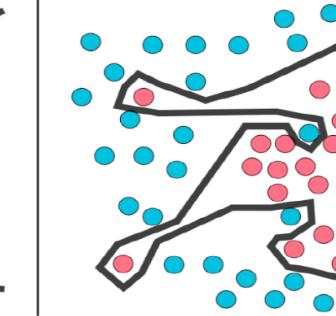
## **max\_features : int, float or {"auto", "sqrt", "log2"}, default=None**

The number of features to consider when looking for the best split:

- If int, then consider max\_features features at each split.
- If float, then max\_features is a fraction and int(max\_features \* n\_features) features are considered at each split.
- If “auto”, then max\_features=sqrt(n\_features).
- If “sqrt”, then max\_features=sqrt(n\_features).
- If “log2”, then max\_features=log2(n\_features).
- If None, then max\_features=n\_features.

**Thank you!**

# Bias v.s. Variance

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"><li>- High training error</li><li>- Training error close to test error</li><li>- High bias</li></ul>	<ul style="list-style-type: none"><li>- Training error slightly lower than test error</li></ul>	<ul style="list-style-type: none"><li>- Low training error</li><li>- Training error much lower than test error</li><li>- High variance</li></ul>
Regression			
Classification			

# SC4000/CZ4041/CE4041: Machine Learning

## Lesson 6b: K-NN Classifiers

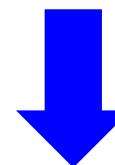
Kelly KE

College of Computing and Data Science  
NTU, Singapore

Acknowledgements: some figures are adapted from the lecture notes of the book  
“Introduction to Data Mining” (Chap. 5)

## Training data

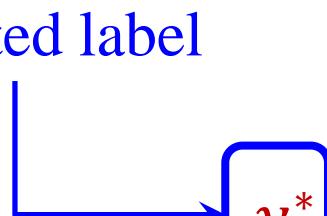
	F1	F2	F3	F4	F5	F6	...	
$y_1$	+1	1	1	0	0	1	0	...
$y_2$	-1	0	0	1	1	0	1	...
	...							
$y_N$	-1	0	1	0	0	1	1	...

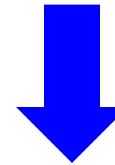


Some classification algorithm

Predicted label

$$f: \mathbf{x} \rightarrow y$$


$$y^* = f(\mathbf{x}^*)$$



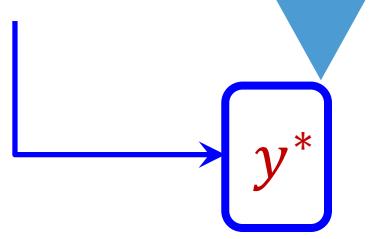
Test data

	F1	F2	F3	F4	F5	F6	...
?	1	1	0	0	1	0	...

## Training data

	F1	F2	F3	F4	F5	F6	...	
$y_1$	+1	1	1	0	0	1	0	...
$y_2$	-1	0	0	1	1	0	1	...
$y_N$	-1	0	1	0	0	1	1	...

Predicted label



No model  $f$   
explicitly learnt in training

Test data

?	F1	F2	F3	F4	F5	F6	...
	1	1	0	0	1	0	...

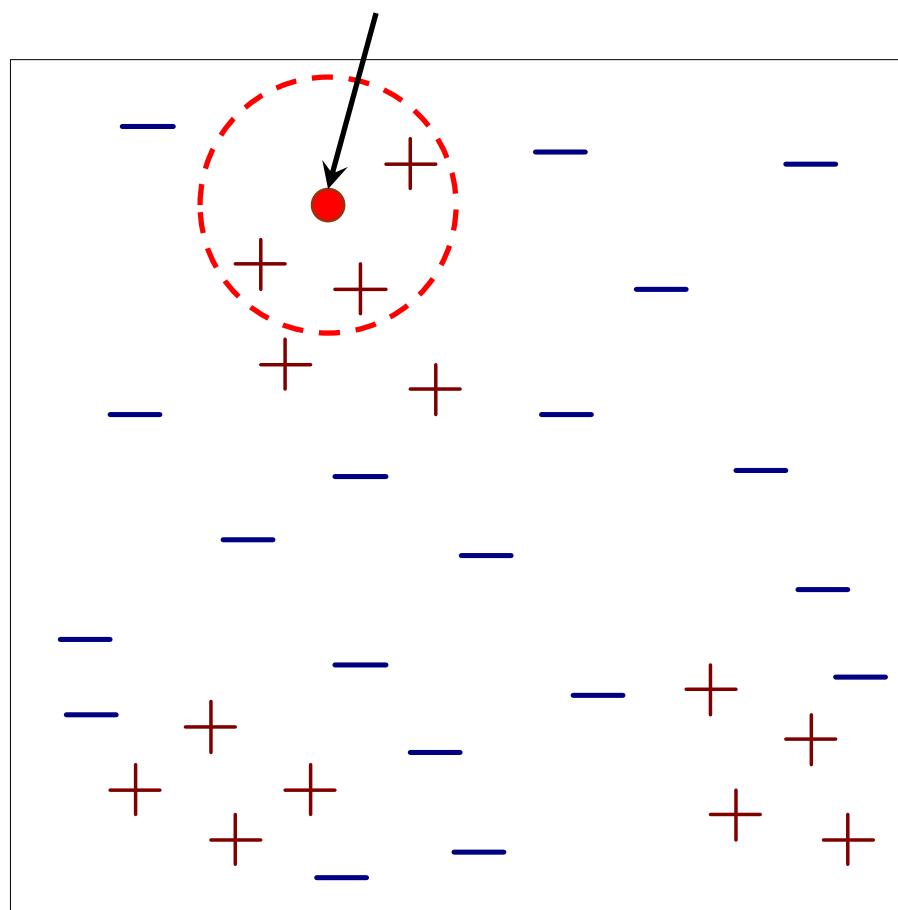
Lazy Learning

# Instance Based Classifiers

- $K$ -Nearest Neighbors classifier
  - Uses  $K$  “closest” points (nearest neighbors) for performing classification

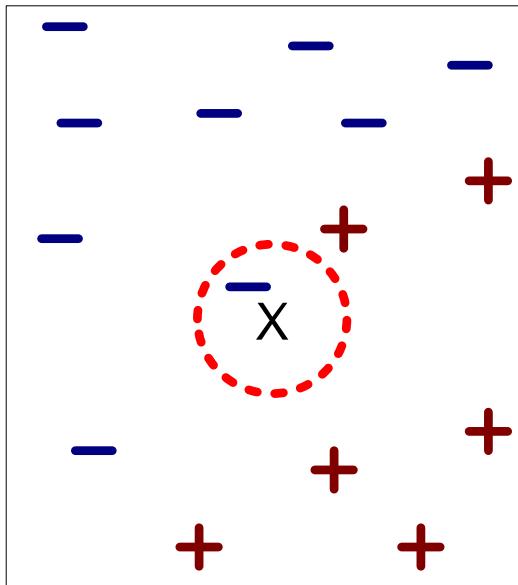
# Nearest-Neighbor Classifiers

Unknown instance

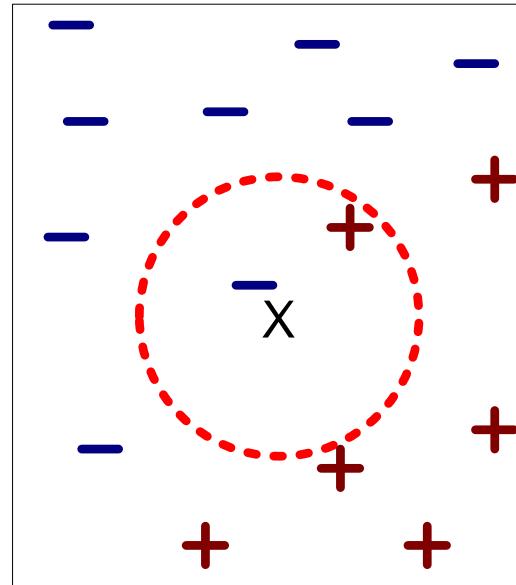


- Requires three things
  - The set of stored labeled instances
  - Distance metric to compute distance between instances
  - The value of  $K$ , the number of nearest neighbors to retrieve
- To classify an unknown instance:
  - Compute distance to all the training instances
  - Identify  $K$  nearest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

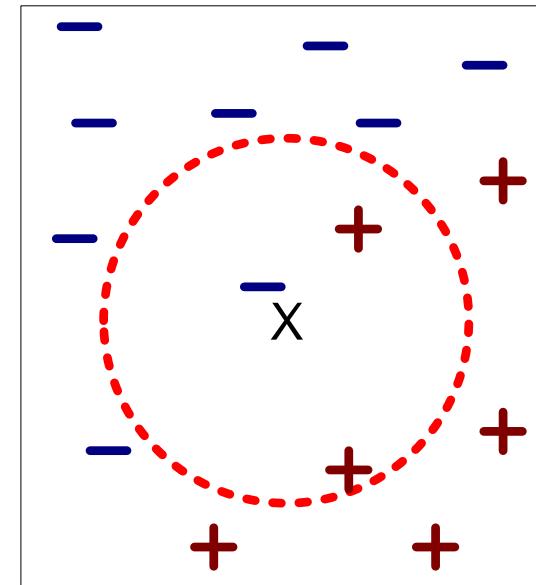
# Definition of Nearest Neighbors



(a) 1-nearest  
neighbor



(b) 2-nearest  
neighbor



(c) 3-nearest  
neighbor

$K$ -nearest neighbors of an instance  $x$  are data points that have the  $K$  smallest distance to  $x$

# Distance Metric

- Compute distance between two data points in a  $d$ -dimensional space:

$\boldsymbol{x}_i$			
$x_{i1}$	$x_{i2}$	$\dots$	$x_{id}$

$\boldsymbol{x}_j$			
$x_{j1}$	$x_{j2}$	$\dots$	$x_{jd}$

- Euclidean distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$$

$L_2$  norm distance

Inner product between  $\boldsymbol{x}_i$  and  $\boldsymbol{x}_j$ :

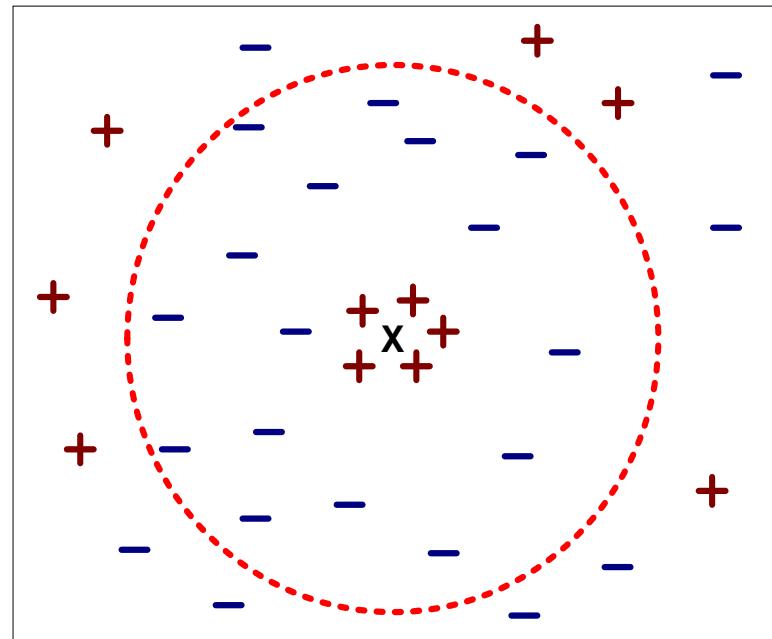
$$\boldsymbol{x}_i \cdot \boldsymbol{x}_j = \sum_{k=1}^d x_{ik} x_{jk}$$

$$\sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

$$L_2 \text{ norm of } \boldsymbol{x}_i: \|\boldsymbol{x}_i\|_2 = \sqrt{\sum_{k=1}^d x_{ik}^2}$$

# Value of $K$

- 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



# Determine Class Label

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the  $K$ -nearest neighbors
- Given test data  $\mathbf{x}^*$ , majority voting:

$$y^* = \arg \max_c \sum_{(x_i, y_i) \in \mathcal{N}_{\mathbf{x}^*}} I(c = y_i)$$

Indicator function that returns 1 if its input is true, otherwise 0

Nearest neighbors of the test instance  $\mathbf{x}^*$

- Every neighbor has the same impact on the classification
- This indeed makes the algorithm sensitive to the choice of  $K$

# Revised Voting Scheme

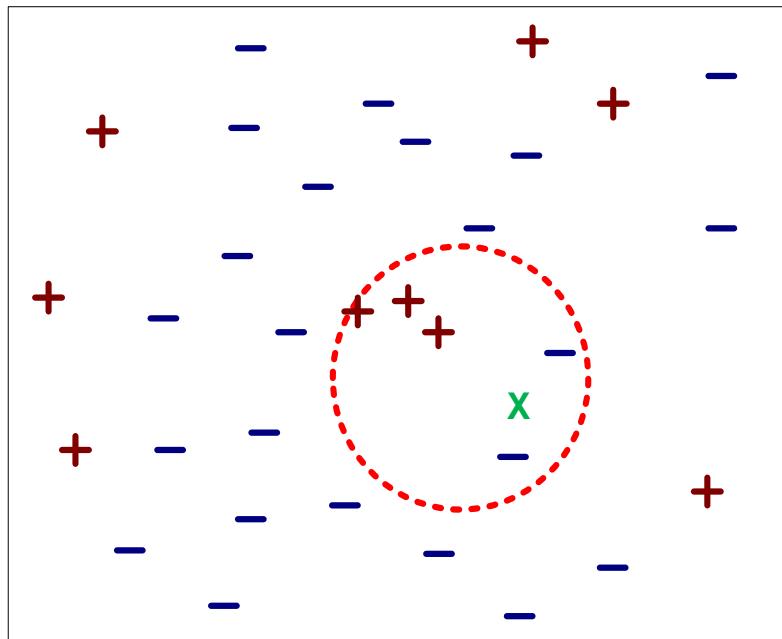
- Solution: distance-weighted voting
  - Weight the influence of each nearest neighbor  $\mathbf{x}_i$  according to its distance to the test instance  $\mathbf{x}^*$ :

$$w_i = \frac{1}{d(\mathbf{x}^*, \mathbf{x}_i)^2}$$

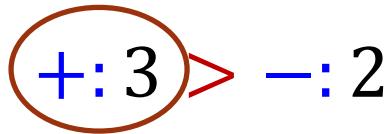
$$y^* = \arg \max_c \sum_{(\mathbf{x}_i, y_i) \in \mathcal{N}_{\mathbf{x}^*}} w_i \times I(c = y_i)$$

# Example

Consider a binary classification problem, and a 5-NN classifier



Training record	Class label	Distance to test instance
1	+	3
2	+	3.5
3	+	4
4	-	1.5
5	-	2

- Majority voting:  

- Distance-weighted voting:



Tutorial

# Other Issues

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2}$$

- Scaling issues
  - Feature may need to be scaled to prevent distance from being dominated by some features
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 40kg to 200kg
    - income of a person may vary from \$10K to \$1M
  - Solution: normalization on features of different scales

# Normalization

- Min-max normalization: to  $[min_{new}, max_{new}]$ 
  - Example: To normalize income ranging from \$12,000 to \$98,000 to  $[0.0, 1.0]$ , what is the value for \$73,600 after normalization?

$$v_{new} = \frac{v_{old} - min_{old}}{max_{old} - min_{old}} (max_{new} - min_{new}) + min_{new}$$

$$73,600 \quad \rightarrow \quad \frac{73600 - 12000}{98000 - 12000} (1.0 - 0) + 0 = 0.716$$

# Normalization (cont.)

- Standardization (z-score normalization) ( $\mu$ : mean,  $\sigma$ : standard deviation):

$$v_{new} = \frac{v_{old} - \mu_{old}}{\sigma_{old}} \quad \rightarrow \quad \mu_{new} = 0, \text{ and } \sigma_{new} = 1$$

- Example: Let  $\mu = 54,000$ ,  $\sigma = 16,000$ . What is the value for \$73,600 after standardization?

$$\frac{73600 - 54000}{16000} = 1.225$$

# Summary of NN Classifier

- The  $K$ -NN classifier is a lazy learner
  - It does not build models explicitly.
  - “Training” is very efficient.
  - Classifying unknown test instances is relatively expensive.

# Implementation

```
>>> from sklearn.neighbors import KNeighborsClassifier
```

...

set number of neighbors

```
>>> knnC = KNeighborsClassifier(n_neighbors=3)
```

```
>>> knnC.fit(X, y)
```

```
>>> pred = knnC.predict(X)
```

Build indices s.t. it is more efficient  
when making predictions on test data

# Thank you!

# CZ4041/SC4000: Machine Learning

## Lesson 7: Artificial Neural Networks

LI Boyang, Albert

School of Computer Science and Engineering,  
NTU, Singapore

# Instructors

- Weeks 7-12: LI Boyang, Albert
  - Nanyang Associate Professor, SCSE
  - Previously: Disney Research Pittsburgh, Baidu Research USA.
  - Email: [boyang.li@ntu.edu.sg](mailto:boyang.li@ntu.edu.sg)

# Arrangement

- Lectures
  - 2.30pm-4.30pm on Wednesdays, LT 23
  - Weeks 7-12
- Tutorials
  - 12.30-1.30pm on Mondays, LT 27
  - Week 8-9, 11-13 (Schedule posted on NTULearn)

# I have questions ...

- Find me after the lectures or the tutorials
- Send questions via email [boyang.li@ntu.edu.sg](mailto:boyang.li@ntu.edu.sg) or via Microsoft Teams
- Make an appointment

# Machine Learning

- Most of CCDS courses discuss computer systems, which are artificial in nature.
  - Programming Languages, Data Structures, Operating Systems, Databases, Compilers, etc.
- Most natural sciences study the objective reality and various types of phenomena within.
  - Physics, Chemistry, Biology, etc.
- Machine Learning is where computer systems meet the objective world; a subject where we use the former to model the latter.

# ML Requires Math

Philosophy is written in this grand book—I mean the universe—which stands continually open to our gaze, but it cannot be understood unless one first learns to comprehend the language and interpret the characters in which it is written. It is written in the language of mathematics, and its characters are triangles, circles, and other mathematical figures, without which it is humanly impossible to understand a single word of it; without these one is wandering about in a dark labyrinth.

**Galileo Galilei** *Il Saggiatore* [1623]

# ML Requires Math

The language of mathematics reveals itself unreasonably effective in the natural sciences..., a wonderful gift which we neither understand nor deserve. We should be grateful for it and hope that it will remain valid in future research and that it will extend, for better or for worse, to our pleasure even though perhaps to our bafflement, to wide branches of learning.

Eugene Wigner [1960]

# Typical ML Approach

- Specify (explicitly or implicitly) a family of functions that contain some free parameters (e.g.,  $a$ ,  $b$ , and  $c$  below).

$$y = ax^2 + bx + c$$

- Determine the values of these free parameters from data.

# Purposes of Machine Learning

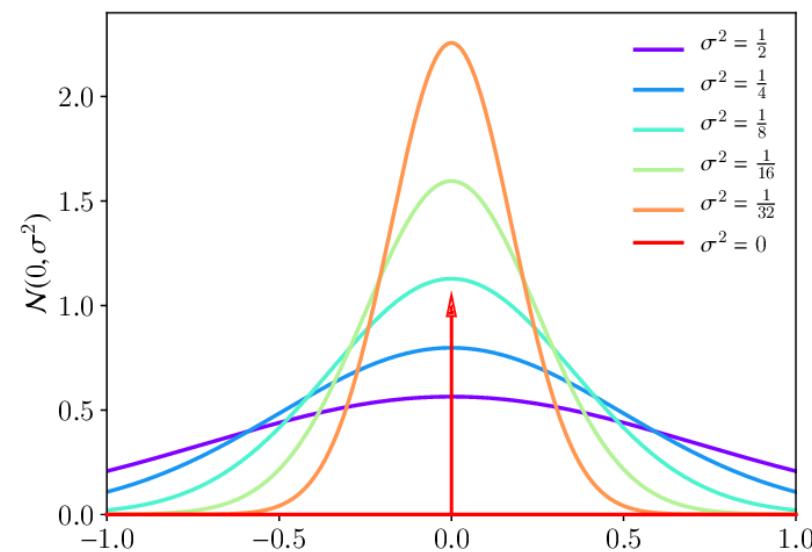
- Make Accurate Predictions
  - Which team will win a soccer match?
  - Which stock will see its price skyrocket?
  - Which patient is at higher risk?
- Usually it is difficult to write down rules manually
- Rather, we learn to make the predictions from paired data  $(x_i, y_i)$

# Purposes of Machine Learning

- Make Accurate Predictions
  - Artificial Neural Networks (Week 7)
  - Support Vector Machines (Week 8)
  - Regression (Week 8)
  - Ensemble Learning (Week 9)

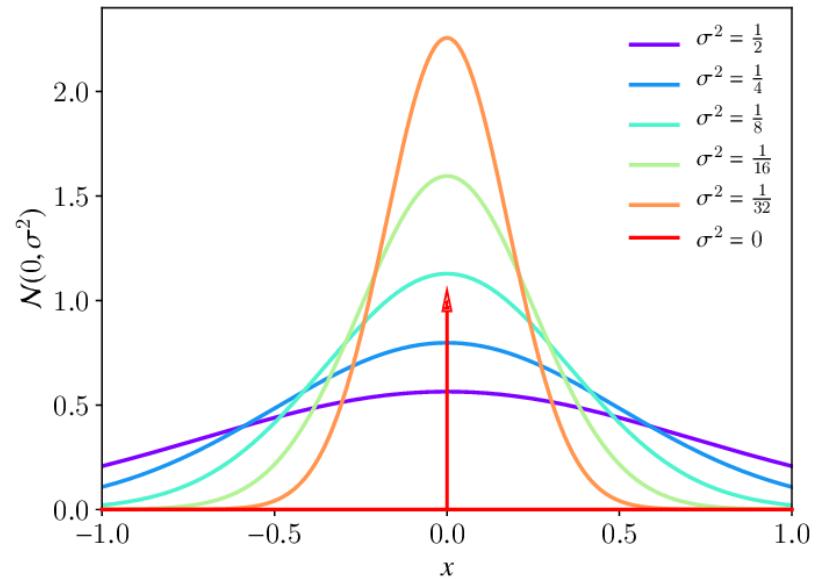
# Purposes of Machine Learning

- Build Probabilistic Models of the World
  - Germany will probably beat Japan, but what are the odds? 60/40, 70/30, or 80/20?
  - I'm willing to bet more money if the odds are in my favor.
  - Often translates to: what is the shape of the probability distribution?



# Purposes of Machine Learning

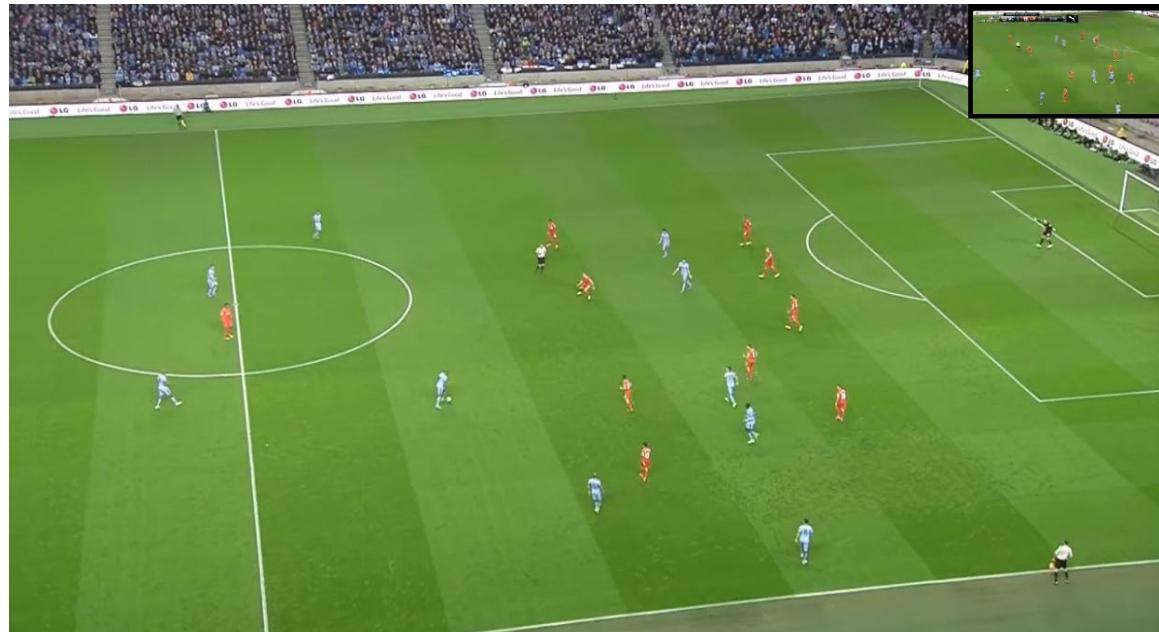
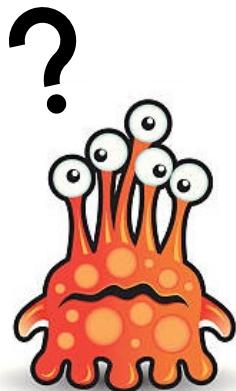
- Build Probabilistic Models of the World
  - Density Estimation (Week 11)



# Purposes of Machine Learning

## ➤ Pattern Discovery

- Imagine you are an alien from another planet. You watch a soccer match. What do you observe?
- Two groups of humans. One ball.
- Behavior change when the ball goes into the net.



# Purposes of Machine Learning

- Pattern Discovery
  - We often have little prior experience, knowledge, or insight into the causal mechanisms that generated the data.
  - Still, with only statistical tools, we can identify many important data characteristics
  - Obviously, domain knowledge can enrich and complement statistical tools.

# Purposes of Machine Learning

- Pattern Discovery
- Clustering (Week 10)
- Dimensionality Reduction (Week 12)

# **Artificial Neural Networks: Perceptron**

# Artificial Neural Networks (ANN)

- The study of ANN was inspired by biological neural systems



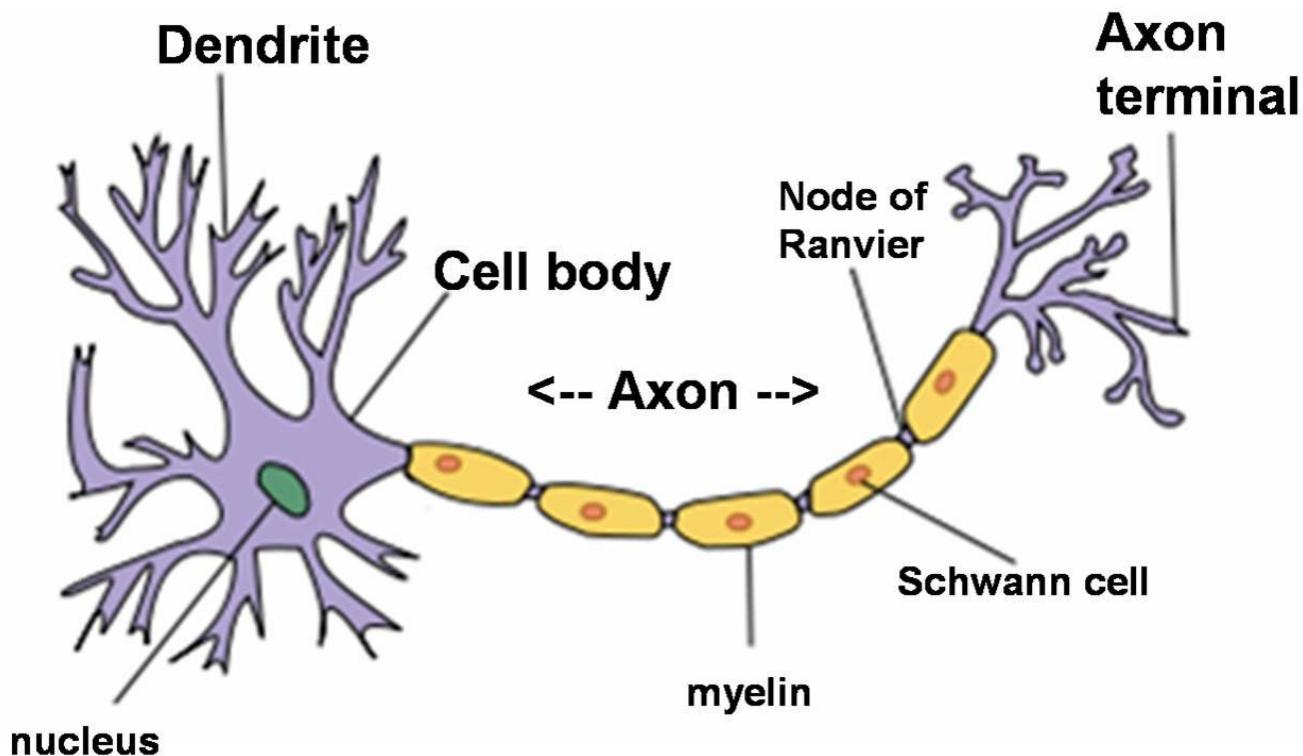
Cat      Dog



# “Biology”

- Human brain is a densely interconnected network of neurons, connected to others via dendrites and axons.
- Dendrites and axons transmit electrical signals from one neuron to another
- The human brain learns by changing the strength of the synaptic connection between neurons
- An ANN is composed of an interconnected assembly of nodes and directed links.

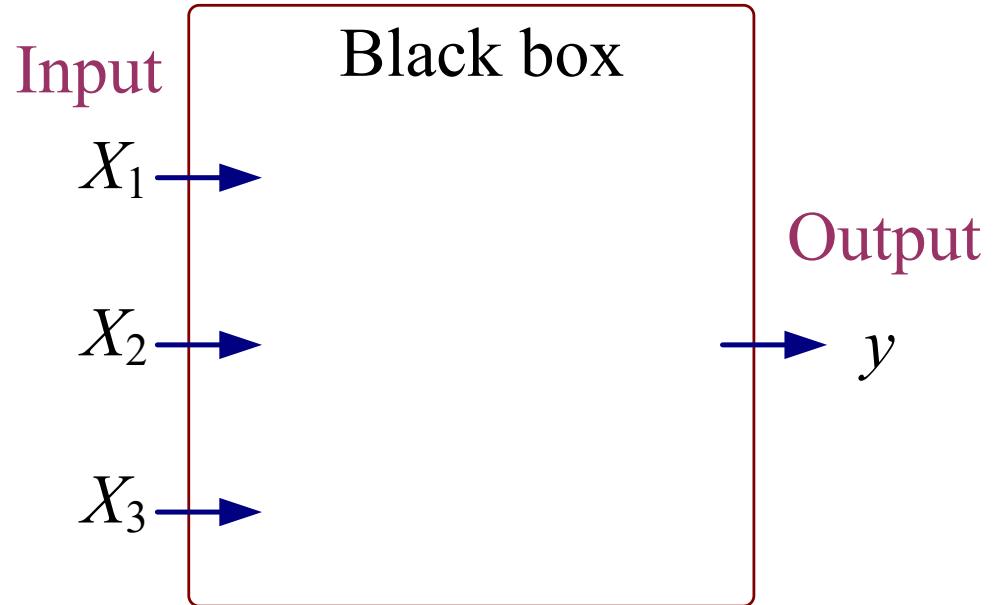
# “Biology”



- A neuron sends out a spike from the axon after receiving enough input from the dendrites.

# Artificial Neural Networks (cont.)

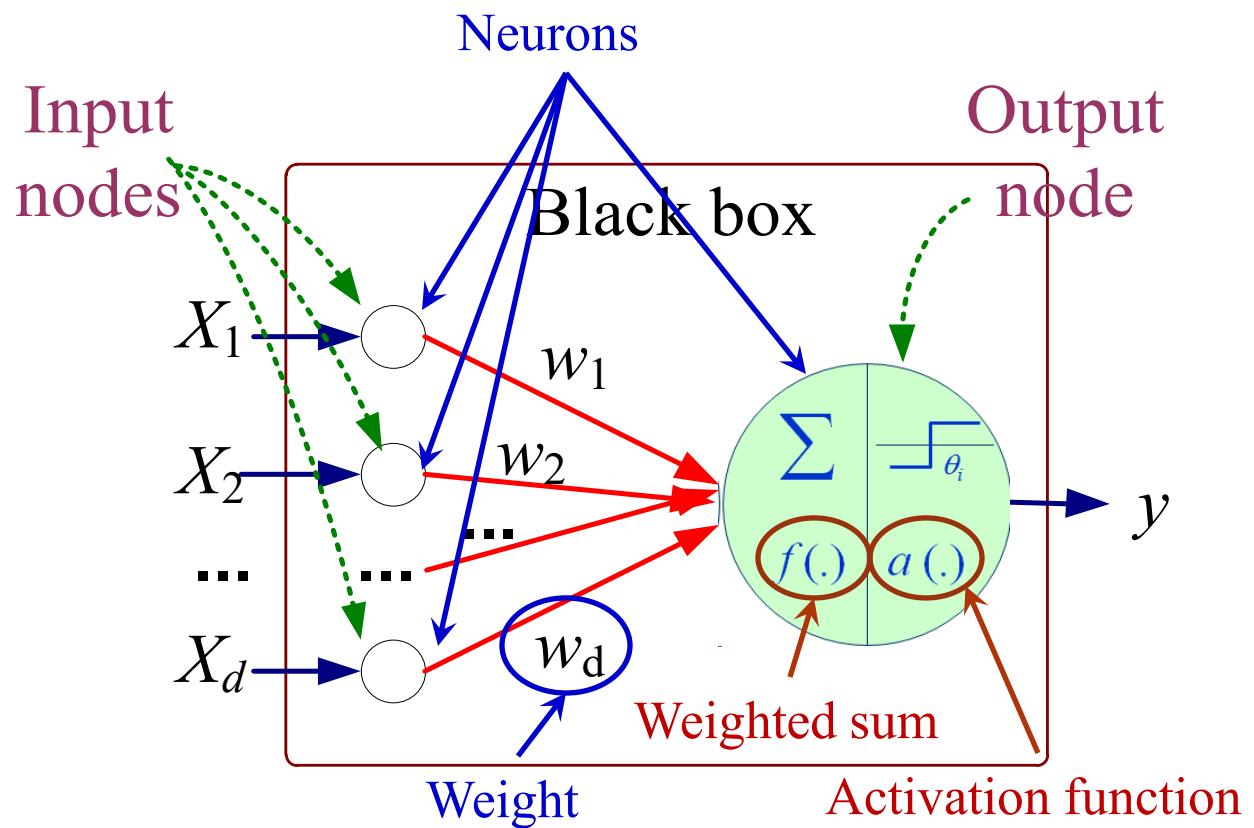
$X_1$	$X_2$	$X_3$	$y$
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output  $y$  is 1 if at least two of the three inputs are equal to 1

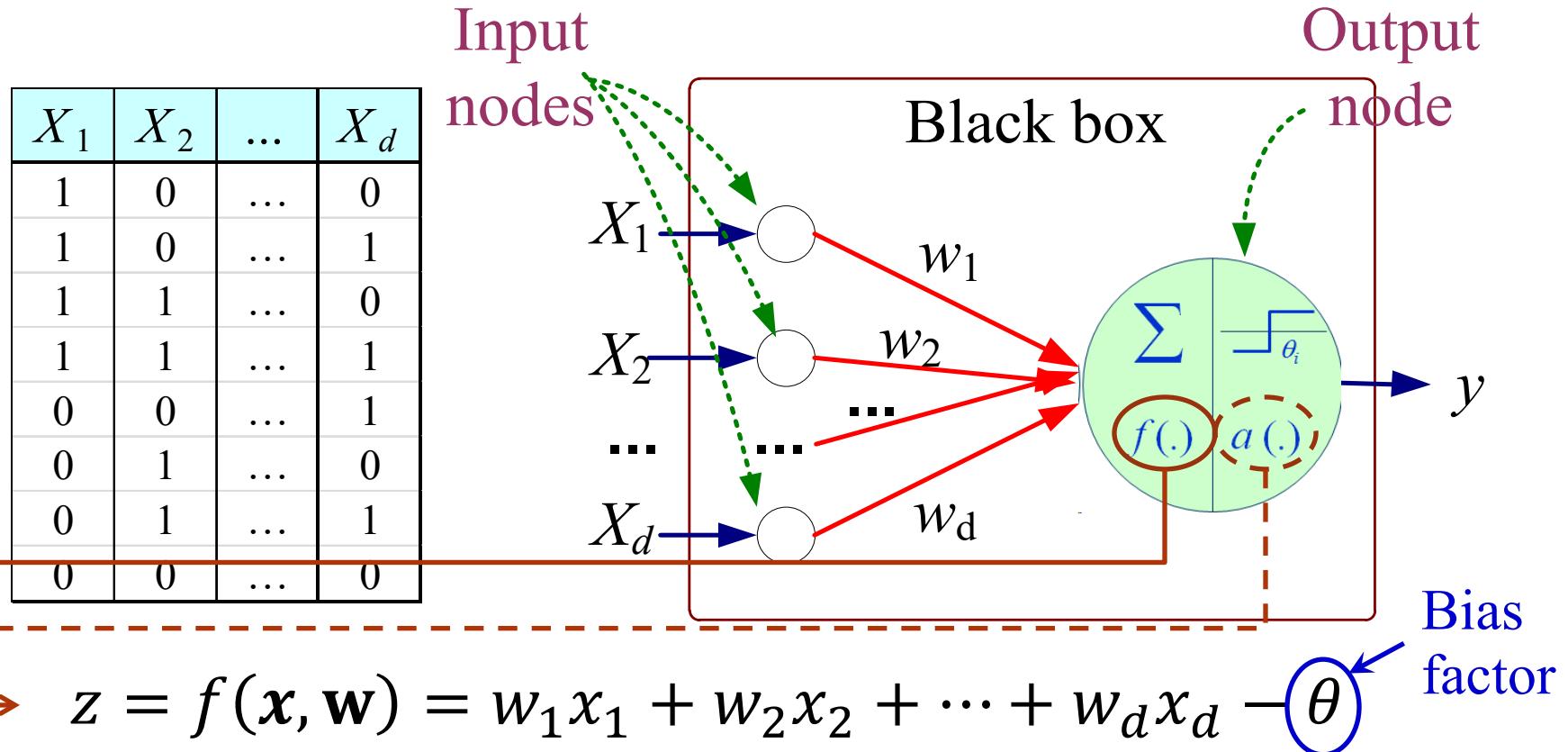
# ANN: Perceptron

$X_1$	$X_2$	...	$X_d$
1	0	...	0
1	0	...	1
1	1	...	0
1	1	...	1
0	0	...	1
0	1	...	0
0	1	...	1
0	0	...	0

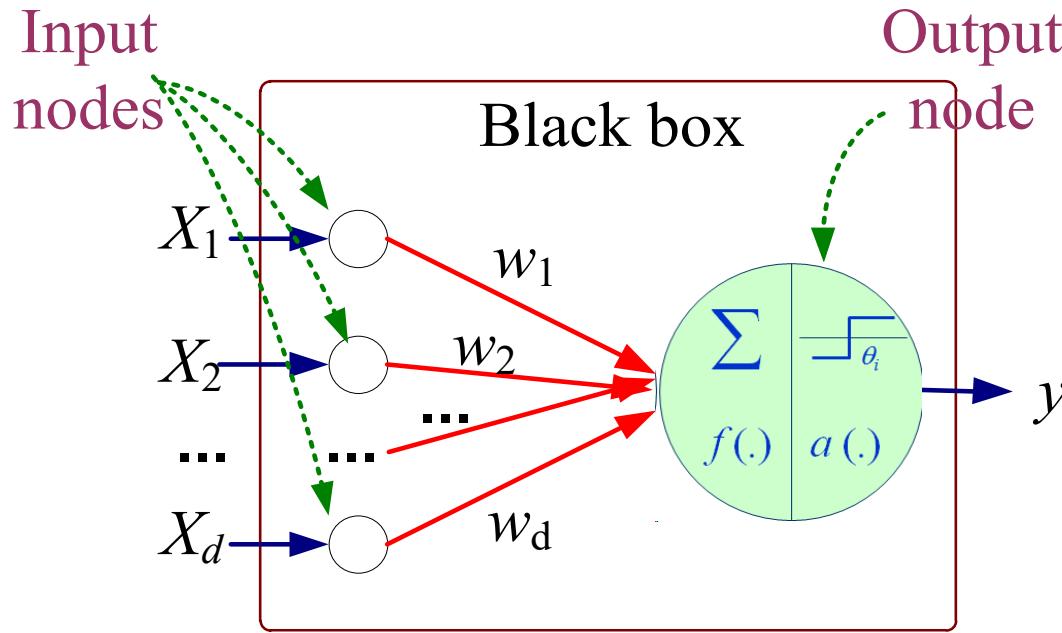


Each input node is connected via a weighted link to the output node.  
Weights can be positive, negative or zero (no connection)

# ANN: Perceptron (cont.)

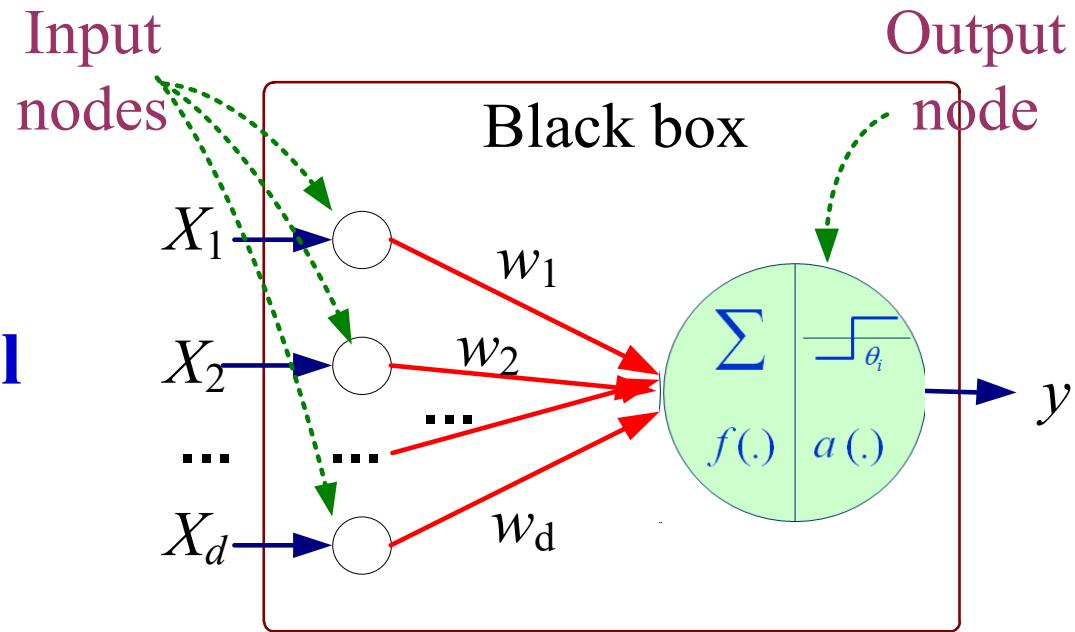


→  $y = a(z)$ , where  $a(z) = \text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & \text{otherwise} \end{cases}$



- Model is an assembly of inter-connected nodes and weighted links
- Output node first sums up each of its input value according to the weights of its links
- Compare the weighted sum against some threshold  $\theta$
- Produce an output based on the sign of the result

## Perceptron Model



$$z = \sum_{i=1}^d w_i x_i - \theta \longrightarrow y = a(z) = \text{sign}(z)$$

$$y = \text{sign} \left( \sum_{i=1}^d w_i x_i - \theta \right)$$

# ANN: Perceptron (cont.)

- Mathematically, the output of a perceptron model can be expressed in a more compact form

$$y = \text{sign} \left( \sum_{i=1}^d w_i x_i - \theta \right)$$

↓

$$y = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

Inner product

$$\text{where } \mathbf{x} = (x_0, x_1, x_2, \dots, x_d)$$

$$\mathbf{w} = (w_0, w_1, w_2, \dots, w_d)$$

$$w_0 = -\theta, \text{ and } x_0 = 1$$

# Inner Product: Review

- Given two vectors  $\mathbf{x}$  and  $\mathbf{z}$ , which are both of  $d$  dimensions, the inner product between  $\mathbf{x}$  and  $\mathbf{z}$  is defined as

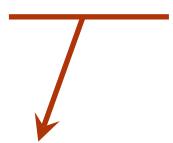
$$\mathbf{x} \cdot \mathbf{z} = \sum_{i=1}^d x_i z_i$$

$$\mathbf{x} = (x_1, x_2, \dots, x_d) \quad \mathbf{z} = (z_1, z_2, \dots, z_d)$$

# ANN: Perceptron (cont.)

$$y = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

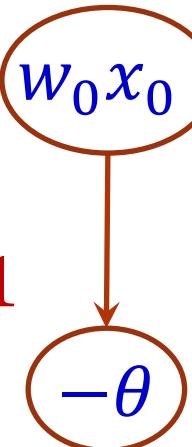
$$\mathbf{x} = (x_0, x_1, x_2, \dots, x_d)$$



$$\mathbf{w} = (w_0, w_1, w_2, \dots, w_d)$$

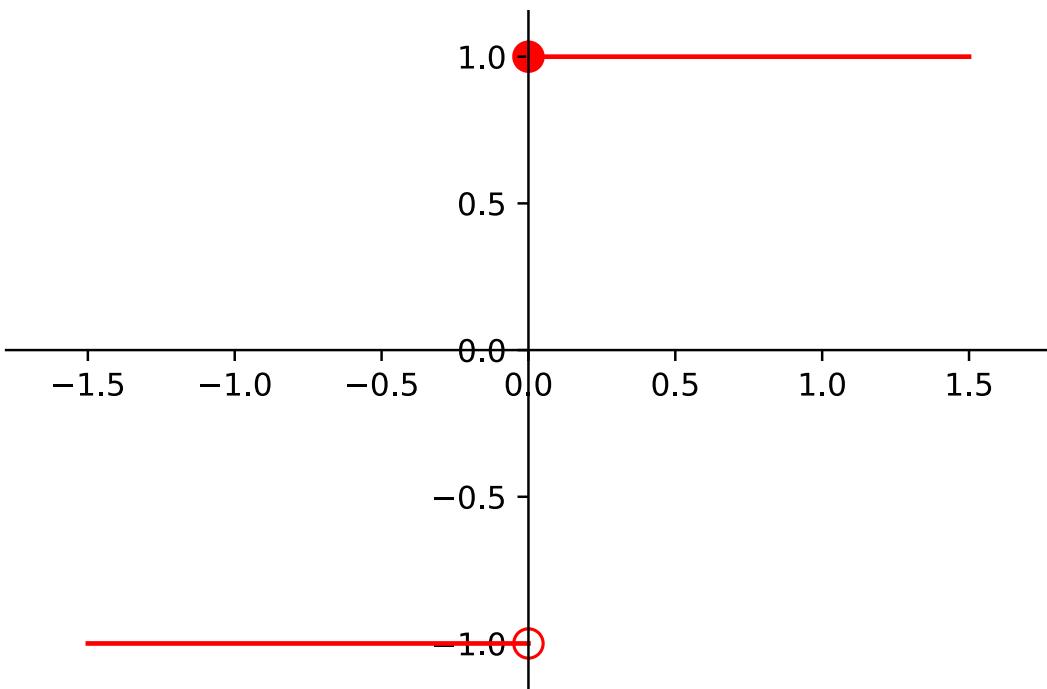
$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=0}^d (w_i x_i) = \sum_{i=1}^d (w_i x_i) + w_0 x_0$$

$$w_0 = -\theta, \text{ and } x_0 = 1$$



$$y = \text{sign} \left( \sum_{i=1}^d w_i x_i - \theta \right) \longleftrightarrow y = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$

# ANN: Sign Function



$$\text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$

Note: the  $\text{sign}(z)$  function has derivative = 0 everywhere, except at  $z = 0$ .

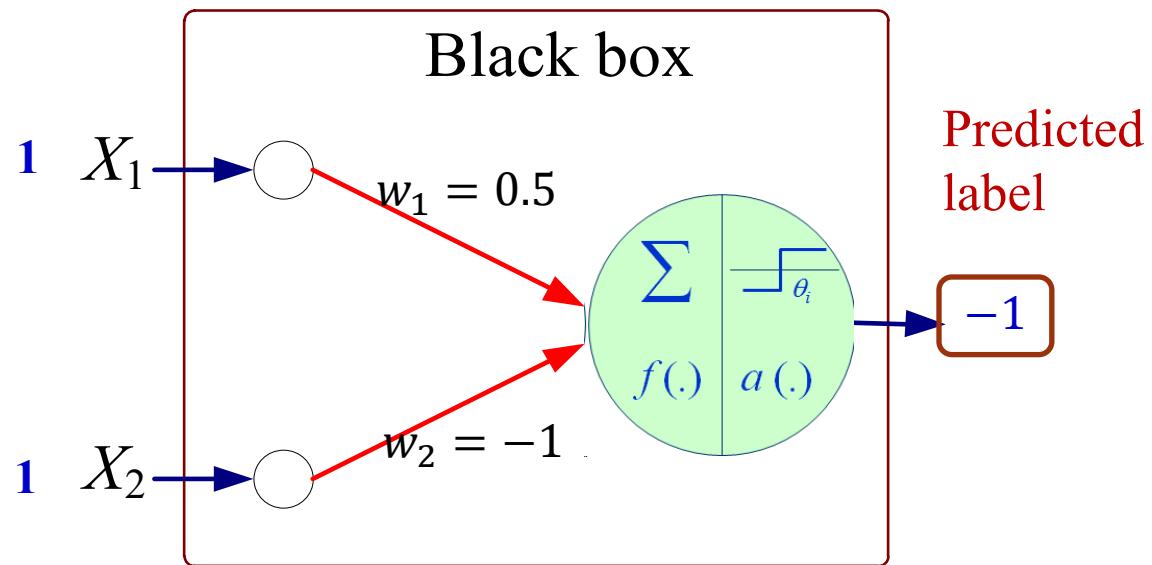
# Perceptron: Making Prediction

- Given a learned perceptron with  $w_1 = 0.5$ ,  $w_2 = -1$ , and  $\theta = 0$

Test data:

$x$

	$X_1$	$X_2$
$X_1$	1	1
$X_2$		

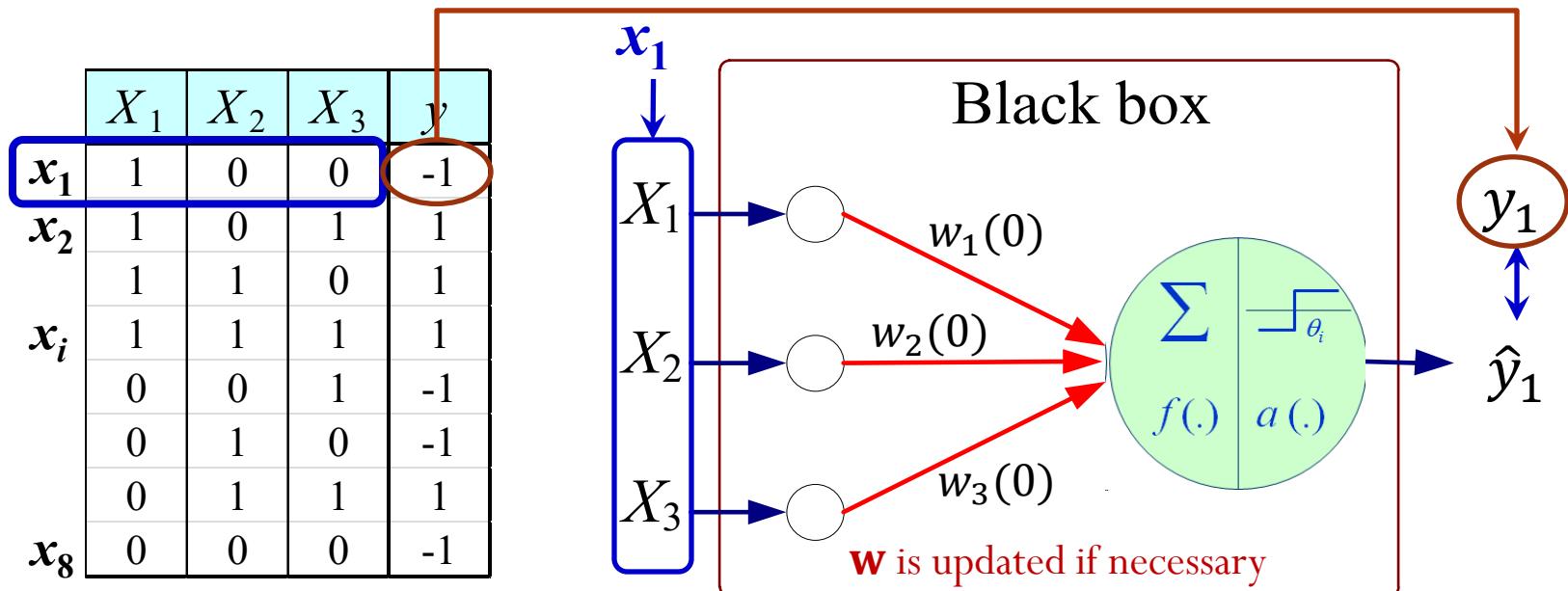


$$y = \text{sign}(1 \times 0.5 + 1 \times (-1))$$

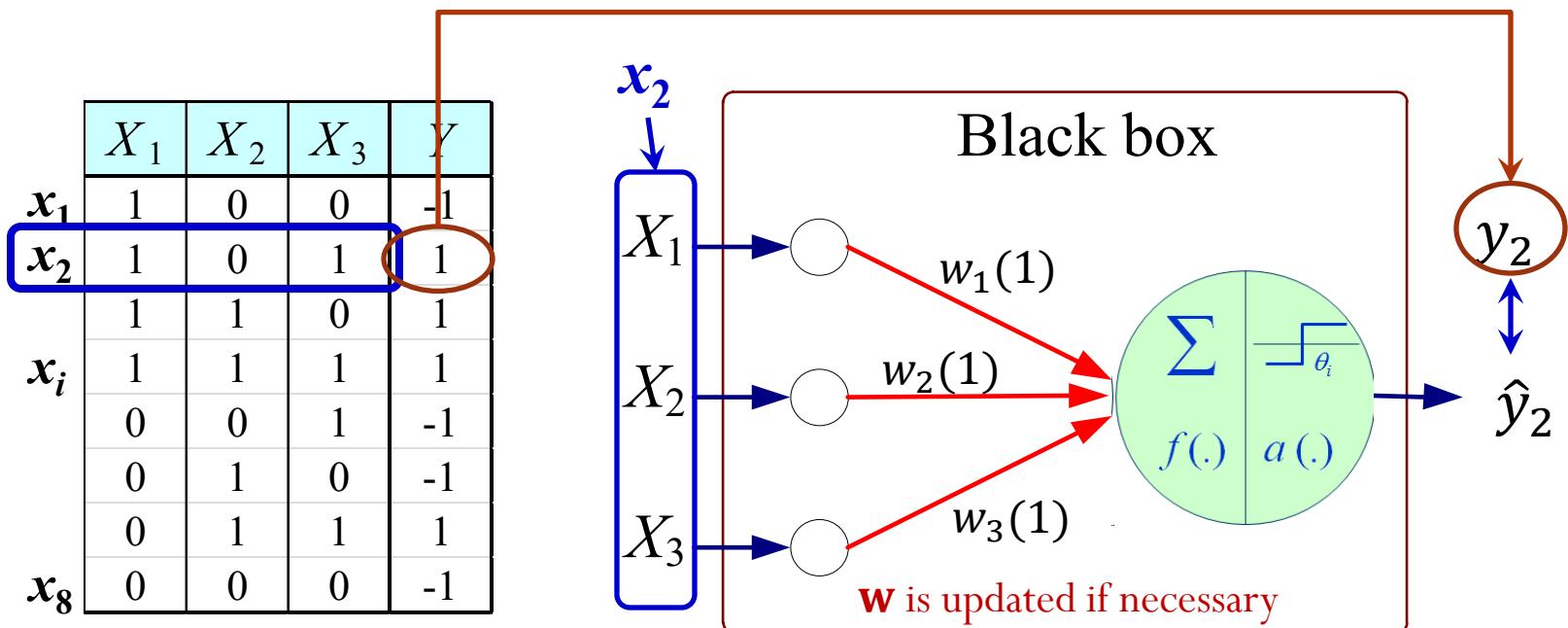
$$= \text{sign}(-0.5) = -1$$

# Perceptron: Learning

- During training, the weight parameters  $\mathbf{w}$  are adjusted until the outputs of the perceptron become consistent with the true outputs of training data
- The weight parameters  $\mathbf{w}$  are updated iteratively or in an online learning manner



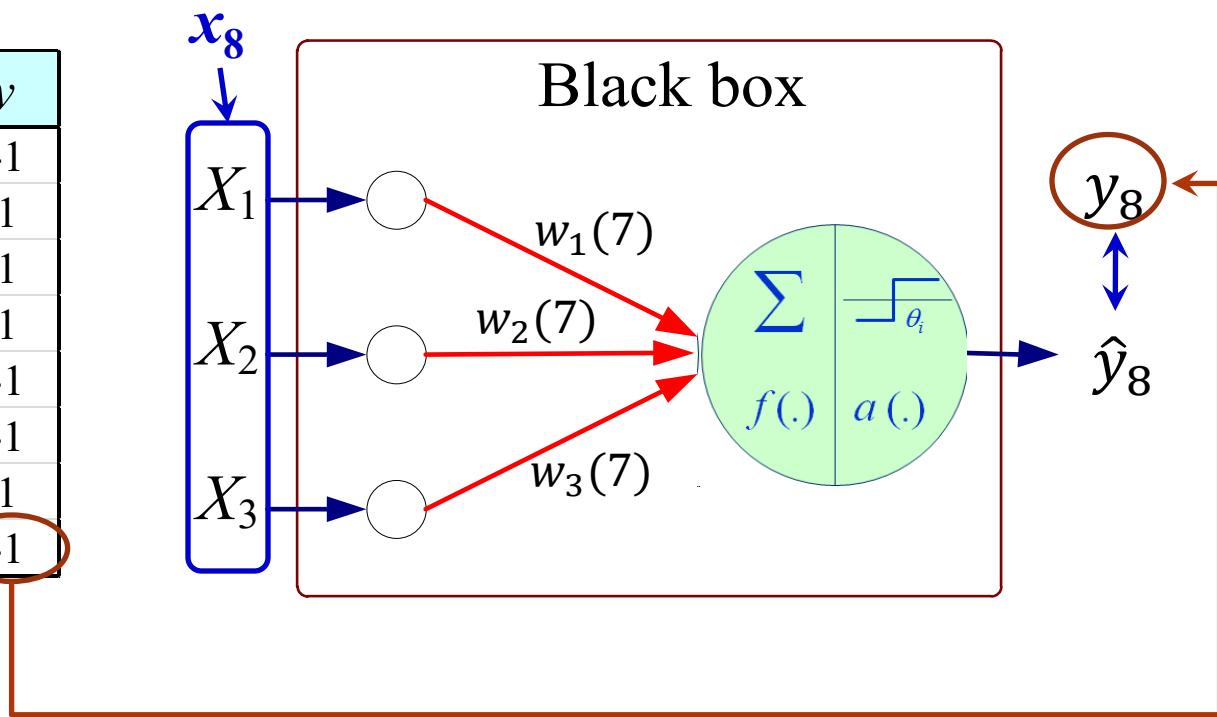
# Perceptron: Learning (cont.)



● ● ●

# Perceptron: Learning (cont.)

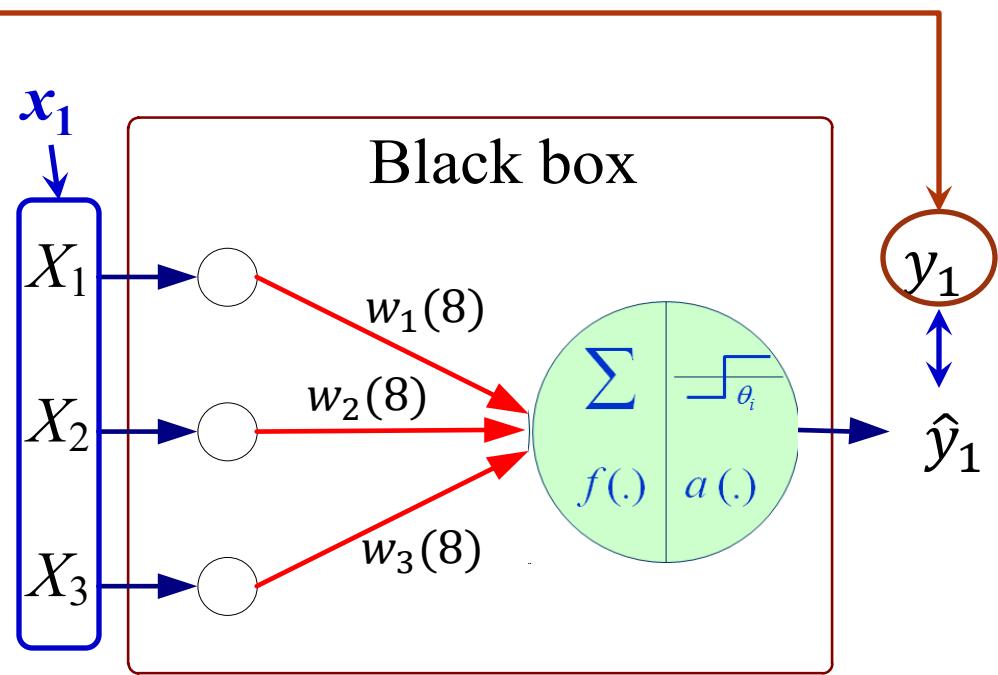
	$X_1$	$X_2$	$X_3$	$y$
$x_1$	1	0	0	-1
$x_2$	1	0	1	1
$x_i$	1	1	0	1
$x_6$	1	1	1	1
$x_7$	0	0	1	-1
$x_8$	0	1	0	-1
$x_9$	0	1	1	1
$x_8$	0	0	0	-1



The 2<sup>nd</sup> Epoch starts

# Perceptron: Learning (cont.)

	$X_1$	$X_2$	$X_3$	$y$
$x_1$	1	0	0	-1
$x_2$	1	0	1	1
$x_i$	1	1	0	1
	1	1	1	1
$x_6$	0	0	1	-1
	0	1	0	-1
$x_8$	0	1	1	1
	0	0	0	-1



● ● ●

# Perceptron: Learning (cont.)

- Algorithm:
  - Let  $D = \{(x_i, y_i) \mid i = 1, 2, \dots, N\}$  be the set of training examples,  $t = 0$
  - Initialize  $\mathbf{w}$  with random values  $\mathbf{w}_0$
  - Repeat**
  - for each training example  $(x_i, y_i)$  do
    - Compute the predicted output  $\hat{y}_i$
    - Update  $\mathbf{w}_t$  by  $\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)x_i$
  - $t = t + 1$
  - end for**
  - Until stopping condition is met

# Perceptron: Learning (cont.)

- Why use the following weight update rule?

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\mathbf{x}_i$$

- Induced based on a gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$

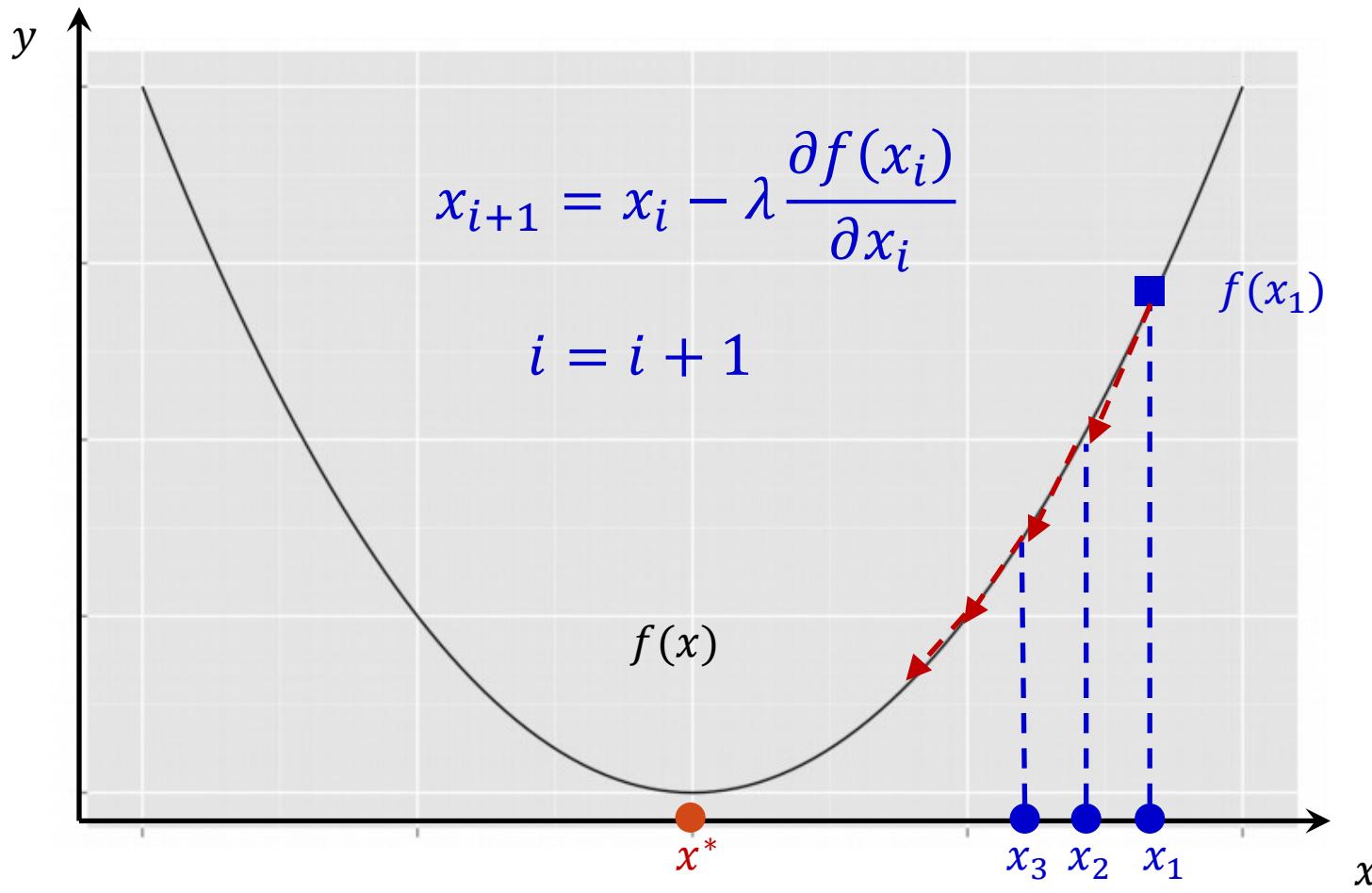
Function to be minimized, e.g., the error loss function

Learning rate  $\lambda \in (0,1]$

The diagram illustrates the gradient descent update rule. It shows the weight vector  $\mathbf{w}$  being updated from  $\mathbf{w}_t$  to  $\mathbf{w}_{t+1}$  by subtracting a scaled gradient. The scale factor is the learning rate  $\lambda$ . The gradient is represented by the term  $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$ , which is highlighted with a red box. A red bracket under the learning rate  $\lambda$  indicates its range. To the right, a red box contains the text "Function to be minimized, e.g., the error loss function". Below the equation, another red box contains the text "Learning rate  $\lambda \in (0,1]$ ". Red arrows point from the text "Function to be minimized..." to the red box around the gradient term and from "Learning rate λ ∈ (0,1]" to the red bracket under λ.

# Gradient Descent

$$x^* = \arg \min_x f(x)$$



# Perceptron: Learning (cont.)

- Weight update rule



$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\mathbf{x}_i \quad \leftarrow \quad \mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$

- Consider the loss function  $\mathcal{L}$  for each training example as  $e_i \triangleq y_i - \hat{y}_i$   $\hat{y}_i = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_i)$

$$\mathcal{L} = \frac{1}{2}e_i^2 = \frac{1}{2}(y_i - \hat{y}_i)^2 = \frac{1}{2}(y_i - \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_i))^2$$

- Update the weight using a gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}(z)}{\partial z} \frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}$$

$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2 \quad \hat{y} = \text{sign}(z) \quad z = \mathbf{w} \cdot \mathbf{x}$

Chain rule

# Chain Rule of Calculus (Review)

- Suppose that  $y = g(x)$  and  $z = f(y) = f(g(x))$
- Chain rule of calculus:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

- Generalized to the vector case: suppose  $\mathbf{x} \in R^m$ ,  $\mathbf{y} \in R^n$ ,  $\mathbf{y} = g(\mathbf{x})$  and  $z = f(\mathbf{y})$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

# Perceptron: Learning (cont.)

- The weight update formula for perceptron:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}(z)}{\partial z} \frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}$$

$\mathcal{L} = \frac{1}{2} (y_i - \hat{y}_i)^2$

$\hat{y}_i = \text{sign}(z_i)$

$\hat{y}_i = z_i$

$1$

$Z_i = \mathbf{w}_t \cdot \mathbf{x}_i$

$\mathbf{x}_i$

$(y_i - \hat{y}_i) \times -1$

$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\mathbf{x}_i$

Note: the  $\text{sign}()$  function has derivative = 0 except at  $z = 0$ . Therefore, to compute meaningful derivative, we remove  $\text{sign}()$  from consideration

# Approximating the derivative?

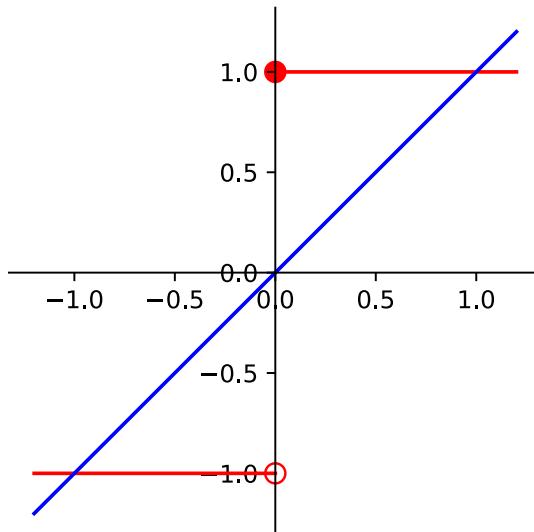
- The equation used to compute  $\hat{y}_i$  from  $z_i$

$$\hat{y}_i = \text{sign}(z_i)$$

- The equation used to compute  $\frac{\partial \hat{y}(z)}{\partial z}$

$$\hat{y}_i = z_i$$

- Why? This is approximating the step function with a linear function.



While the derivative itself is incorrect,  
its direction is correct.

That is, if you want to increase  $\hat{y}_i$ ,  
you should increase  $z_i$ , and vice  
versa.

# Perceptron Weights Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\mathbf{x}_i$$

- If the prediction is correct,  $(y - \hat{y}) = 0$ , then weight remains unchanged  $\mathbf{w}_{t+1} = \mathbf{w}_t$
- If  $y = +1$  and  $\hat{y} = -1$ , then  $(y - \hat{y}) = 2$
- The weights of all links with positive inputs need to be updated by increasing their values
- The weights of all links with negative inputs need to be updated by decreasing their weights

$\mathbf{x}_i$	$x_{i1}$	...	$x_{ik}$	...	$x_{id}$
	$> 0$		$= 0$		$< 0$

$\mathbf{w}_{t+1}$	$w_1$ ↑	...	$w_k$	...	$w_d$ ↓
--------------------	---------	-----	-------	-----	---------

# Perceptron Weights Update (cont.)

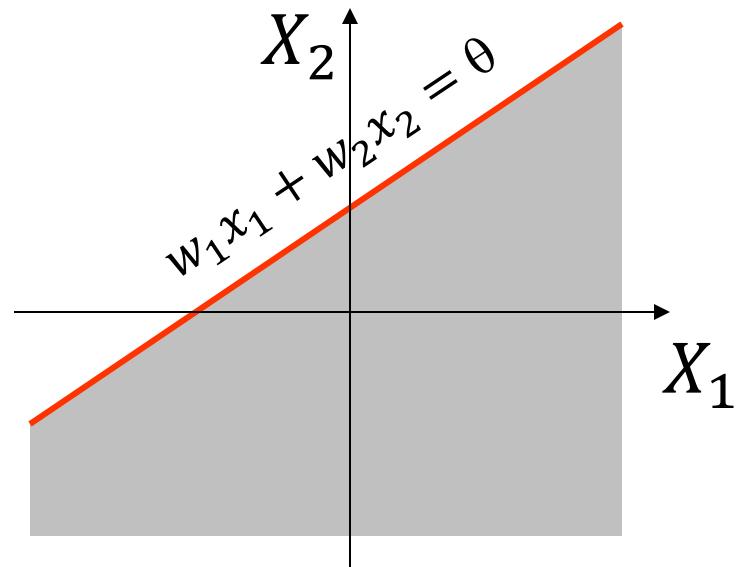
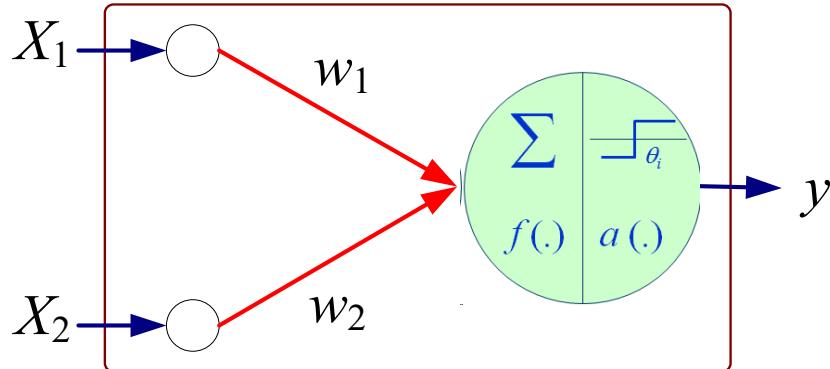
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\mathbf{x}_i$$

- If  $y = -1$  and  $\hat{y} = +1$ , then  $(y - \hat{y}) = -2$
- The weights of all links with positive inputs need to be updated by decreasing their values
- The weights of all links with negative inputs need to be updated by increasing their weights

$\mathbf{x}_i$	$X_{i1}$	...	$X_{ik}$	...	$X_{id}$
	$> 0$		$= 0$		$< 0$
$\mathbf{w}_{t+1}$	$w_1$ ↓	...	$w_k$	...	$w_d$ ↑

# Convergence

- The decision boundary of a perceptron is a linear hyperplane



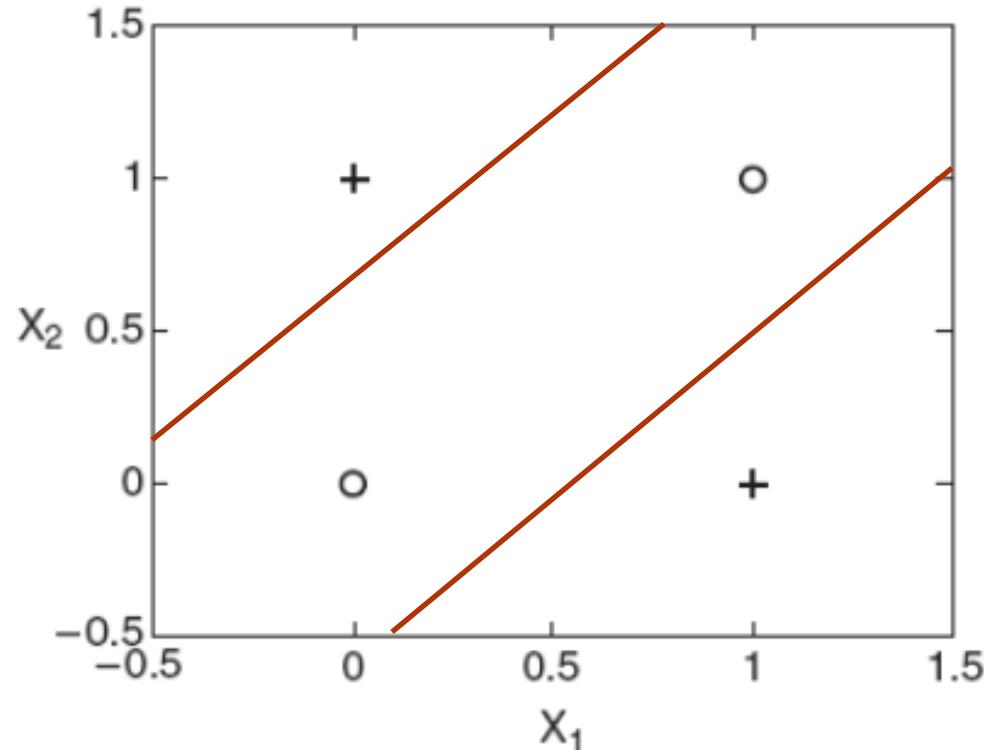
- The perceptron learning algorithm is guaranteed to converge to an optimal solution for linear classification problems

# Perceptron Limitation

- If the problem is not linearly separable, the algorithm fails to converge

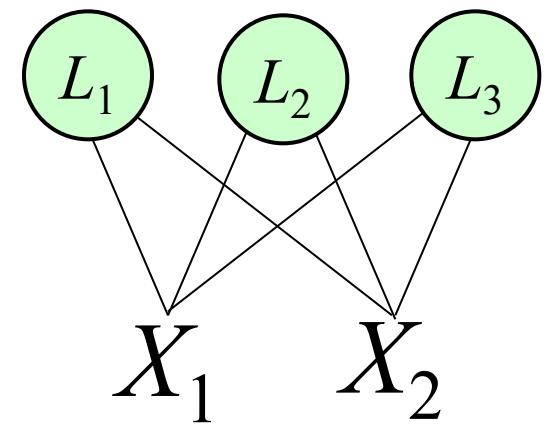
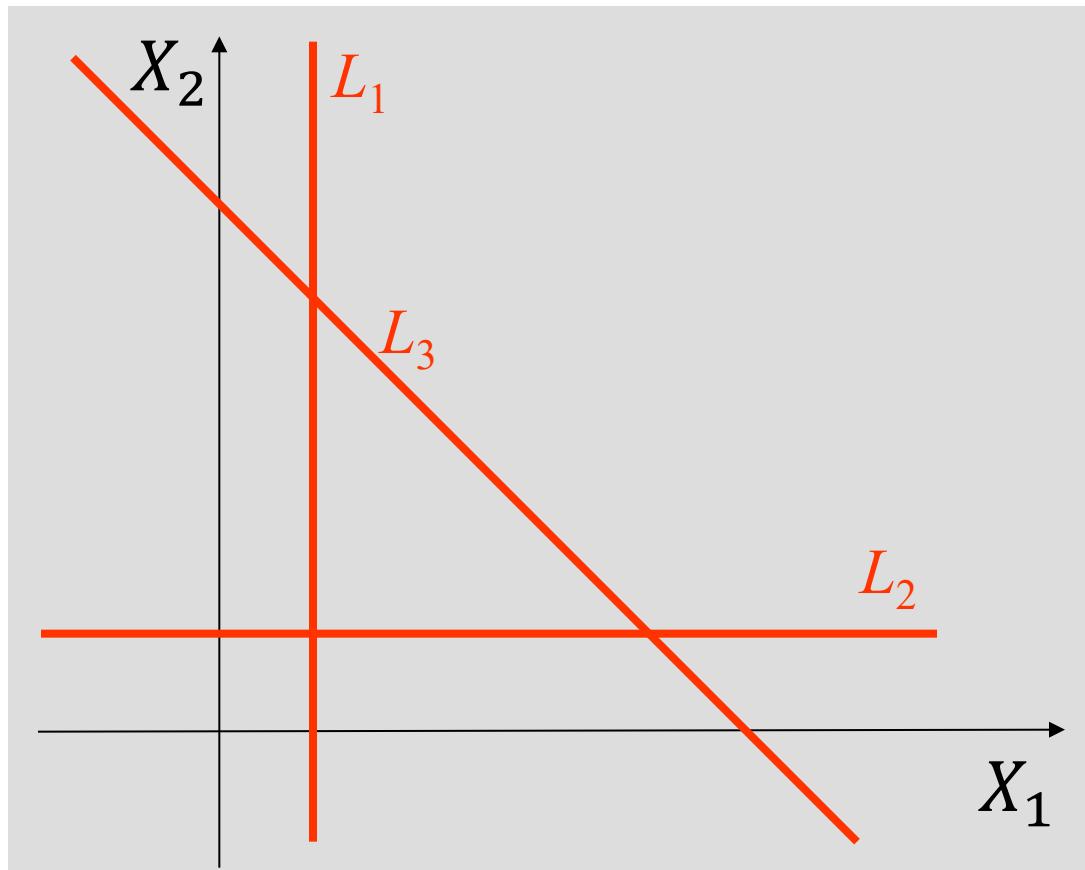
Nonlinearly separable data given by the XOR function

$X_1$	$X_2$	$y$
0	0	-1
1	0	1
0	1	1
1	1	-1

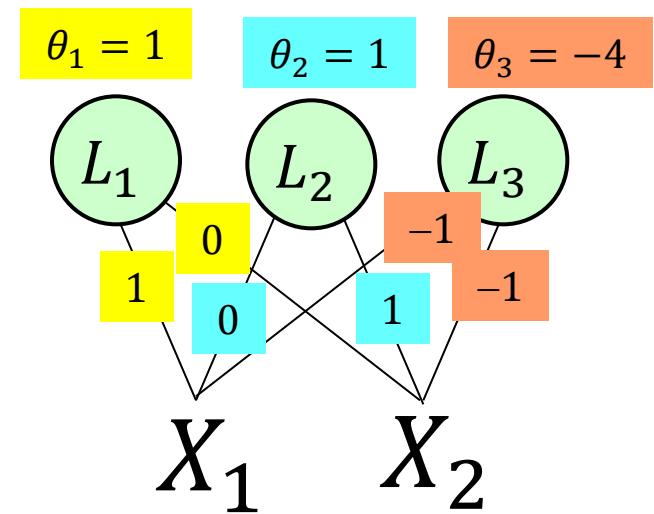
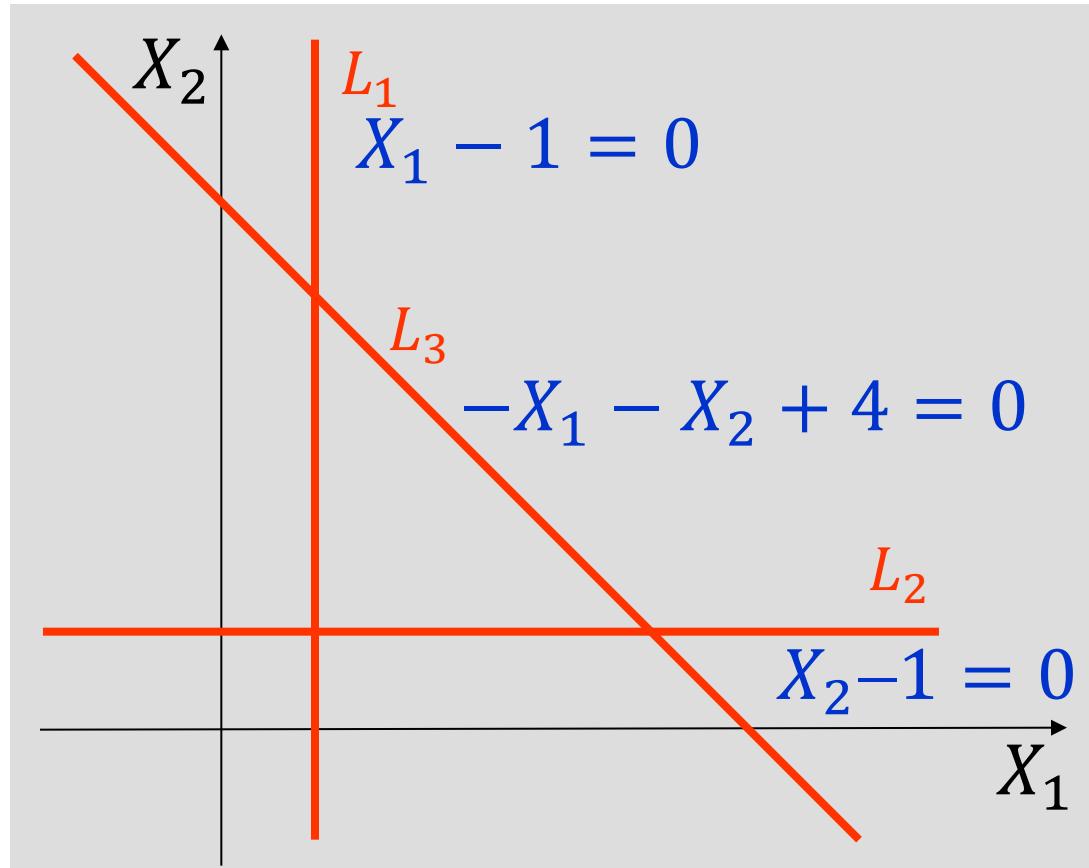


# Multi-layer Perceptron

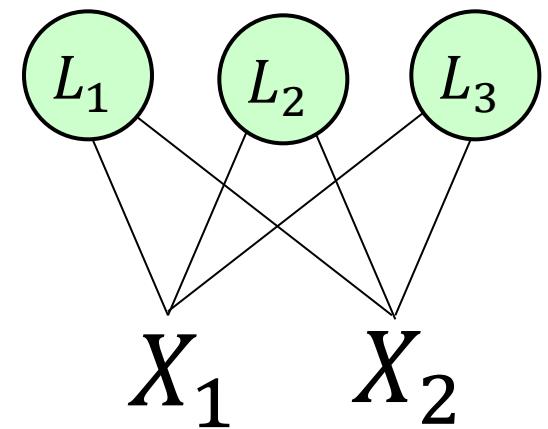
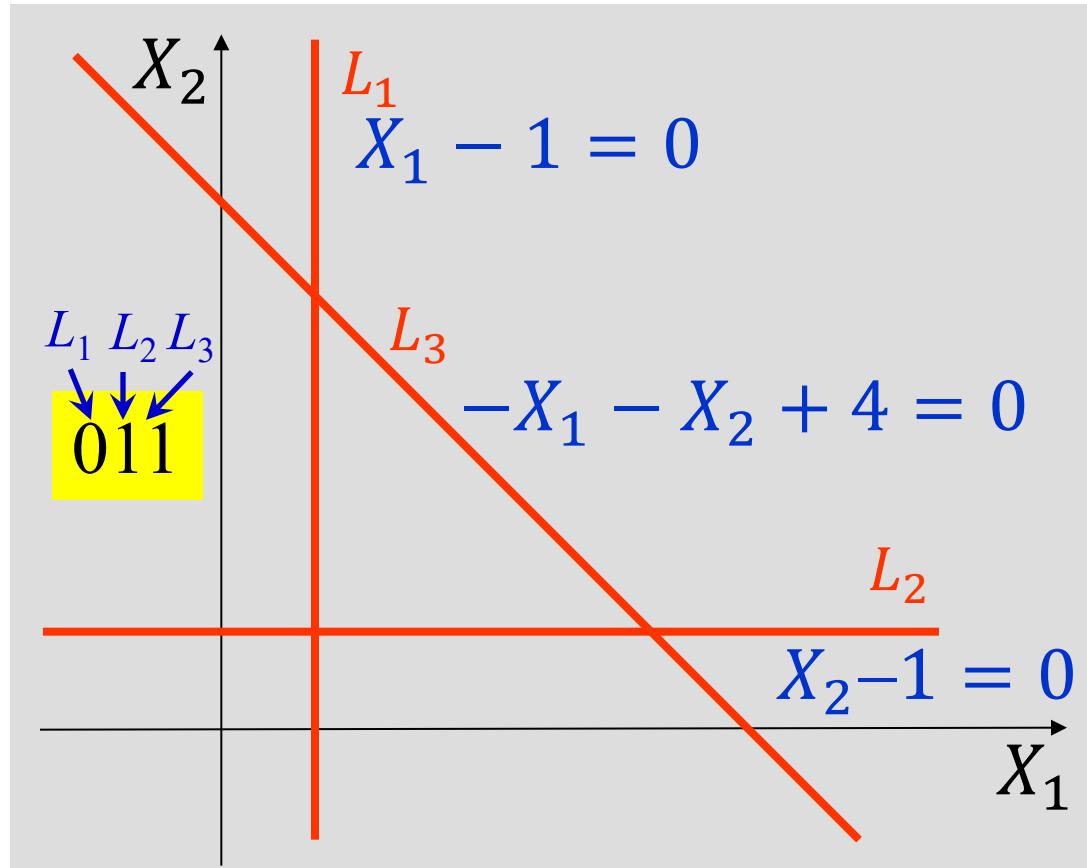
# Example: Not Linearly Separable



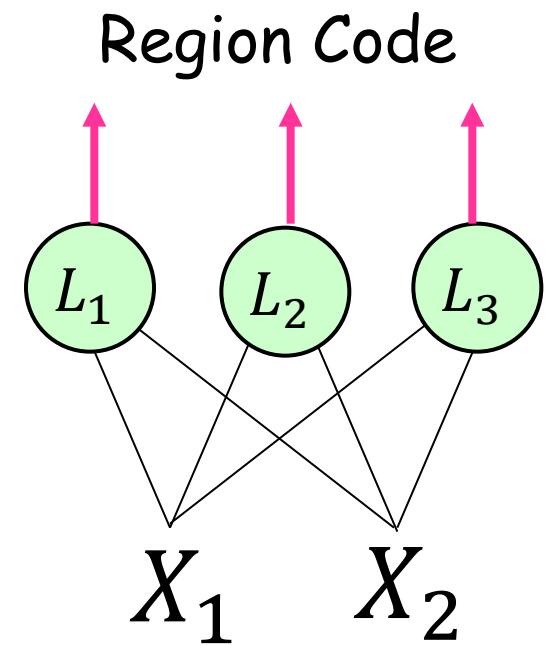
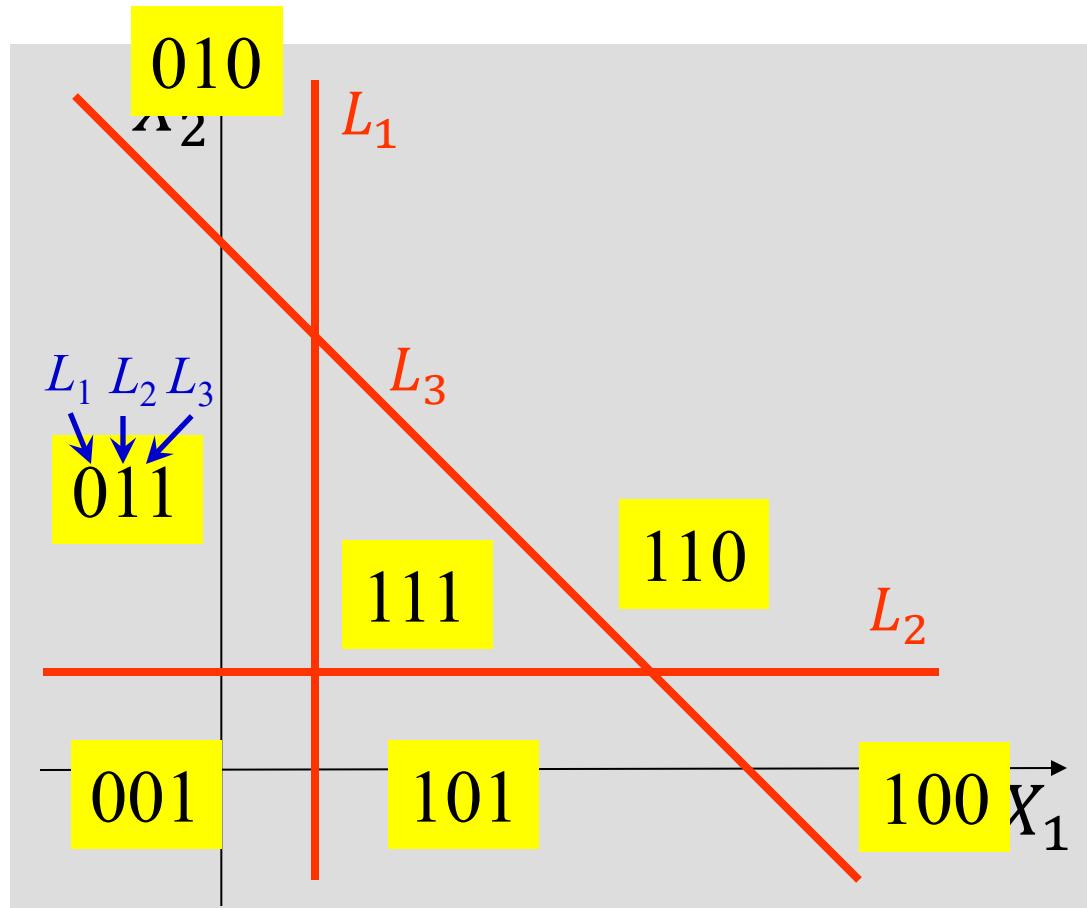
# Example: Not Linearly Separable



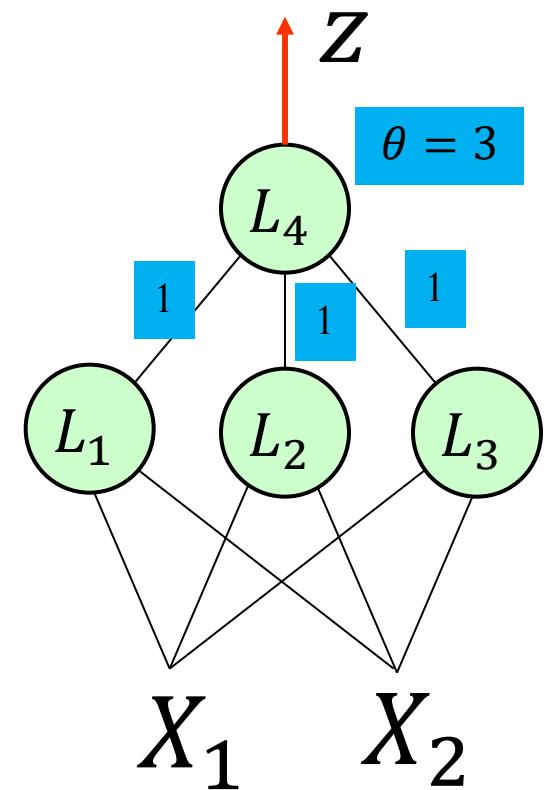
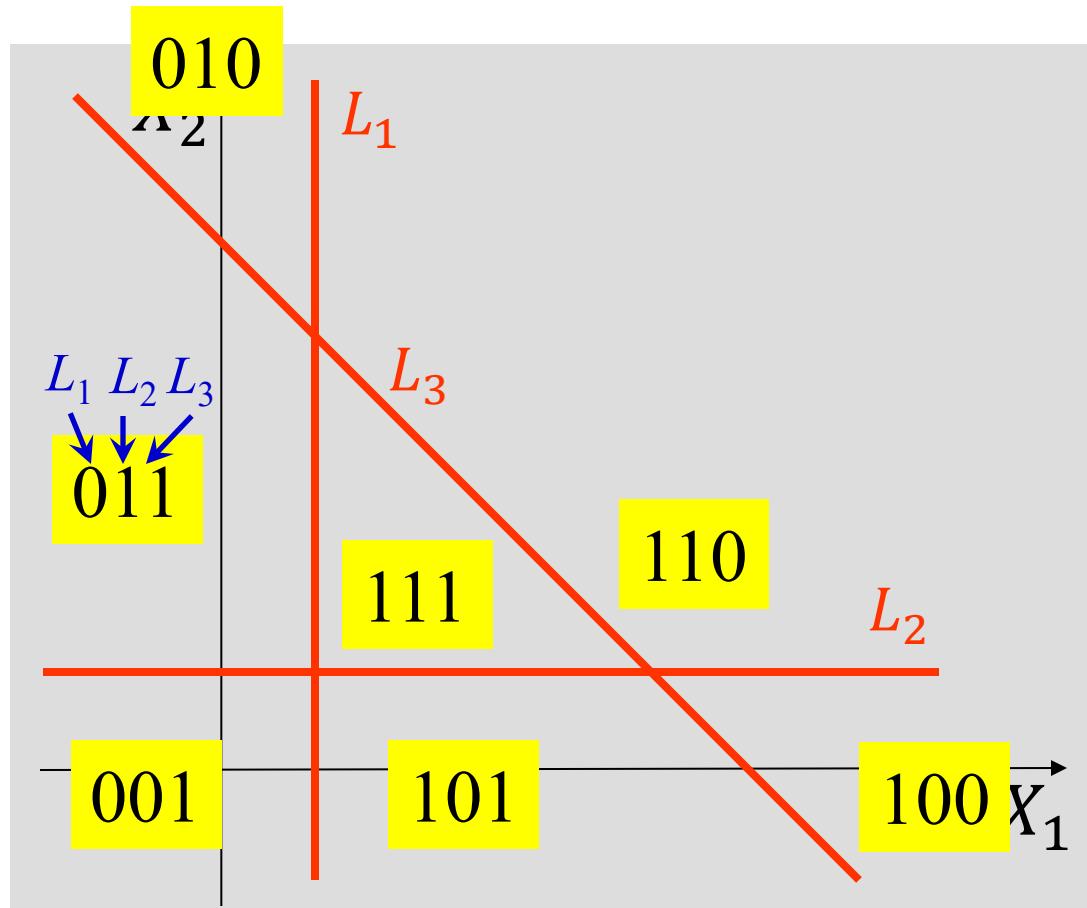
# Example: Not Linearly Separable



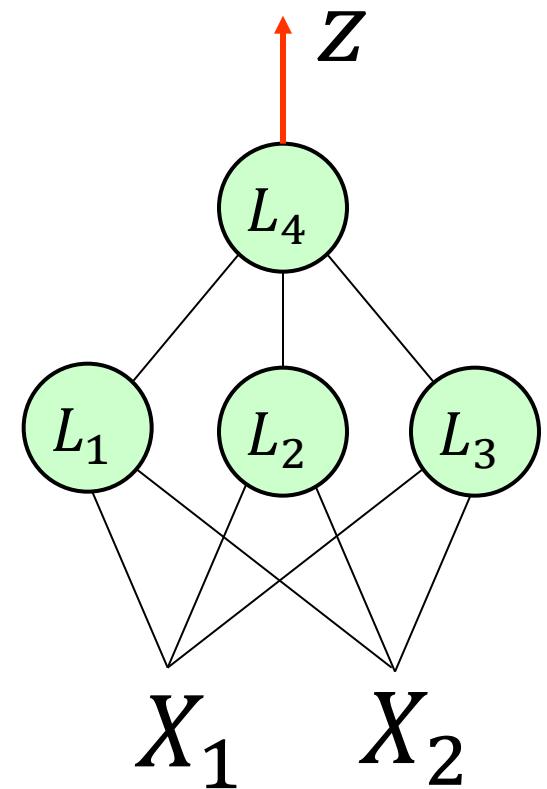
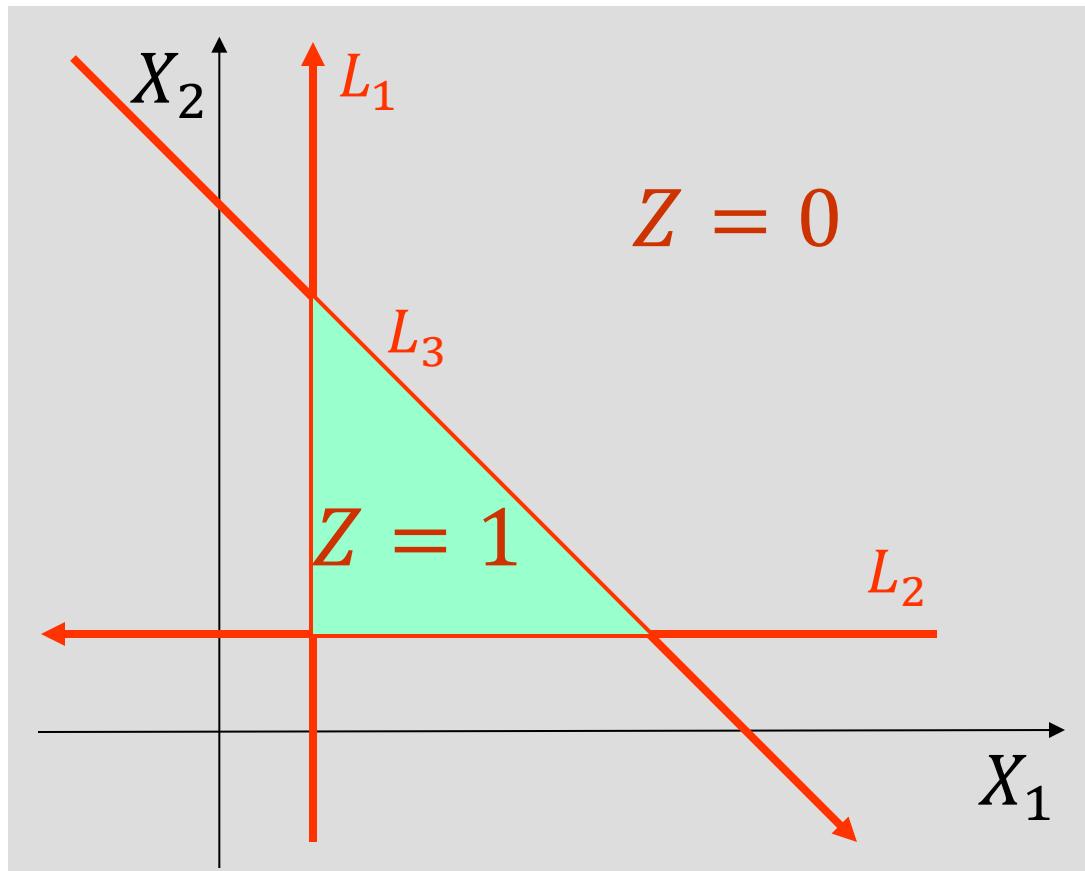
# Example: Not Linearly Separable



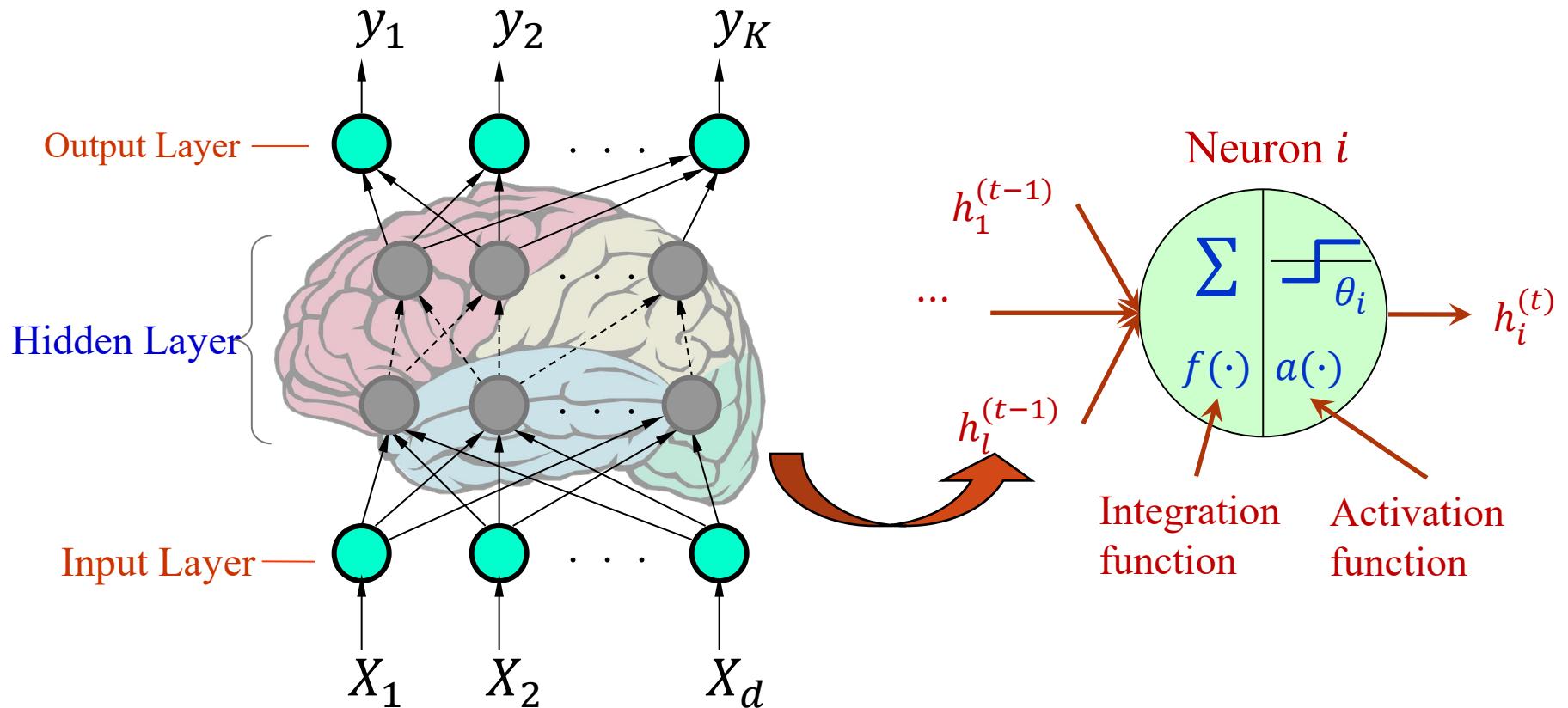
# Example: Not Linearly Separable



# Example: Not Linearly Separable ...



# General Structure: Multilayer ANN



# Integration Functions

- Weighted sum:

$$\sum_{i=1}^d w_i X_i - \theta$$


- Quadratic function

$$\sum_{i=1}^d w_i X_i^2 - \theta$$

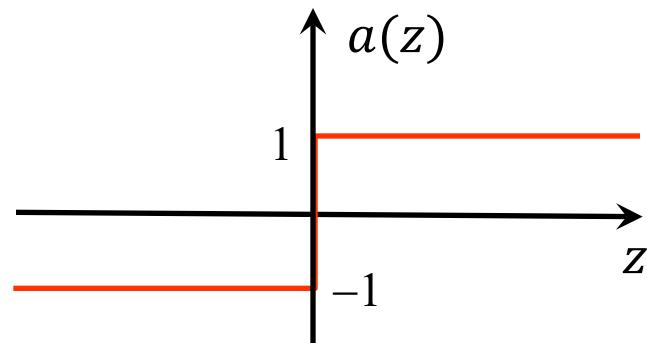
- Spherical function

$$\sum_{i=1}^d (X_i - w_i)^2 - \theta$$

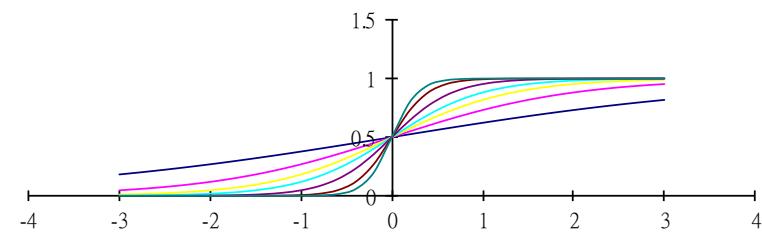
# Activation Functions

- Sign function (Threshold function)
- Unipolar sigmoid function:

$$a(z) = \text{sign}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$



$$a(z) = \frac{1}{1 + e^{-\lambda z}}$$



When  $\lambda = 1$ , it is called  
sigmoid function

# Update Weights for Multi-layer NNs

- Initialize the weights in each layer ( $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(k)}, \dots, \mathbf{w}^{(m)}$ )
- Adjust the weights such that the output of ANN is consistent with class labels of training examples
  - Loss function for each training instance:

$$\mathcal{L} = \frac{1}{2} (y_i - \hat{y}_i)^2$$

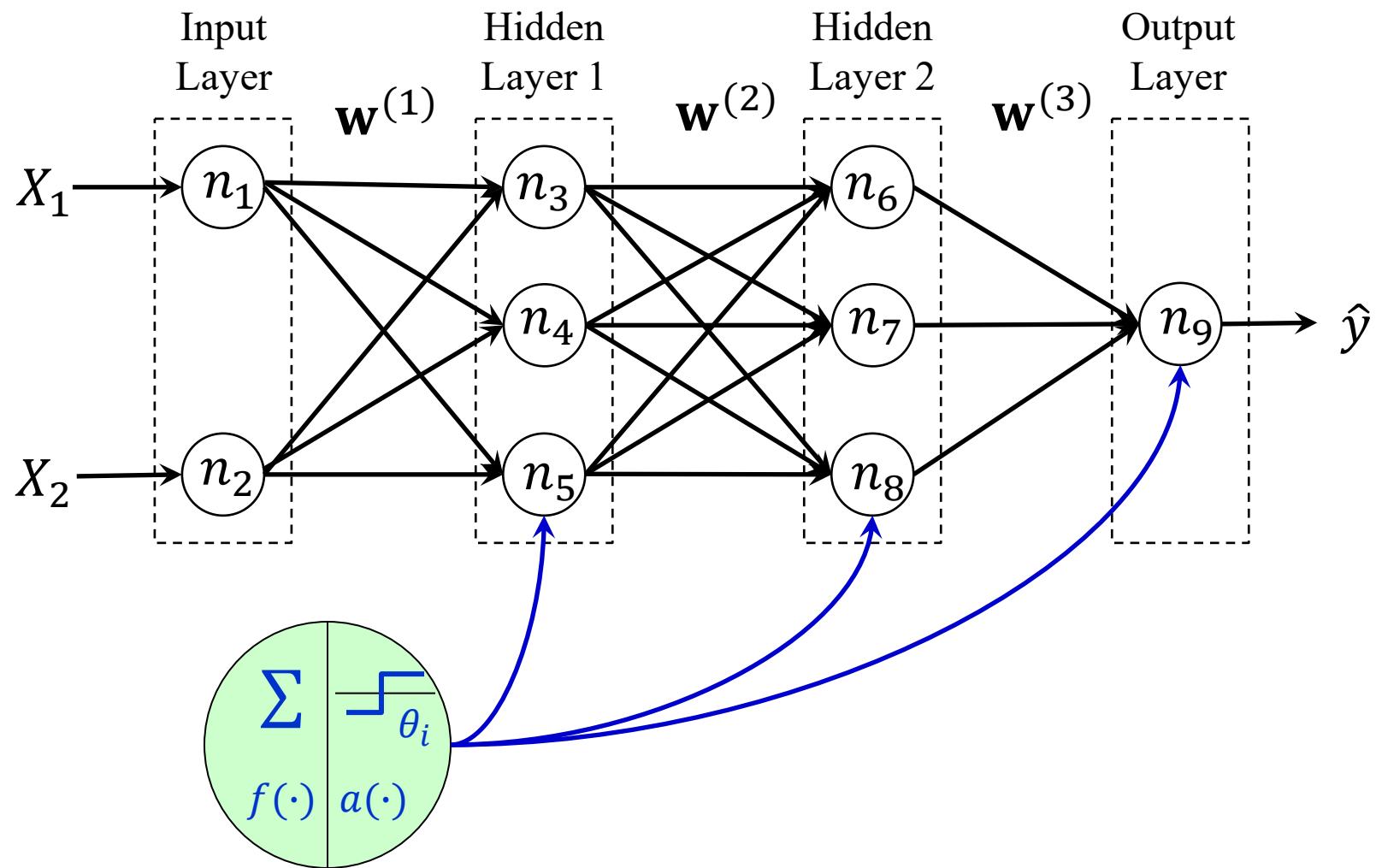
- For each layer  $k$ , update the weights,  $\mathbf{w}^{(k)}$ , by gradient descent at each iteration  $t$ :

$$\mathbf{w}_{t+1}^{(k)} = \mathbf{w}_t^{(k)} - \lambda \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(k)}}$$

- Computing the gradient w.r.t. weights in each layer is computationally expensive!

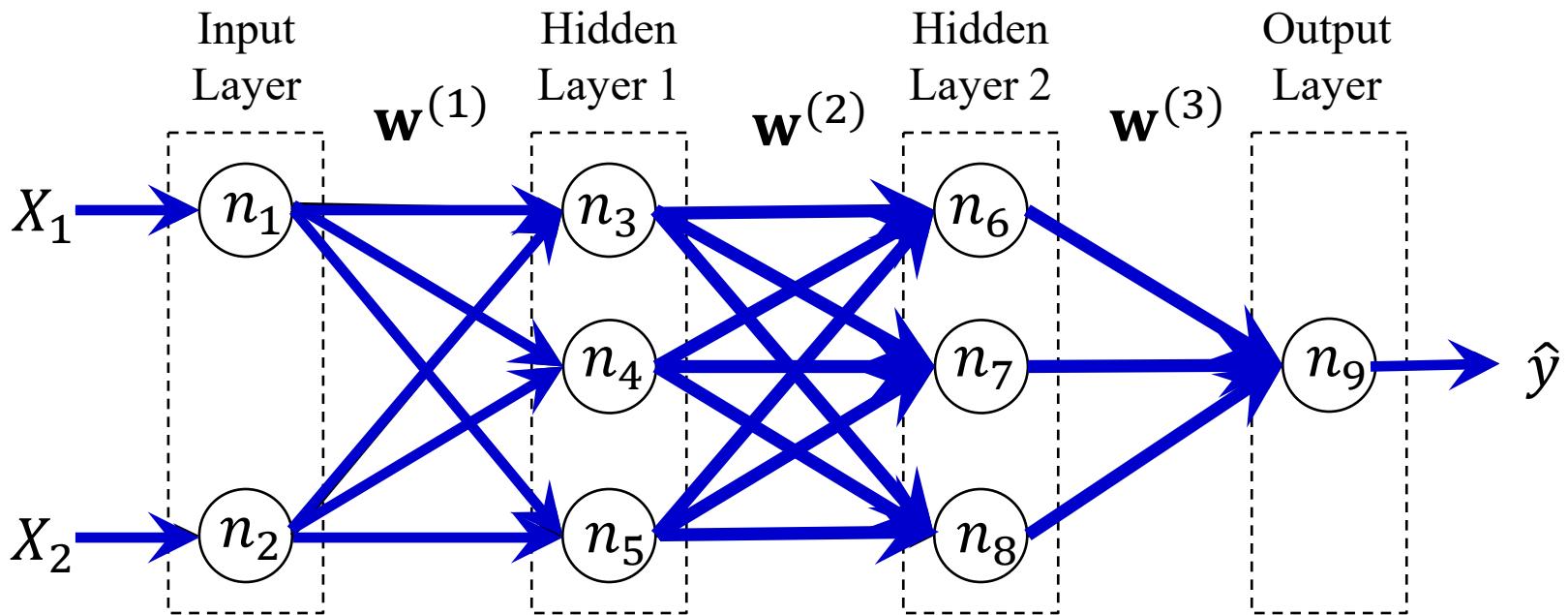
**The Backpropagation Algorithm**

# A Multi-layer Feed-forward NN



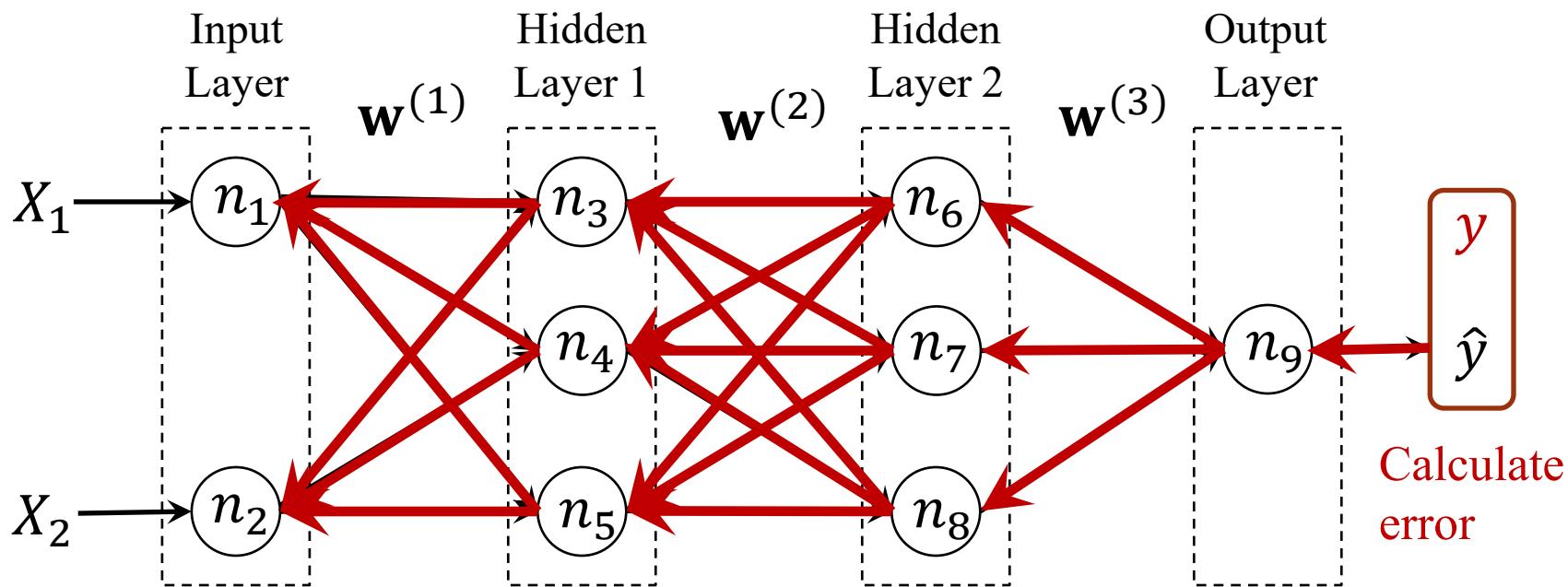
# Backpropagation: Basic Idea

- Initialize the weights ( $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(3)}$ )
- **Forward pass:** each training examples  $(x_i, y_i)$  is used to compute outputs of each hidden layer and generate the final output  $\hat{y}_i$  based on the ANN

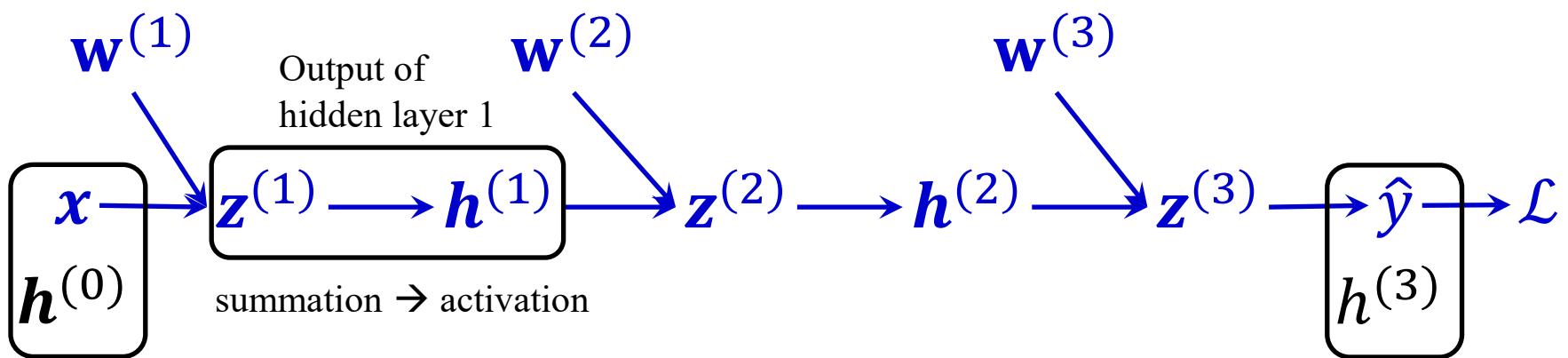
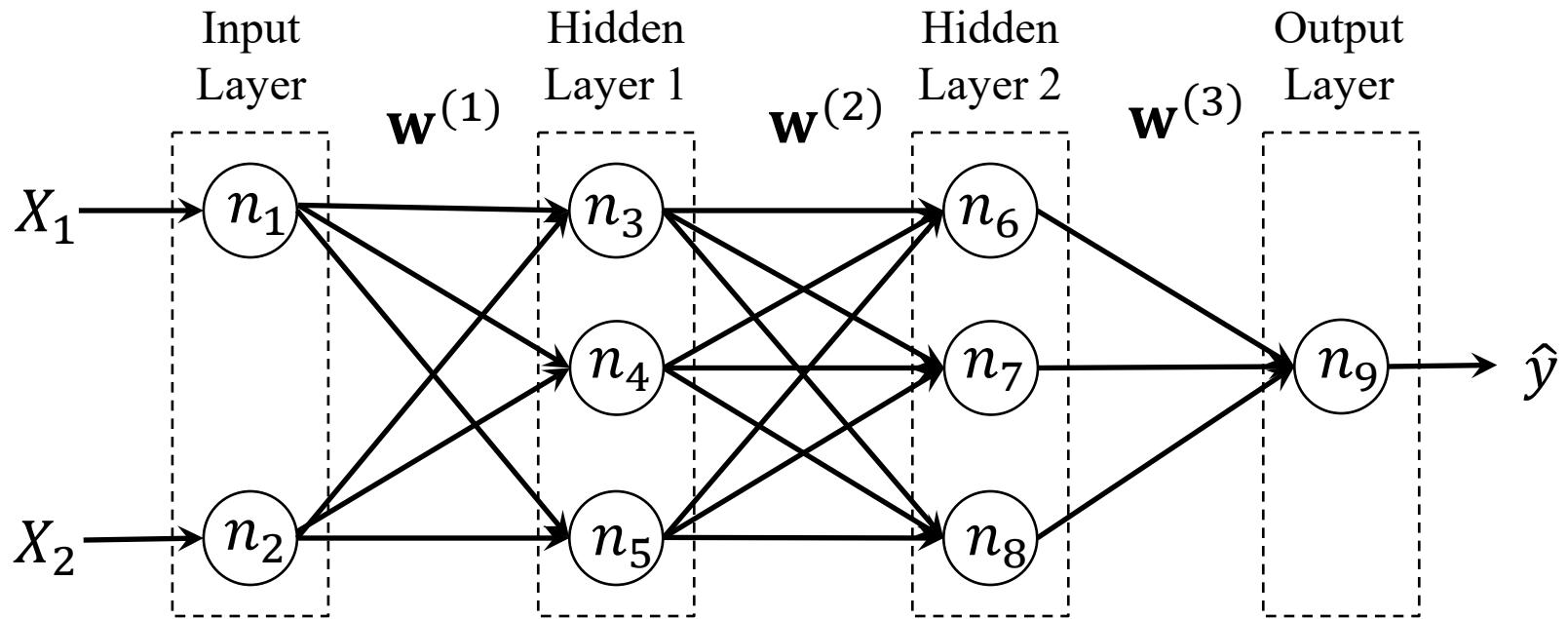


# Backpropagation: Basic Idea (cont.)

- **Backpropagation:** Starting with the output layer, to propagate error back to the previous layer in order to update the weights between the two layers, until the earliest hidden layer is reached



# The Computational Graph



# Backpropagation (BP)

- Gradient of  $\mathcal{L}$  w.r.t.  $w^{(3)}$ : 
$$\frac{\partial \mathcal{L}}{\partial w^{(3)}} = \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}}} \frac{\partial z^{(3)}}{\partial w^{(3)}}$$

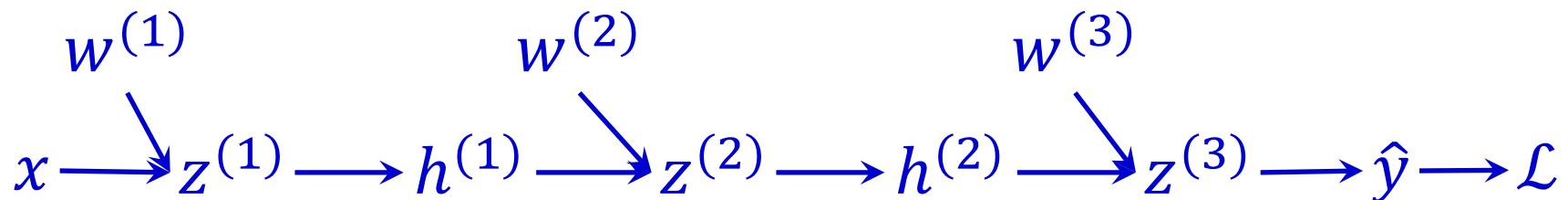
- Gradient of  $\mathcal{L}$  w.r.t.  $w^{(2)}$ :

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

- Gradient of  $\mathcal{L}$  w.r.t.  $w^{(1)}$ :

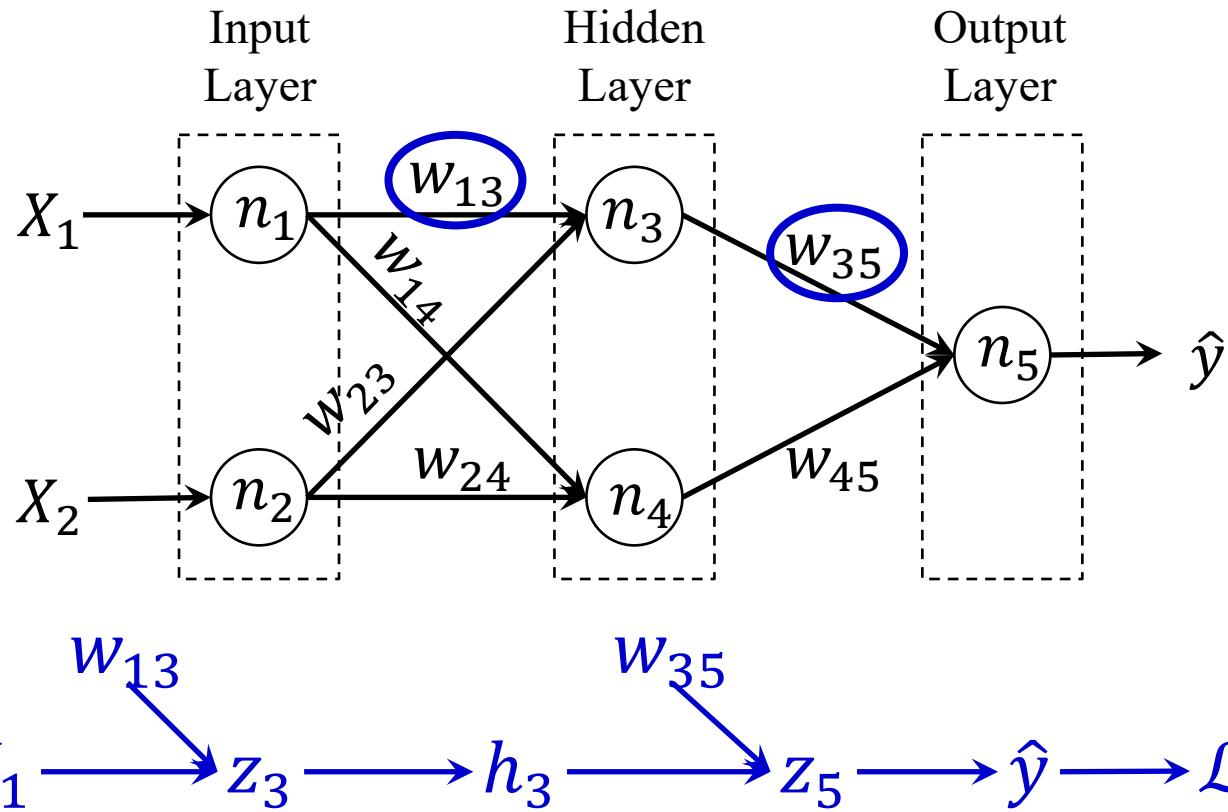
$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}}$$

Consider each layer contains a single unit



# An Example

- Consider an ANN of 1 hidden layer as follows. Suppose the sign function and the weighted sum function are used for both hidden and output nodes



$$w'_{35} = w_{35} + \lambda e_i h_3$$

$$w'_{35} = w_{35} - \lambda \frac{\partial \mathcal{L}}{\partial w_{35}} = w_{35} - \lambda \frac{\frac{\partial \mathcal{L}}{\partial \hat{y}}}{\frac{\partial \hat{y}}{\partial z_5}} \frac{\partial z_5}{\partial w_{35}}$$

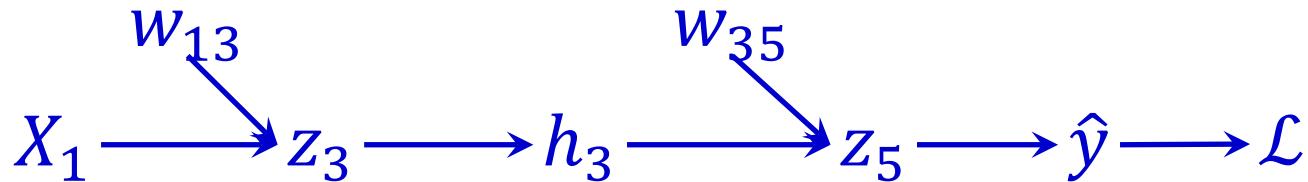
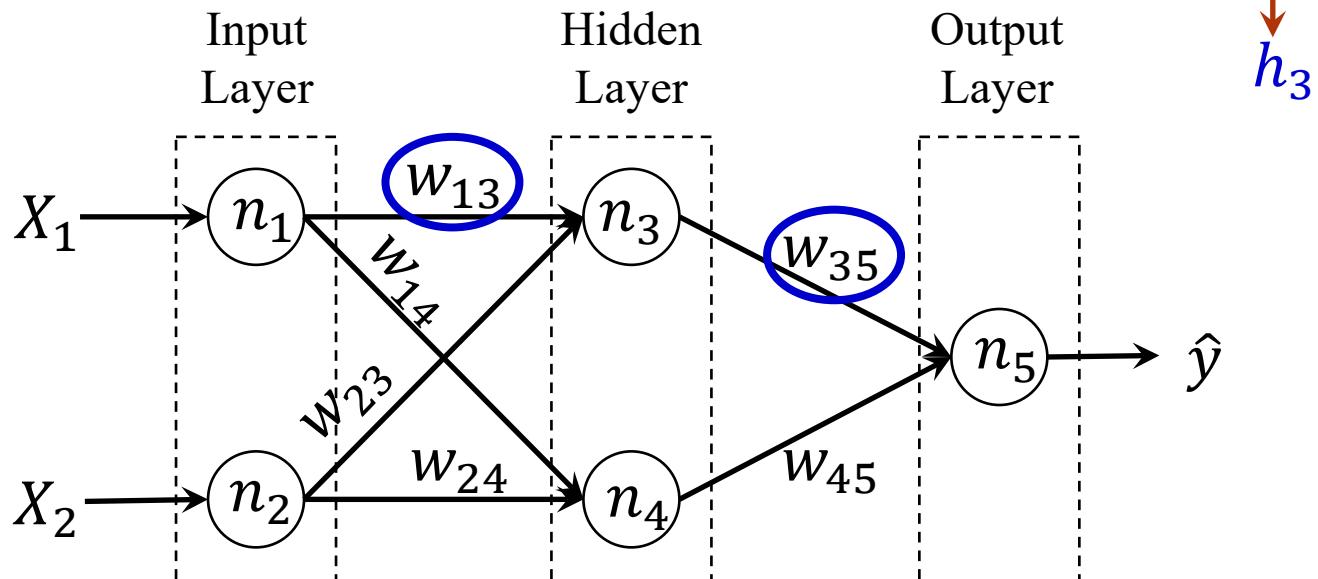
$$\mathcal{L} = \frac{1}{2} e_i^2 = \frac{1}{2} (y_i - \hat{y}_i)^2$$

$$-1 \times (y_i - \hat{y}_i) = -e_i$$

$$z_5 = w_{35} h_3 + w_{45} h_4$$

~~$\hat{y} = \text{sgn}(z_5)$~~        $\frac{\partial \hat{y}}{\partial z_5} = 1$

$\hat{y} = z_5$



$$w'_{13} = w_{13} + \lambda e_i w_{35} X_1$$

Obtained when  
updating  $w_{35}$

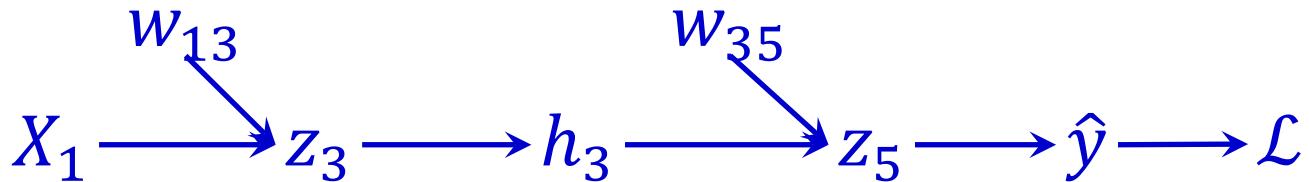
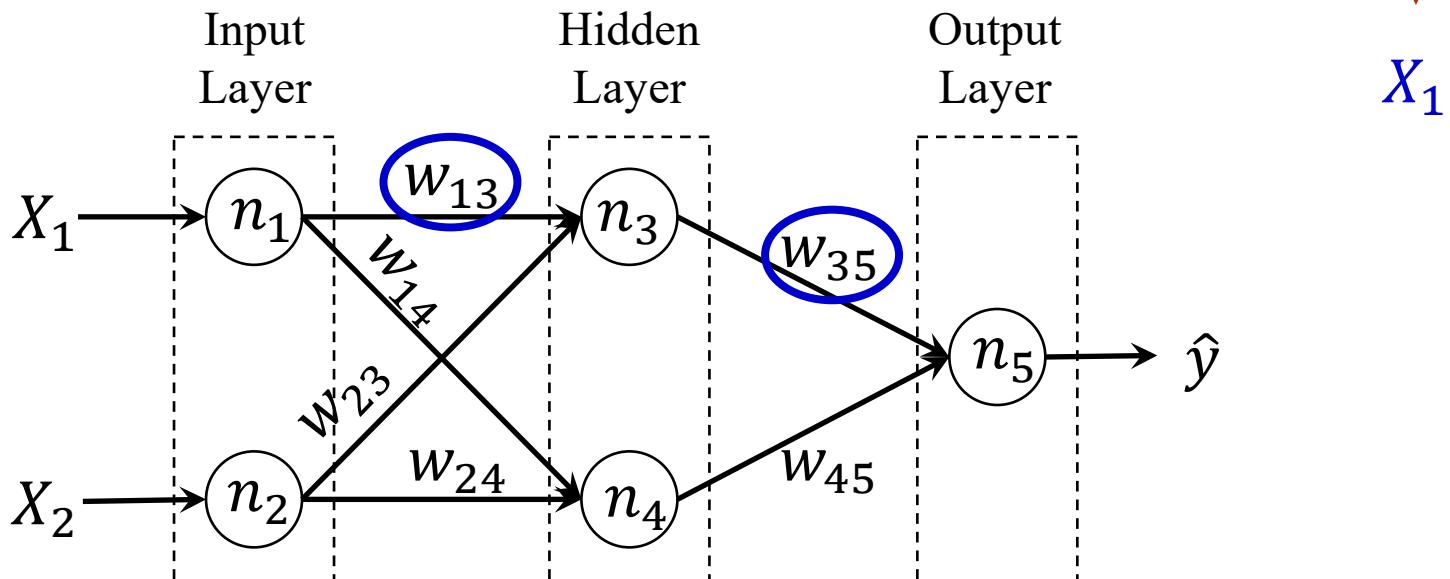
$$w'_{13} = w_{13}$$

$$\lambda \frac{\partial \mathcal{L}}{\partial w_{13}} = w_{13} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial h_3} \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial w_{13}}$$

$$z_5 = w_{35} h_3 + w_{45} h_4$$

$$\begin{aligned} h_3 &= \text{sgn}(z_3) \\ h_3 &= z_3 \end{aligned} \quad \frac{\partial h_3}{\partial z_3} = 1$$

$$\begin{aligned} w_{35} \\ z_3 = w_{13} X_1 + w_{23} X_2 \\ X_1 \end{aligned}$$



$$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$$

Obtained when  
updating  $w_{35}$

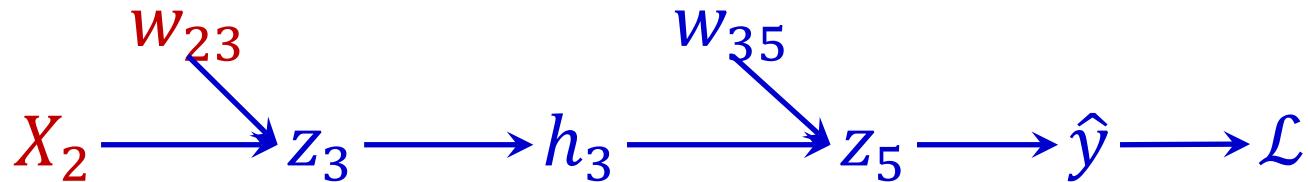
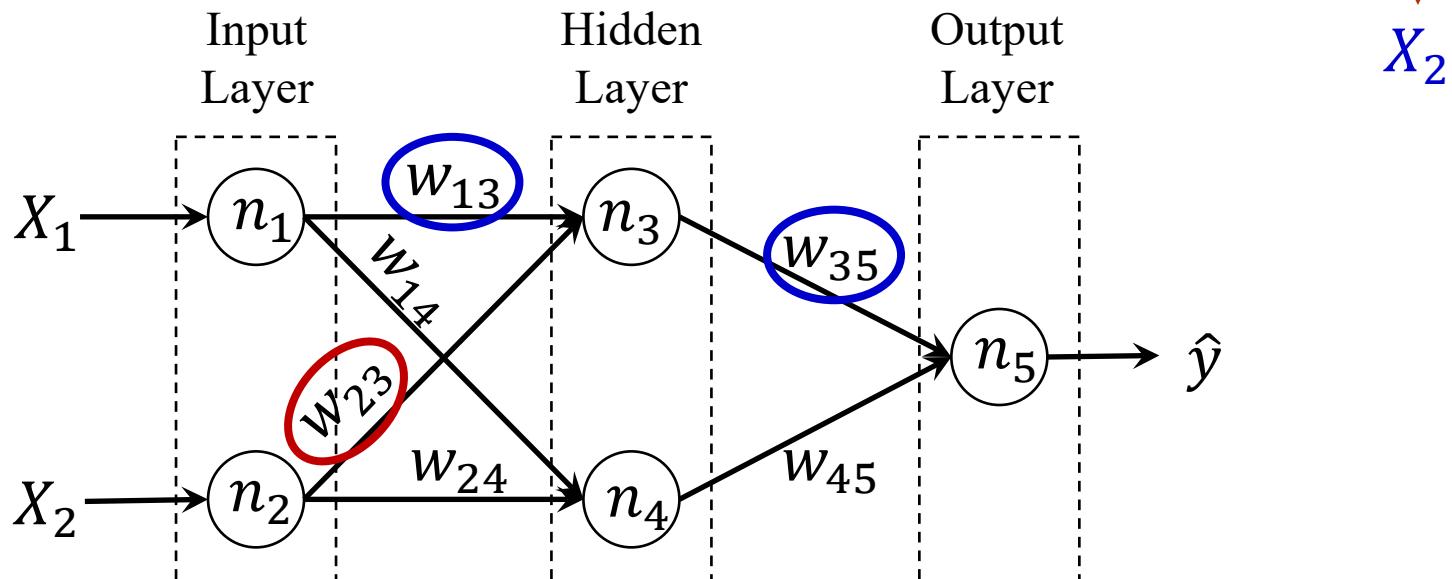
$$w'_{23} = w_{23}$$

$$\lambda \frac{\partial \mathcal{L}}{\partial w_{23}} = w_{23} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial h_3} \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial w_{23}}$$

$$z_5 = w_{35} h_3 + w_{45} h_4$$

$$\begin{aligned} h_3 &= \text{sgn}(z_3) \\ h_3 &= z_3 \end{aligned} \quad \frac{\partial h_3}{\partial z_3} = 1$$

$$\begin{aligned} w_{35} \\ z_3 = w_{13} X_1 + w_{23} X_2 \end{aligned}$$



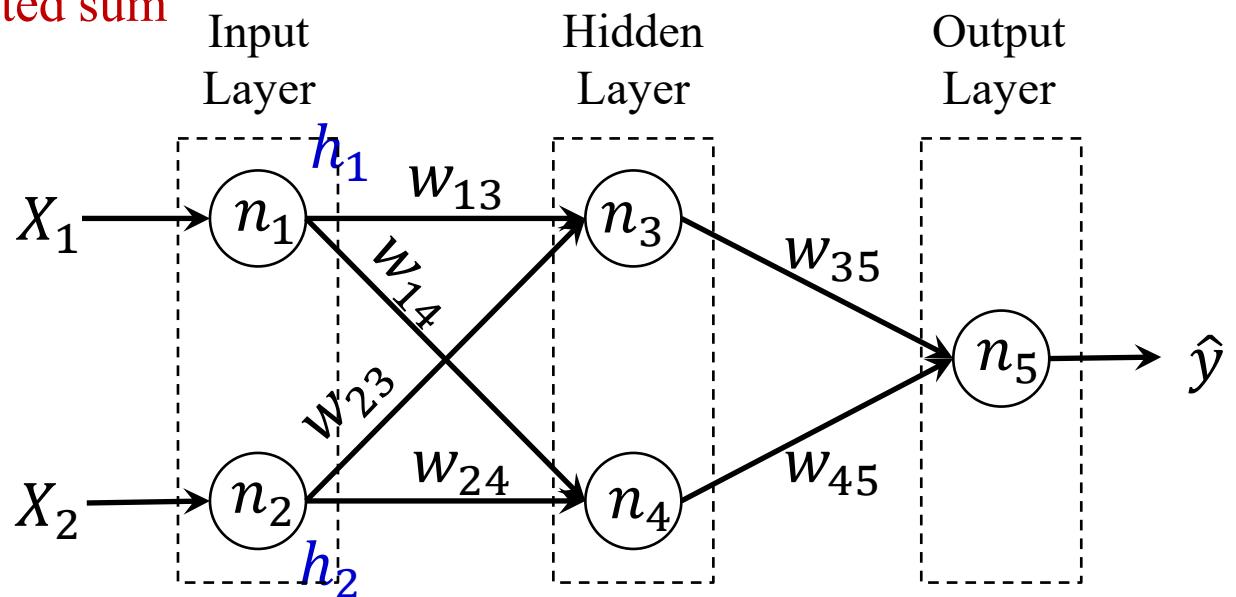
# BP Algorithm: Example

Activation function: sign()

Integration function: weighted sum

$$\lambda = 0.4, \theta = 0$$

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



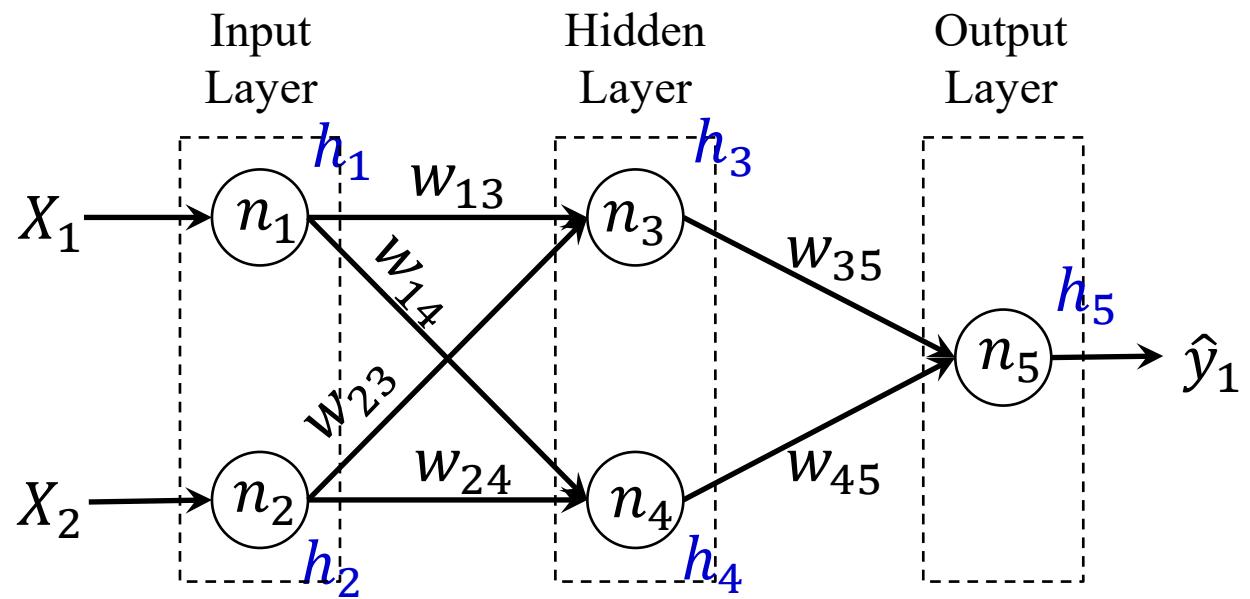
- Initialization:

$$(w_{13} = 1, w_{14} = 1, w_{23} = 1, w_{24} = 1, w_{35} = 1, w_{45} = 1)$$

For the 1<sup>st</sup> example:  $h_1 = 0$  and  $h_2 = 0$

# BP Algorithm: Example (cont.)

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



Forward pass:

$$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1 \text{ and } h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$$

$$\text{Then } \hat{y}_1 = h_5 = \text{sign}(1 \times 1 + 1 \times 1) = 1$$

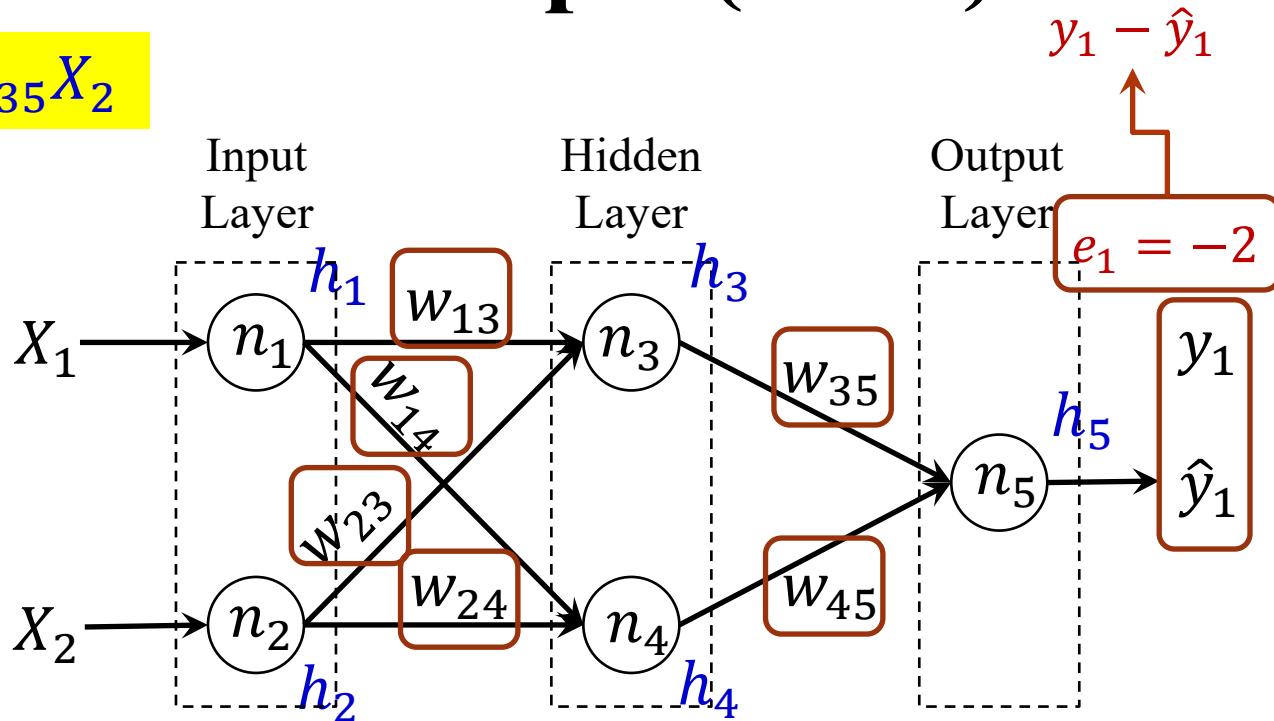
$$w'_{13} = w_{13} + \lambda e_1 w_{35} X_1$$

$$w'_{35} = w_{35} + \lambda e_1 h_3$$

# BP Algorithm: Example (cont.)

$$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$$

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



Backpropagation:

$$w_{35} = 1 + 0.4 \times (-2) \times 1 = 0.2$$

$$w_{13} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$$

$$w_{23} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$$

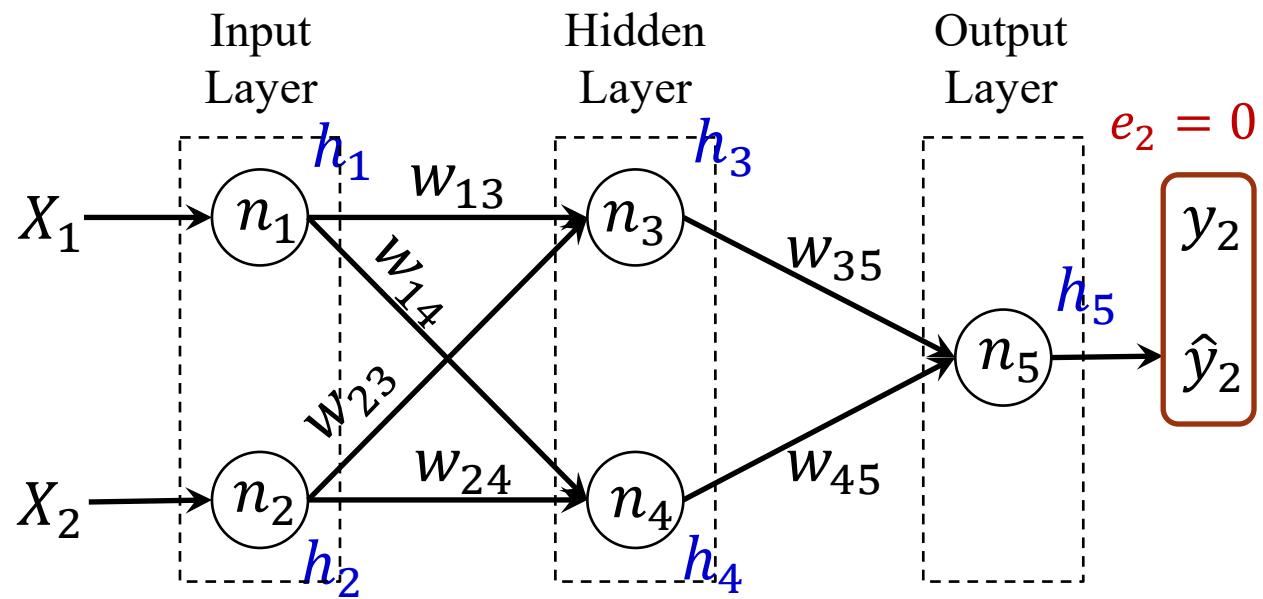
$$w_{45} = 1 + 0.4 \times (-2) \times 1 = 0.2$$

$$w_{14} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$$

$$w_{24} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$$

# BP Algorithm: Example (cont.)

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



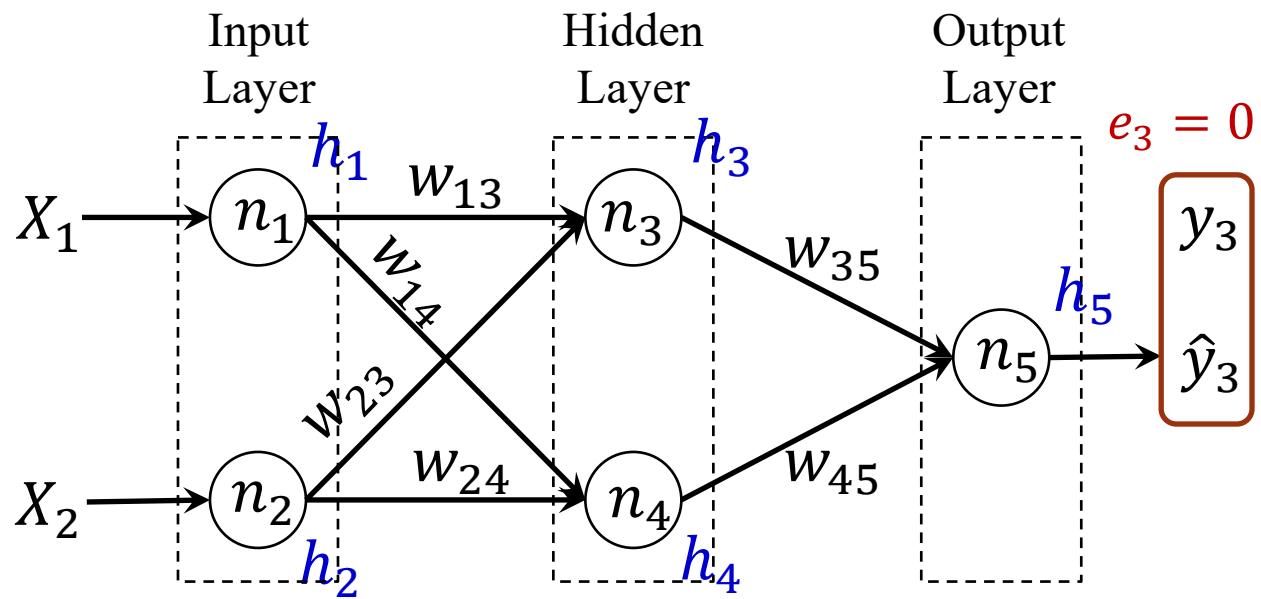
For the 2<sup>nd</sup> example:  $h_1 = 1$  and  $h_2 = 0$

$$h_3 = \text{sign}(1 \times 1 + 0 \times 1) = 1 \text{ and } h_4 = \text{sign}(1 \times 1 + 0 \times 1) = 1$$

$$\text{Then } \hat{y}_2 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$$

# BP Algorithm: Example (cont.)

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



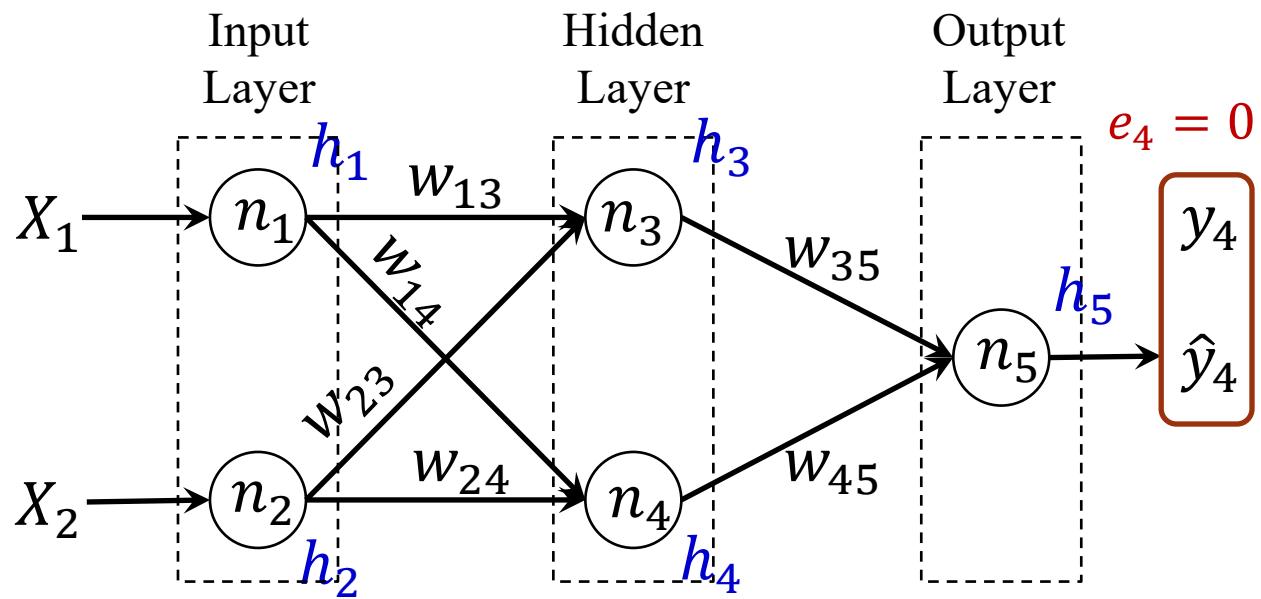
For the 3<sup>rd</sup> example:  $h_1 = 0$  and  $h_2 = 1$

$h_3 = \text{sign}(0 \times 1 + 1 \times 1) = 1$  and  $h_4 = \text{sign}(0 \times 1 + 1 \times 1) = 1$

Then  $\hat{y}_3 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

# BP Algorithm: Example (cont.)

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



For the 4<sup>th</sup> example:  $h_1 = 1$  and  $h_2 = 1$

$h_3 = \text{sign}(1 \times 1 + 1 \times 1) = 1$  and  $h_4 = \text{sign}(1 \times 1 + 1 \times 1) = 1$

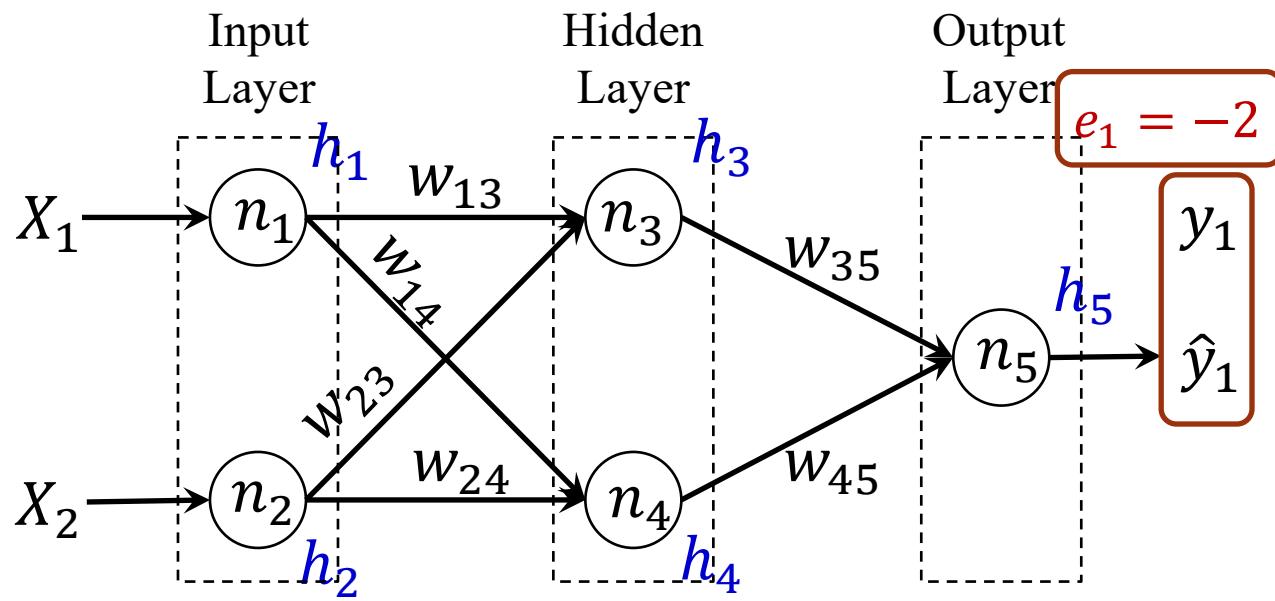
Then  $\hat{y}_4 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

End of the 1<sup>st</sup> Epoch

The 2<sup>nd</sup> Epoch starts

# BP Algorithm: Example (cont.)

$X_1$	$X_2$	y
0	0	-1
1	0	1
0	1	1
1	1	1



For the 1<sup>st</sup> example again:  $h_1 = 0$  and  $h_2 = 0$

$$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1 \text{ and } h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$$

$$\text{Then } \hat{y}_1 = h_5 = \text{sign}(0.2 \times 1 + 0.2 \times 1) = 1$$

Weights need to be further updated via backpropagation

# Design Issues for ANN

- The number of nodes in the input layer
  - Assign an input node to each numerical or binary input variable
- The number of nodes in the output layer
  - Binary class problem → single node
  - $C$ -class problem →  $C$  output nodes
- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data

# Design Issues for ANN

- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data
- If the network underfits
  - Try to increase the number of hidden units
- If the network overfits
  - Try to decrease the number of hidden units

# CZ4041/SC4000: Machine Learning

## Lesson 8a: Support Vector Machines

LI Boyang, Albert

School of Computer Science and Engineering,  
NTU, Singapore

# Support Vector Machines (SVMs)

- SVMs have shown promising empirical results in many practical applications, such as computer vision, sensor networks and text mining
- The motivation behind SVMs is from the geometry perspective of linear algebra
- The objective of SVMs is to learn a maximum margin hyperplane
  - Based on statistical learning theory

# If You Have Questions



Go to [wooclap.com](https://wooclap.com) and enter the event code in the top banner:

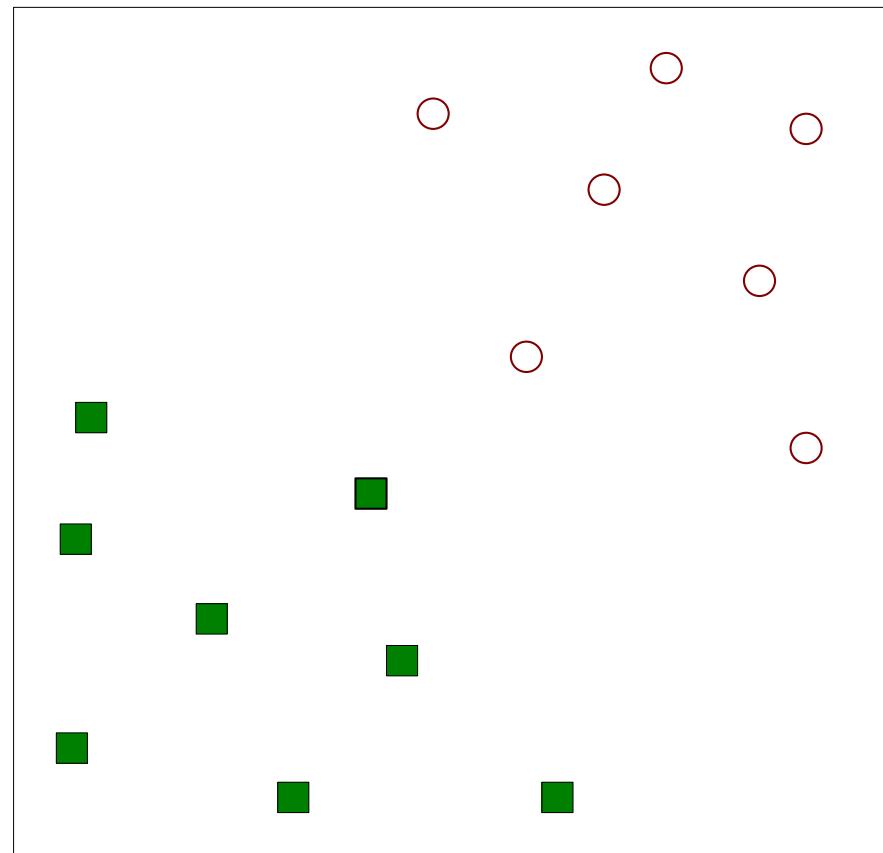
Event code  
**NESKSR**

# Separating Hyperplane

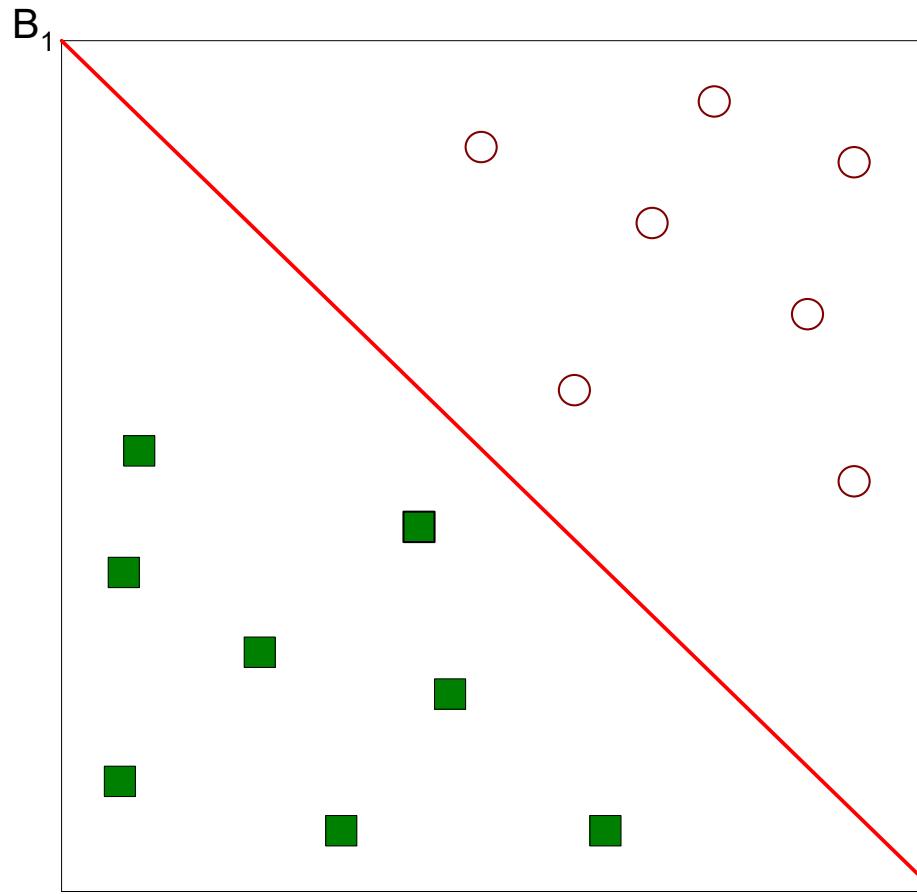
- To learn a binary classifier



- To find a hyperplane (linear decision boundary) so that all the squares reside on one side of the hyperplane and all the circles reside on the other

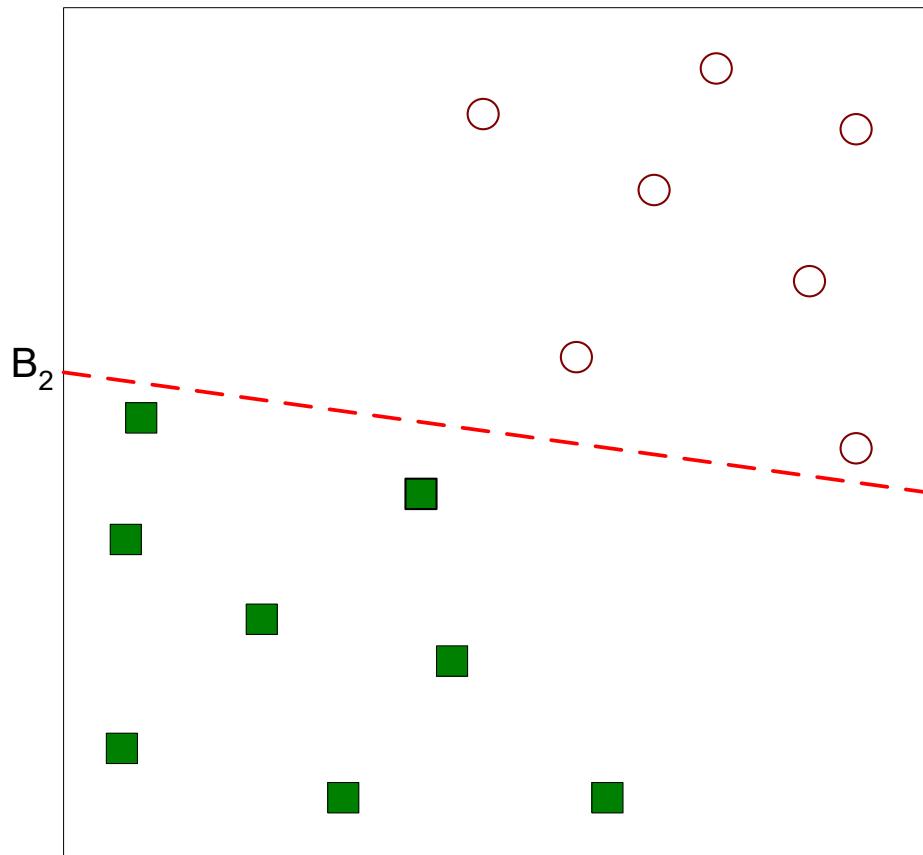


# Separating Hyperplane (cont.)



One Possible Solution

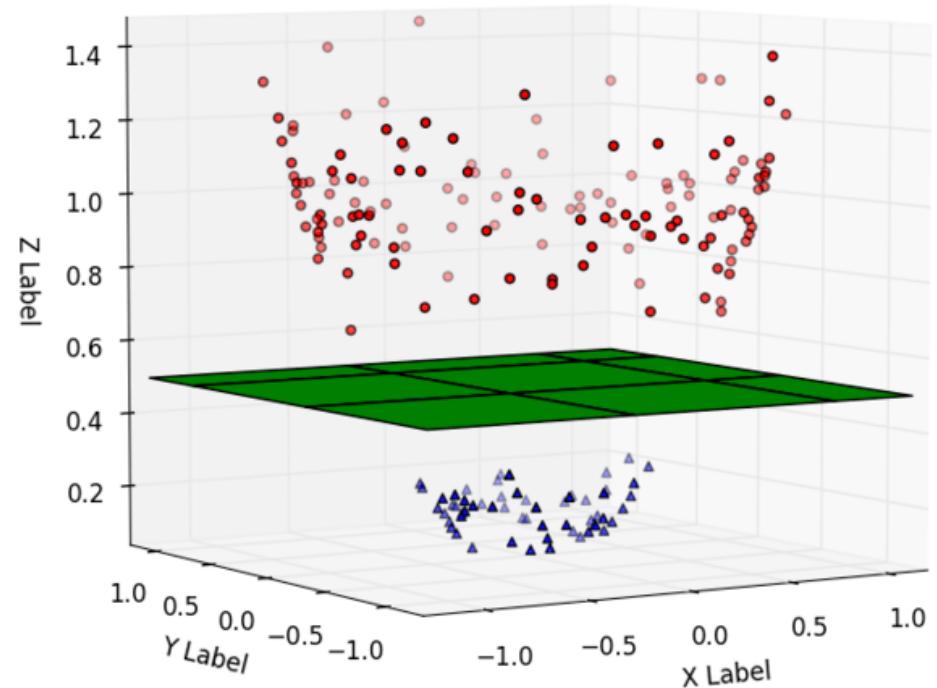
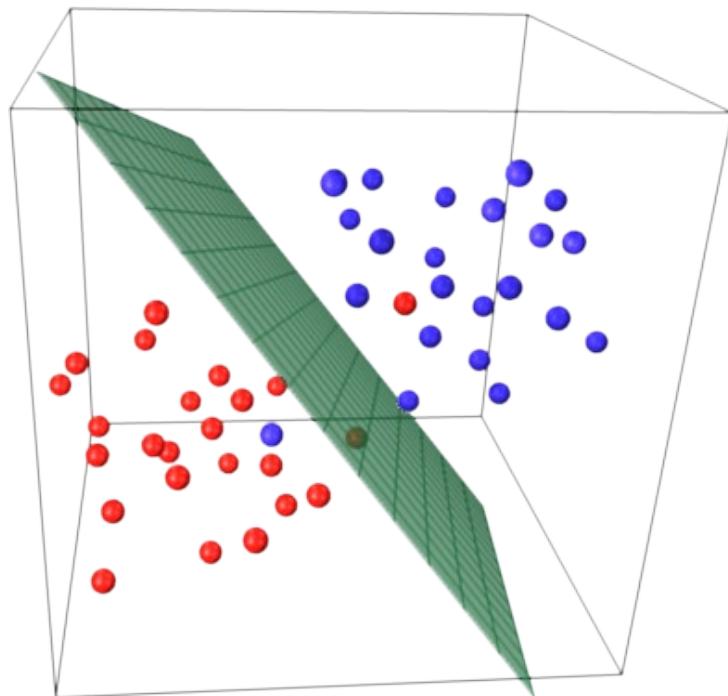
# Separating Hyperplane (cont.)



Another possible solution

# Separating Hyperplane in 3D

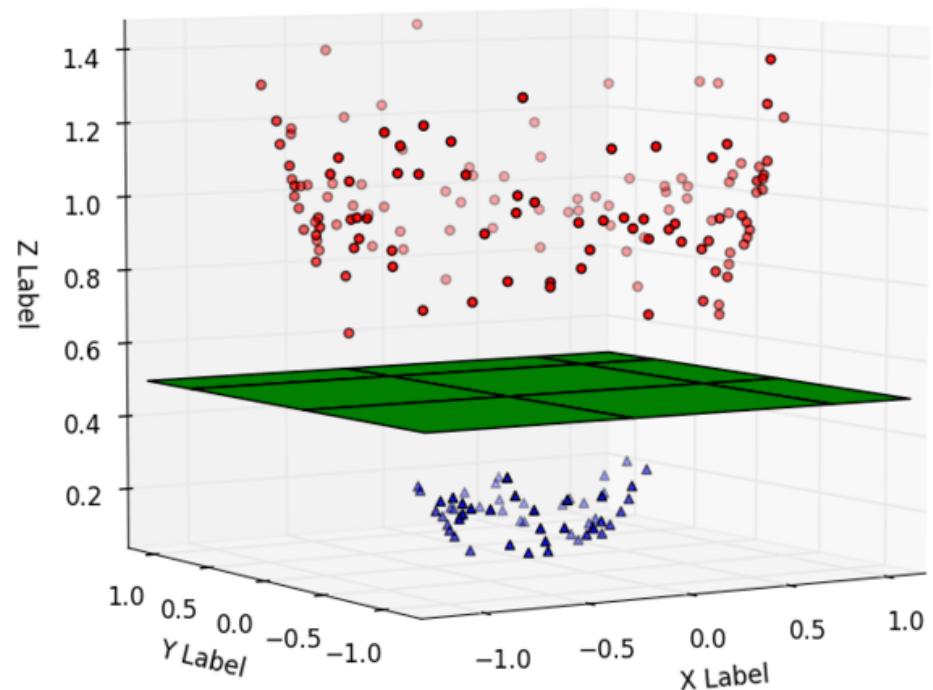
Data in  $\mathbb{R}^3$  (separable w/ hyperplane)



# Separating Hyperplane in 3D

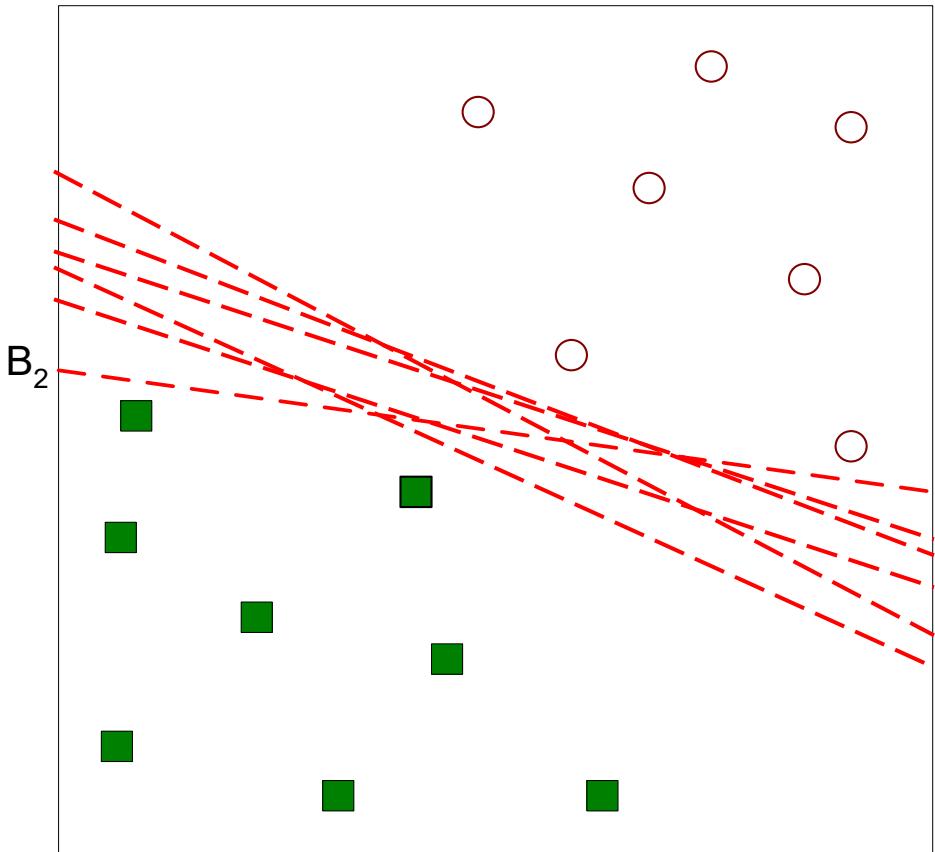
Data not separable in  
2D may be separable in  
3D

Data in  $\mathbb{R}^3$  (separable w/ hyperplane)

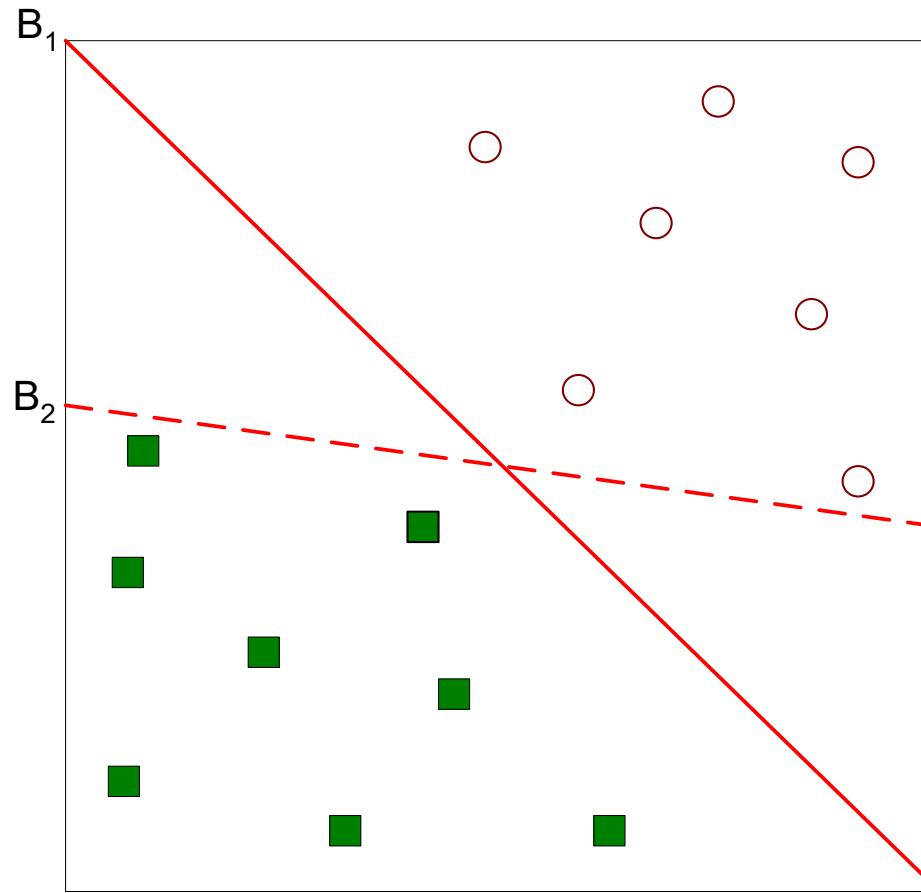


# Maximum Margin

- Although all the hyperplanes in the figure can separate training examples perfectly, their generalization errors may differ.
- How to choose one of these hyperplanes to construct a classifier's decision boundary with small generalization errors?

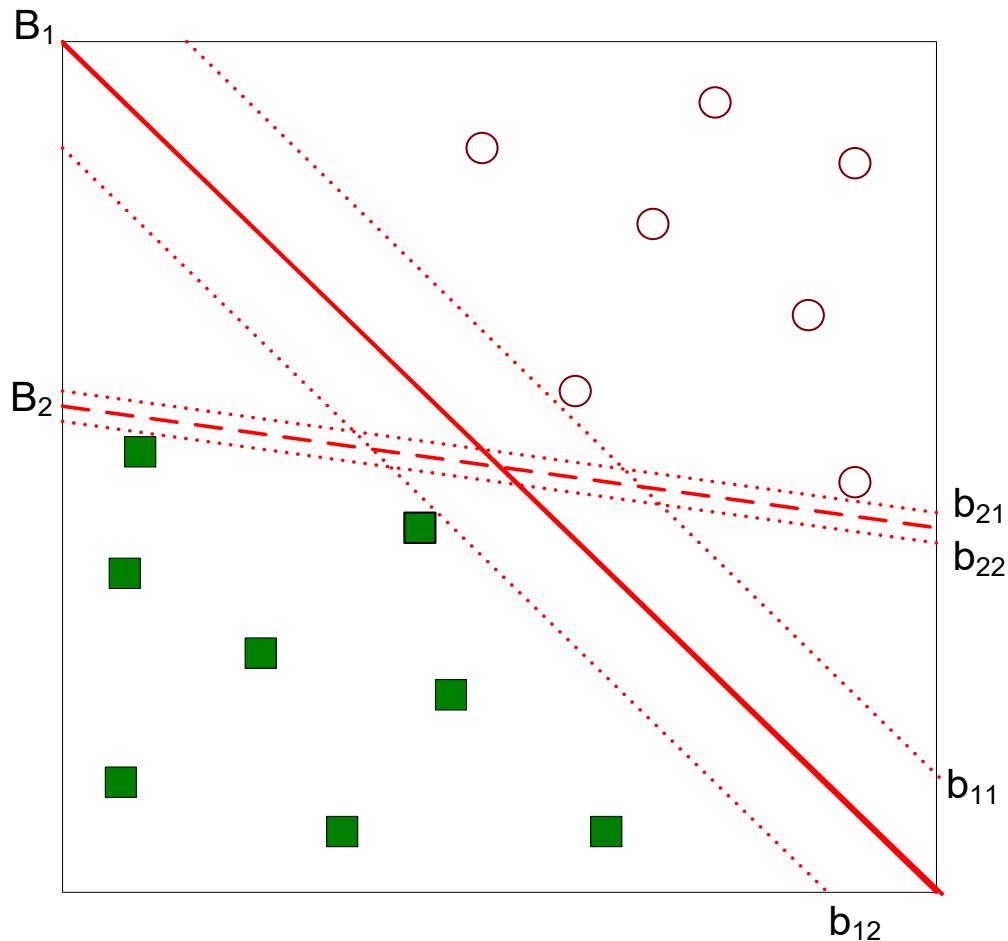


# Maximum Margin (cont.)



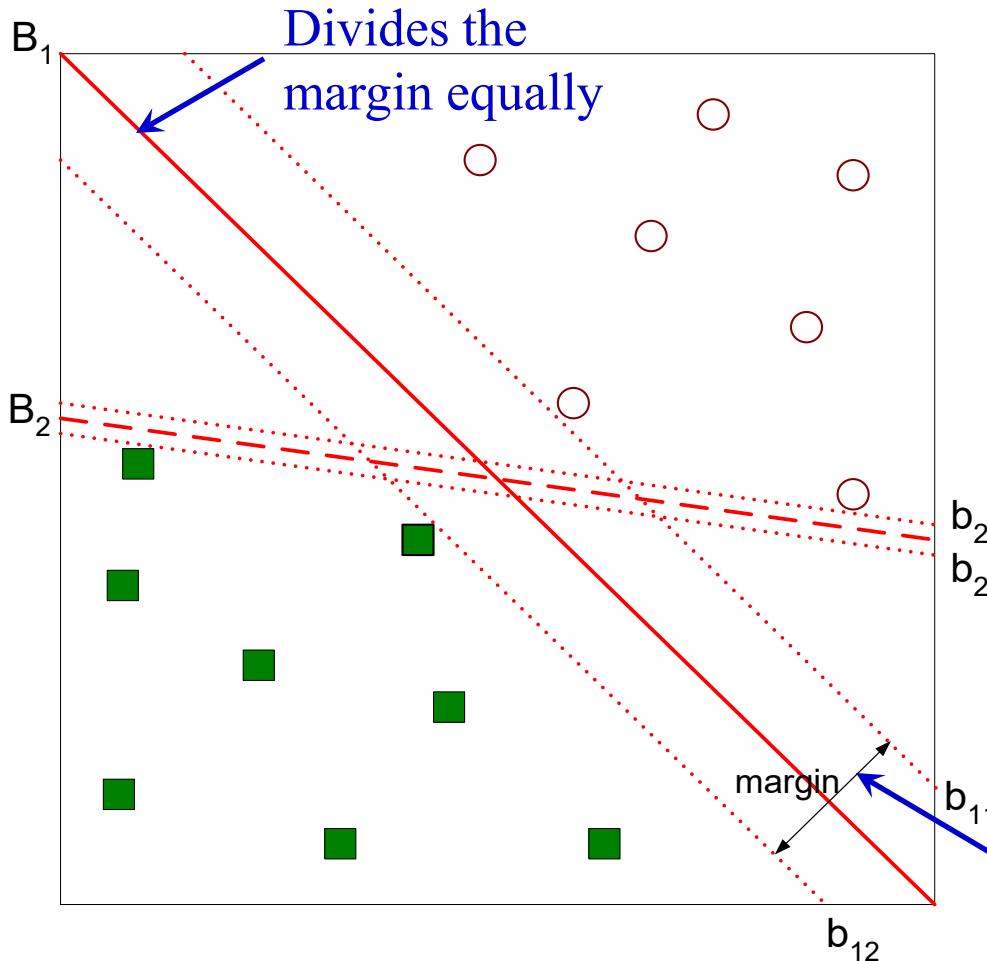
Which one is better?  $B_1$  or  $B_2$ ?

# Maximum Margin (cont.)



- Each decision boundary  $B_i$  is associated with a pair of parallel hyperplanes:  $b_{i1}$  and  $b_{i2}$
- $b_{i1}$  is obtained by moving the hyperplane away from the original boundary until it touches the closest circles
- $b_{i2}$  is obtained by moving a hyperplane until it touches the closest squares
- The distances from  $b_{i1}$  and  $b_{i2}$  to  $B_i$  are the same

# Maximum Margin (cont.)



- Assumption: larger margins imply better generalization errors
- The margin of  $B_1$  is much larger than that of  $B_2$ . Therefore,  $B_1$  is better than  $B_2$

The distance between these two hyperplanes is known as the margin of the classifier

# Decision Boundary

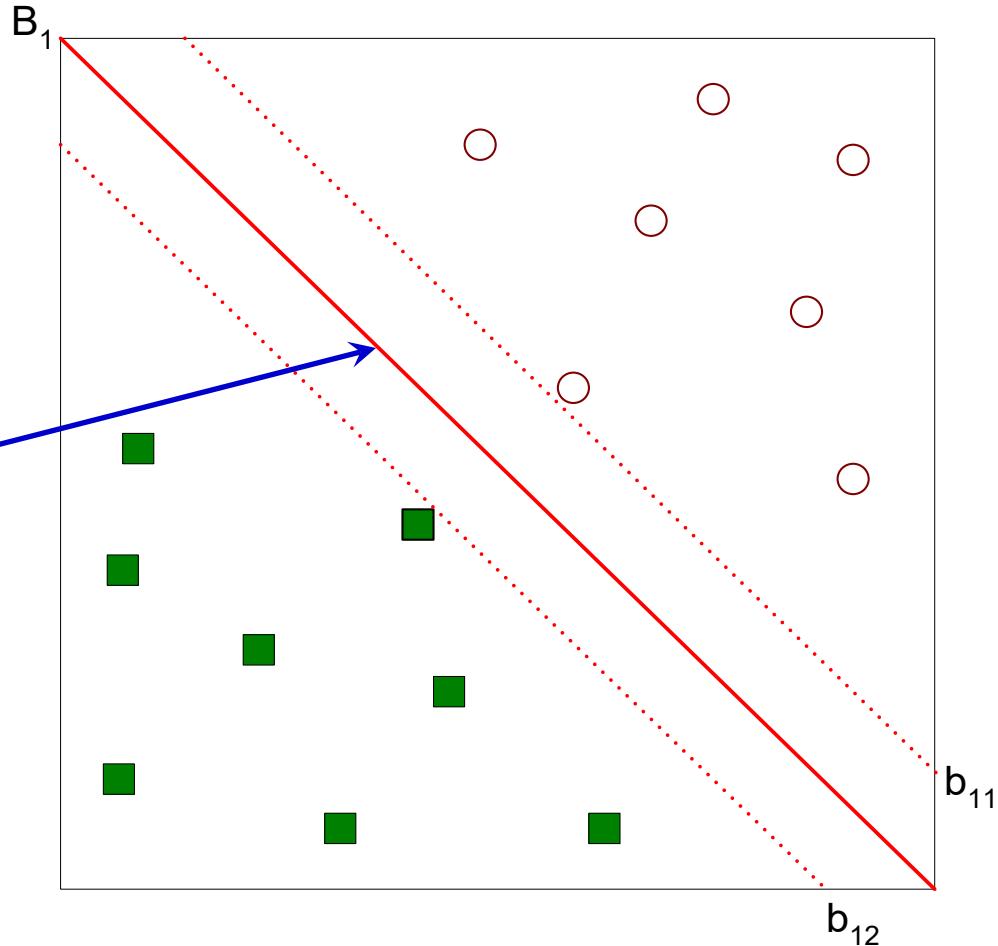
- Given a binary classification task, denote  $y_i = +1$  the circle class, and  $y_i = -1$  the square class

Decision boundary:

$$w_1x_1 + w_2x_2 + b = 0$$

General form:  $w \cdot x + b = 0$

Inner product:  $w \cdot x = \sum_{i=1}^D w_i x_i$



# Review: Inner Products of Vectors

- We use bold letters to denote vectors, such as  $\mathbf{a}$  and  $\mathbf{b}$ .
- A vector can have many dimensions:  $\mathbf{a} = (a_1, a_2, \dots, a_D)$
- The inner product of two  $D$ -dimensional vectors ( $D$ -vectors),  $\mathbf{a}$  and  $\mathbf{b}$  is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_D b_D = \sum_i a_i b_i$$

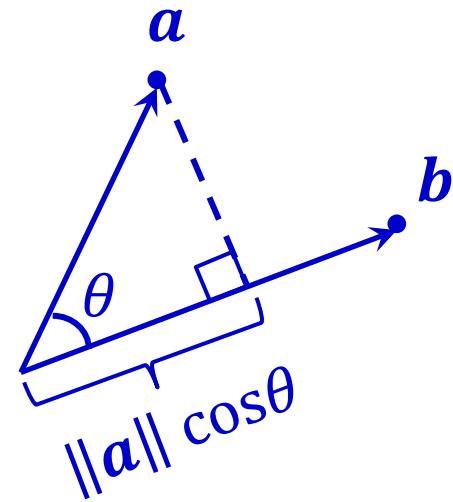
- Also,  $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos\theta$ , where  $\theta$  is the angle between  $\mathbf{a}$  and  $\mathbf{b}$
- $\|\mathbf{a}\|$  is the Euclidean norm of  $\mathbf{a}$

$$\|\mathbf{a}\| = \|\mathbf{a}\|_2 = \sqrt{a_1^2 + a_2^2 + \cdots + a_D^2} = \sqrt{\mathbf{a} \cdot \mathbf{a}}$$

# Review: Geometry of Inner Products

$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos\theta$ , where  $\theta$  is the angle between  $\mathbf{a}$  and  $\mathbf{b}$

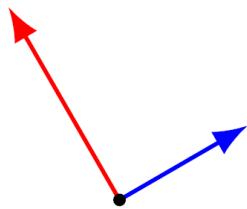
$\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|} = \|\mathbf{a}\| \cos\theta$  is the length of the projection of  $\mathbf{a}$  on (or onto)  $\mathbf{b}$



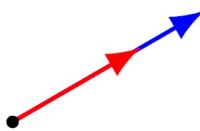
What if  $\mathbf{a}$  and  $\mathbf{b}$  form an obtuse angle?

If  $\mathbf{a}$  and  $\mathbf{b}$  are orthogonal,  
 $\theta = 90^\circ, \cos\theta = 0, \mathbf{a} \cdot \mathbf{b} = 0$

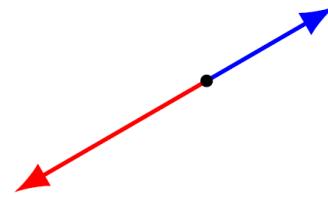
# Which dot products are negative?



(a)



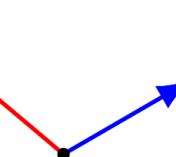
(b)



(c)



(d)



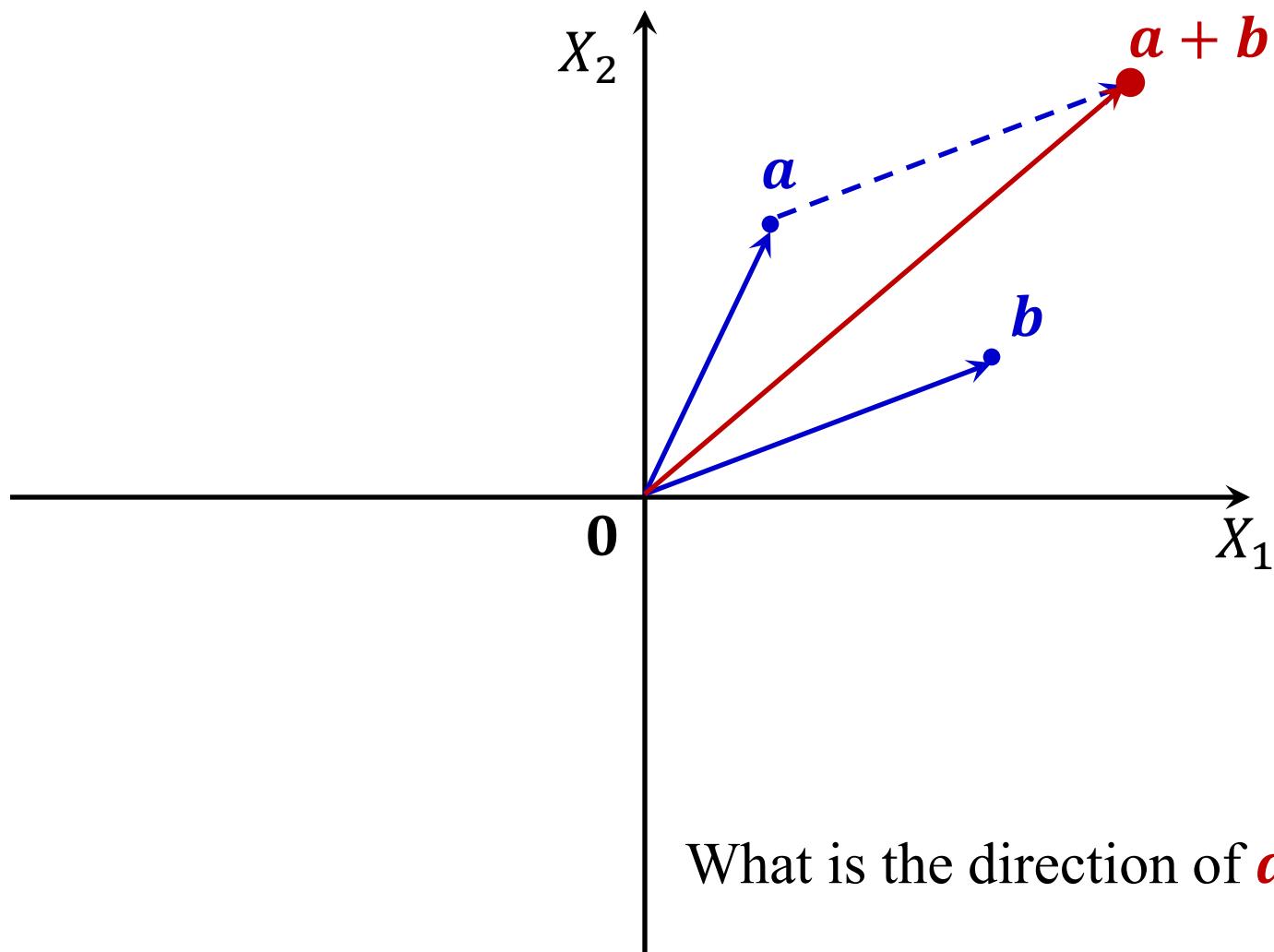
(e)



Go to [wooclap.com](https://wooclap.com) and enter the event code in the top banner:

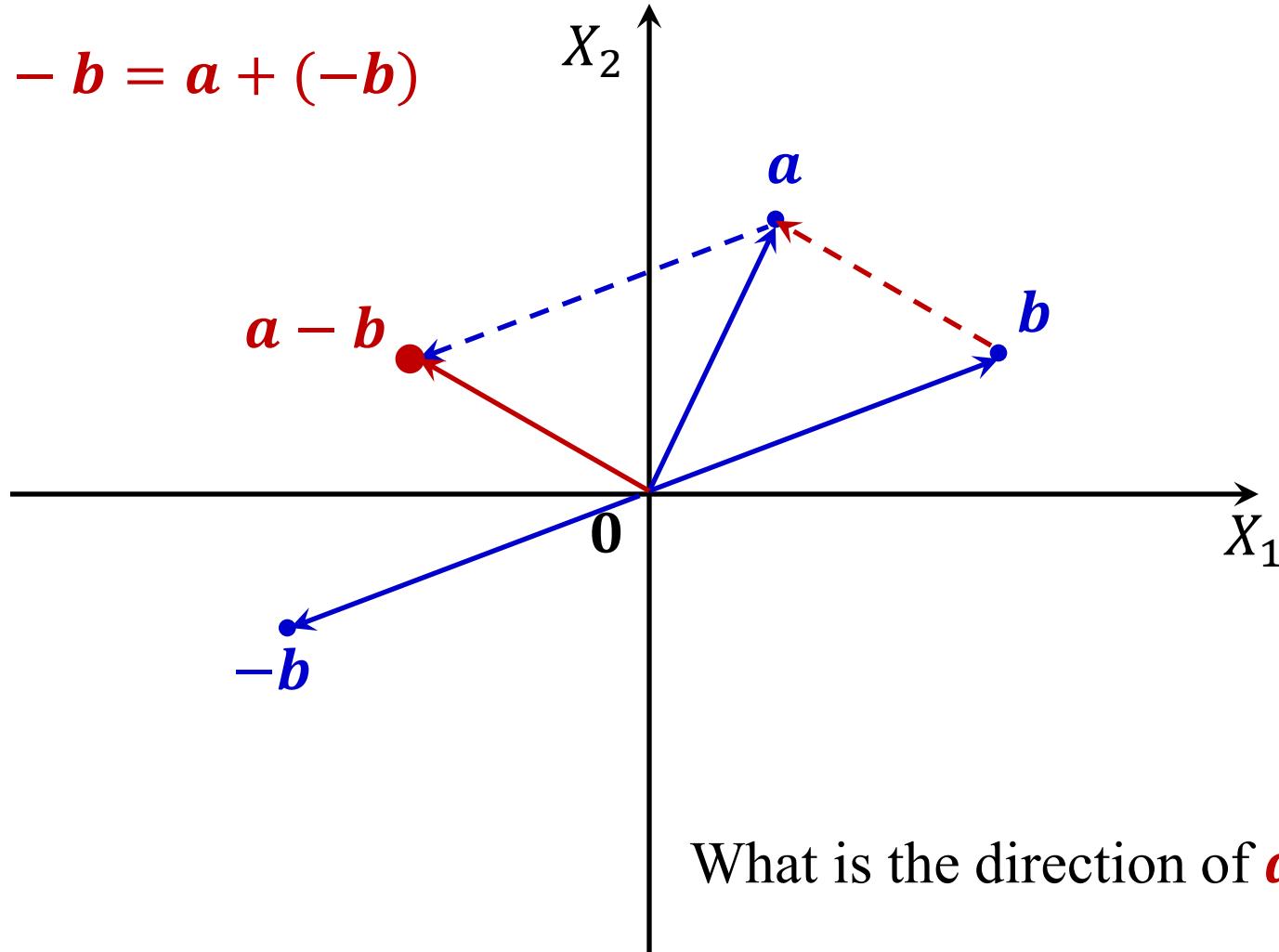
Event code  
**NESKSR**

# Review: Vector Addition

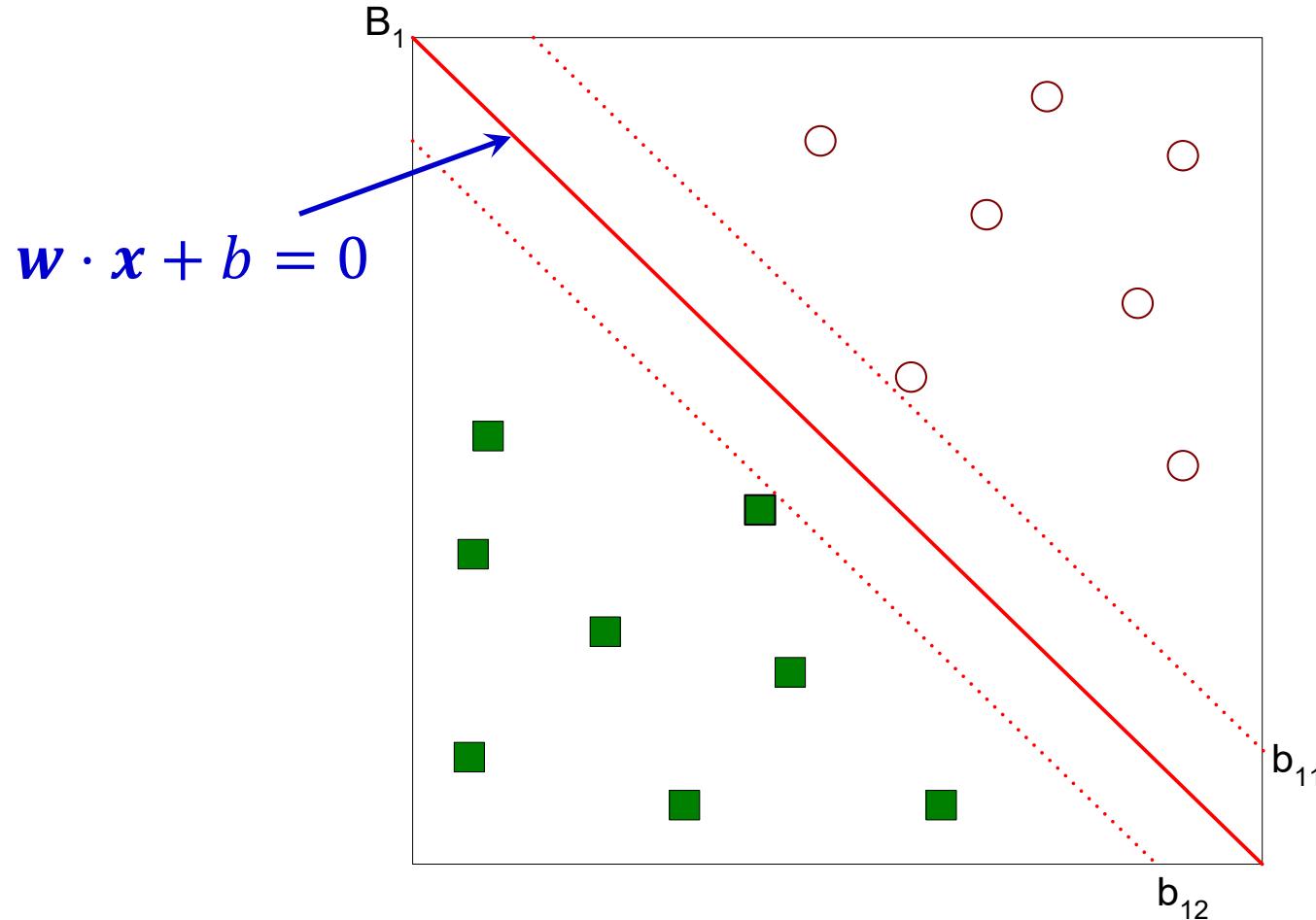


# Review: Vector Subtraction

$$\mathbf{a} - \mathbf{b} = \mathbf{a} + (-\mathbf{b})$$



# Making Predictions



For any test example  $x^*$ :  $\begin{cases} f(x^*) = +1, & \text{if } w \cdot x^* + b \geq 0 \\ f(x^*) = -1, & \text{if } w \cdot x^* + b < 0 \end{cases}$

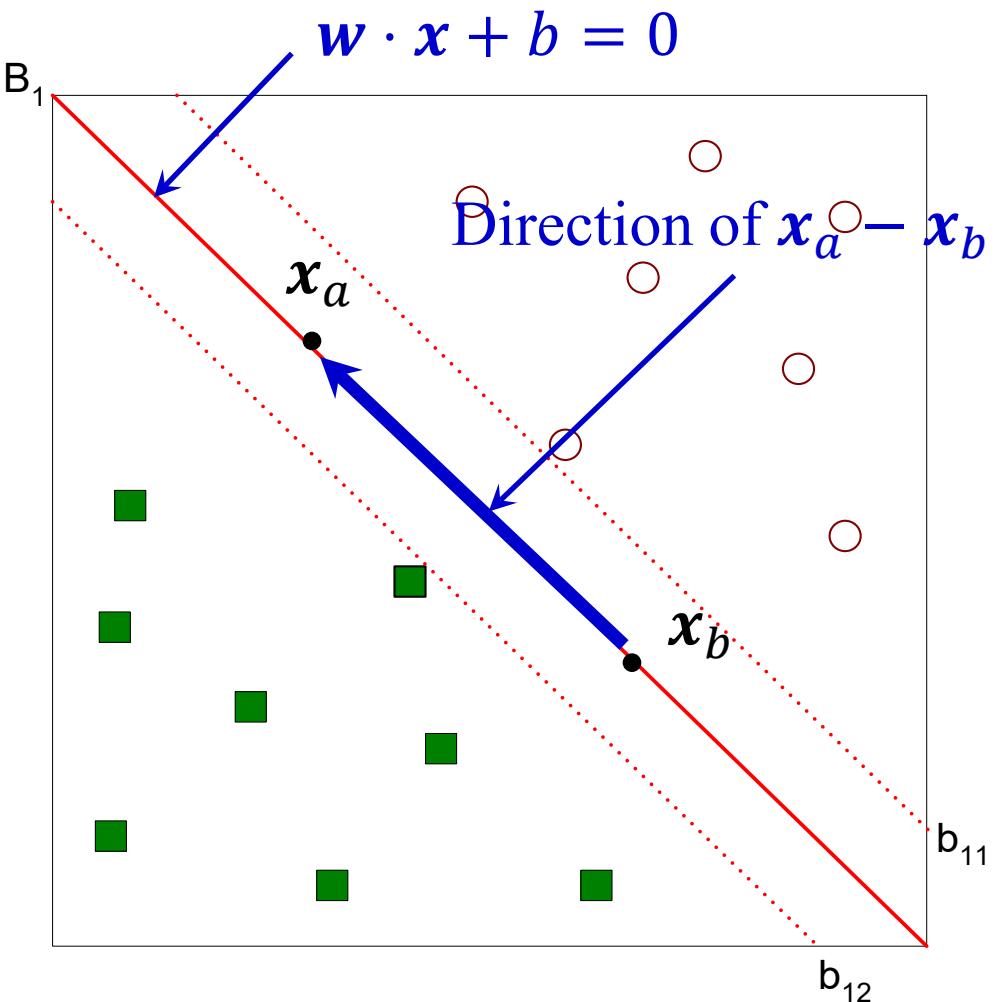
# Margin – Induction

- Suppose  $x_a$  and  $x_b$  are two points located on the decision boundary

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_a + b = 0 \\ \mathbf{w} \cdot \mathbf{x}_b + b = 0 \end{cases}$$



$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_b) = 0$$



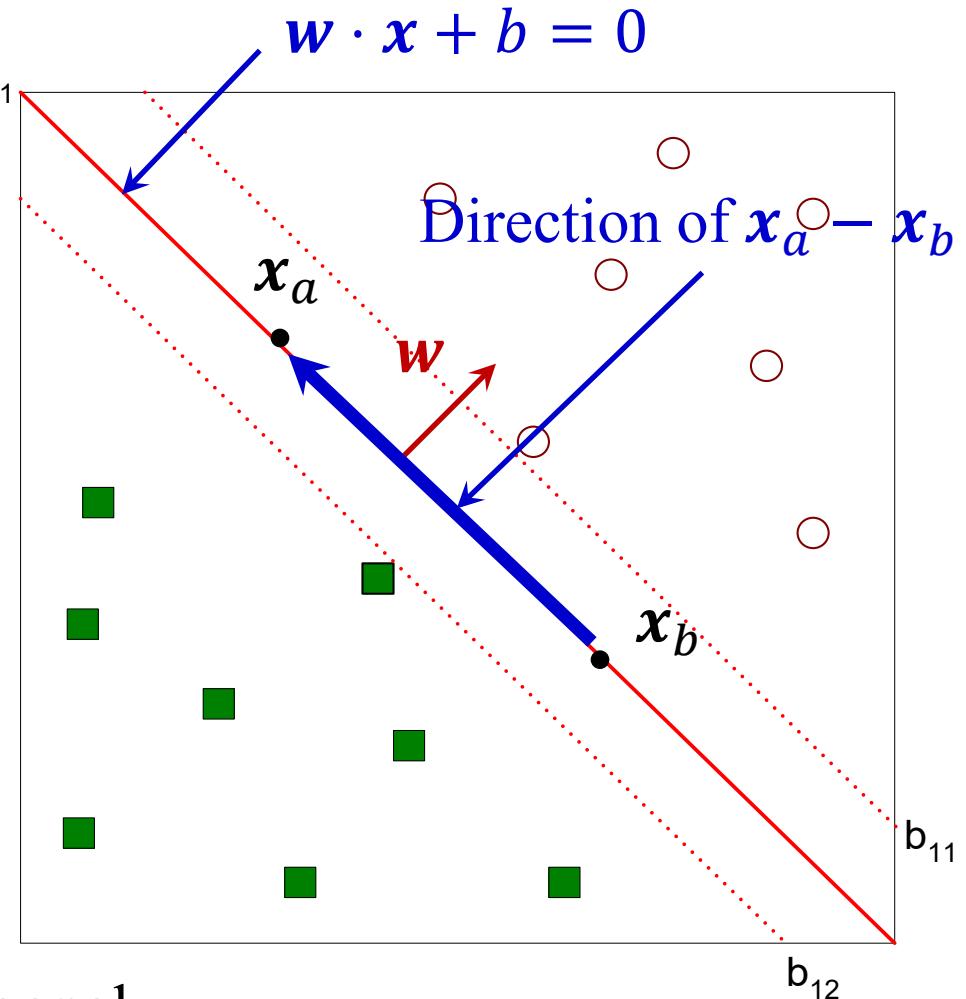
# Margin – Induction (cont.)

- Suppose  $x_a$  and  $x_b$  are two points located on the decision boundary,

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_a + b = 0 \\ \mathbf{w} \cdot \mathbf{x}_b + b = 0 \end{cases}$$

$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_b) = 0$$

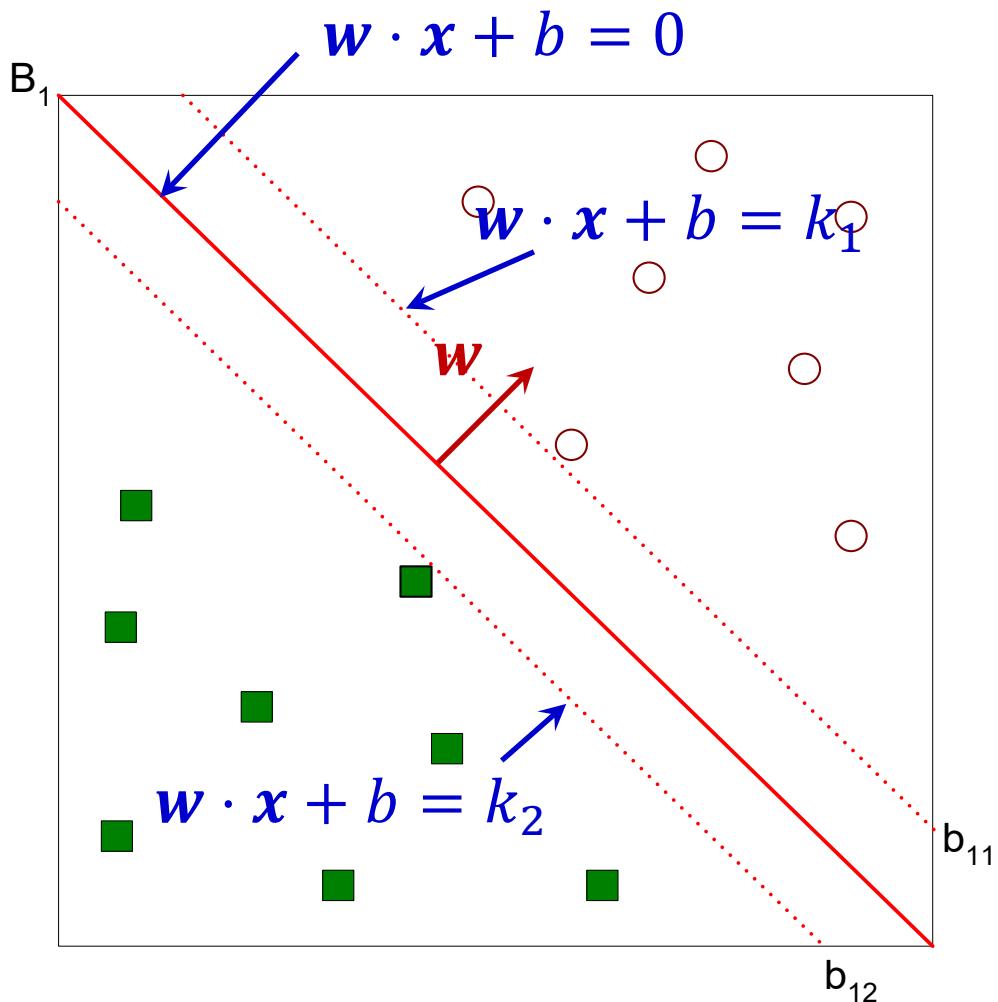
Based on definition  
of inner product



The direction of  $w$  is orthogonal  
to the decision boundary

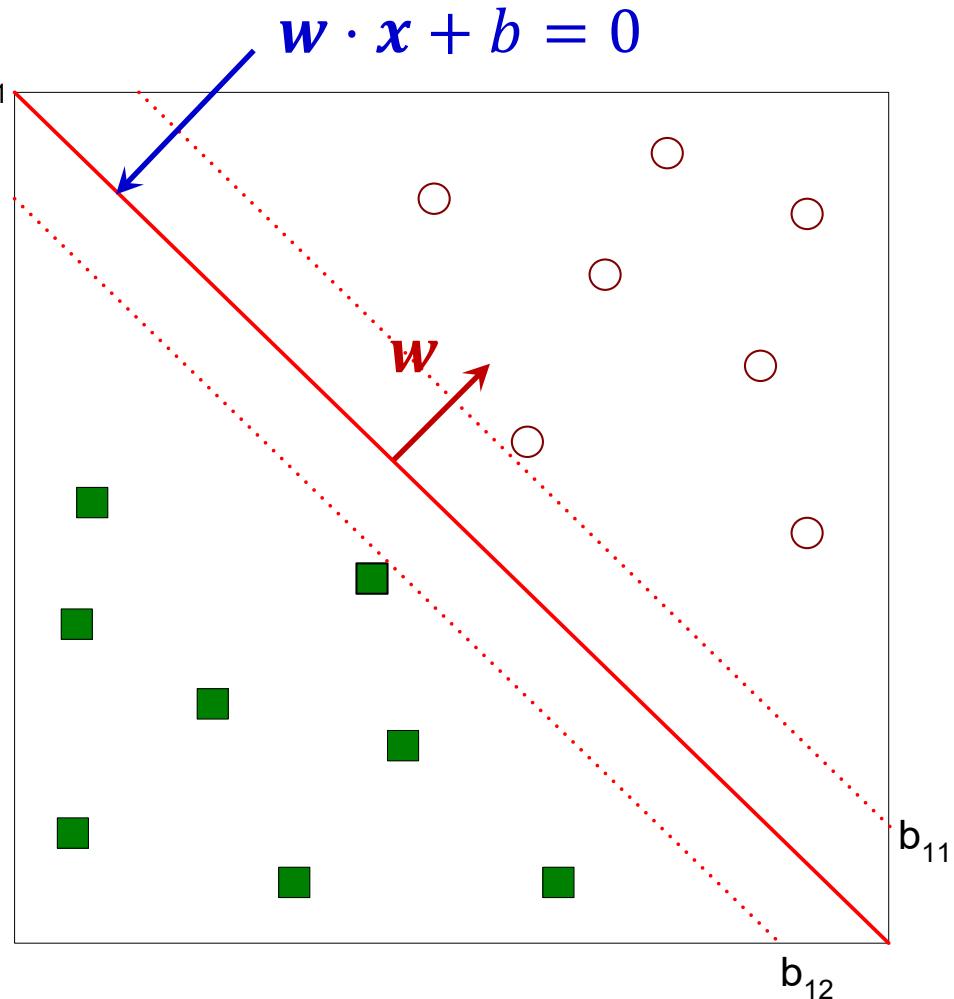
# Margin – Induction (cont.)

- Corollary: any two points on the same line parallel to the decision boundary has the same value for  $\mathbf{w} \cdot \mathbf{x} + b$
- $b$  is a free parameter I can set so that on the decision boundary  $\mathbf{w} \cdot \mathbf{x} + b = 0$



# Margin – Induction (cont.)

- For any circle  $x_c$  located above the decision boundary:  
$$\mathbf{w} \cdot \mathbf{x}_c + b \geq k,$$
 where  $k > 0$
- For any square  $x_s$  located below the decision boundary:  
$$\mathbf{w} \cdot \mathbf{x}_s + b \leq k',$$
 where  $k' < 0$



# Margin – Induction (cont.)

The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

$$\mathbf{w} \cdot \mathbf{x}_c + b = k, \text{ where } k > 0$$

$$\mathbf{w} \cdot \mathbf{x}_s + b = k', \text{ where } k' < 0$$

It can be shown that, these two parallel hyperplanes can be further rewritten as

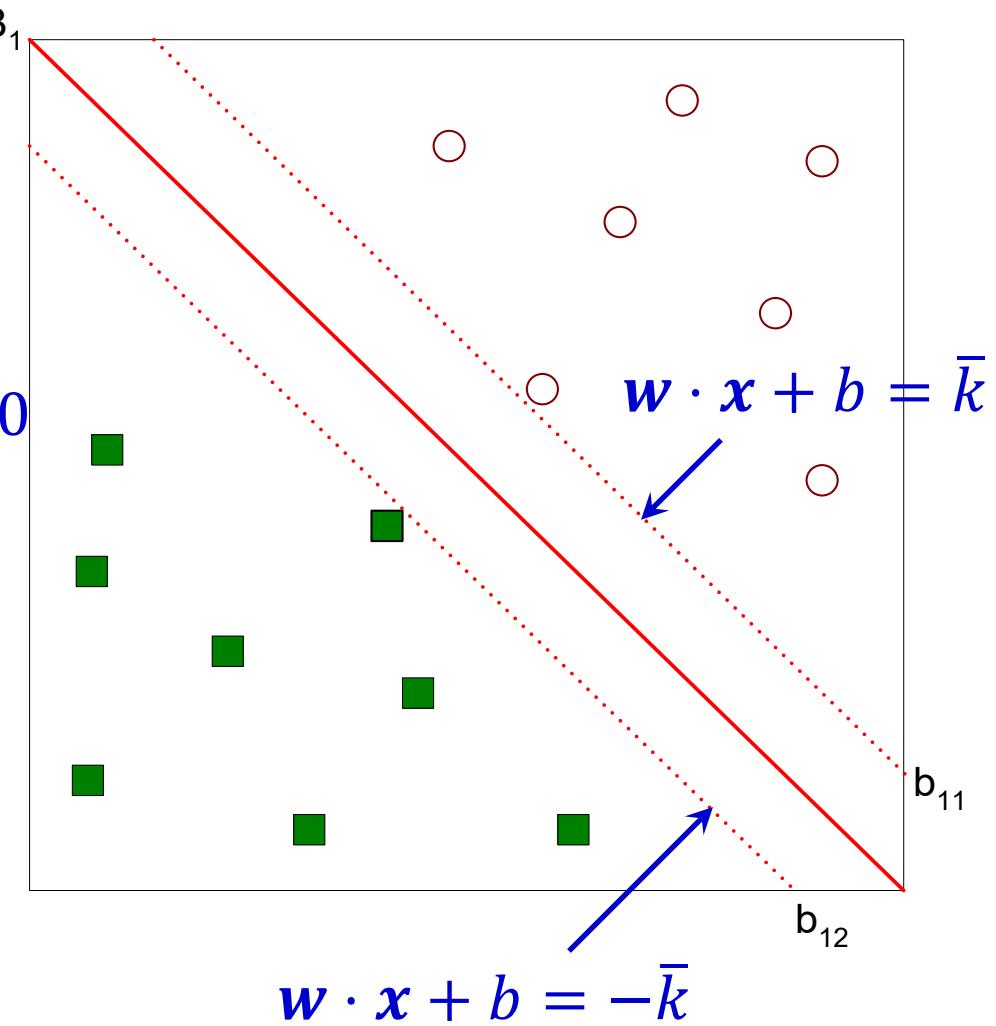
$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = \bar{k}$$

$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = -\bar{k}$$

$$\text{where } \bar{k} > 0$$



Tutorial



# Margin – Induction (cont.)

The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = \bar{k} \quad \text{where } \bar{k} > 0$$

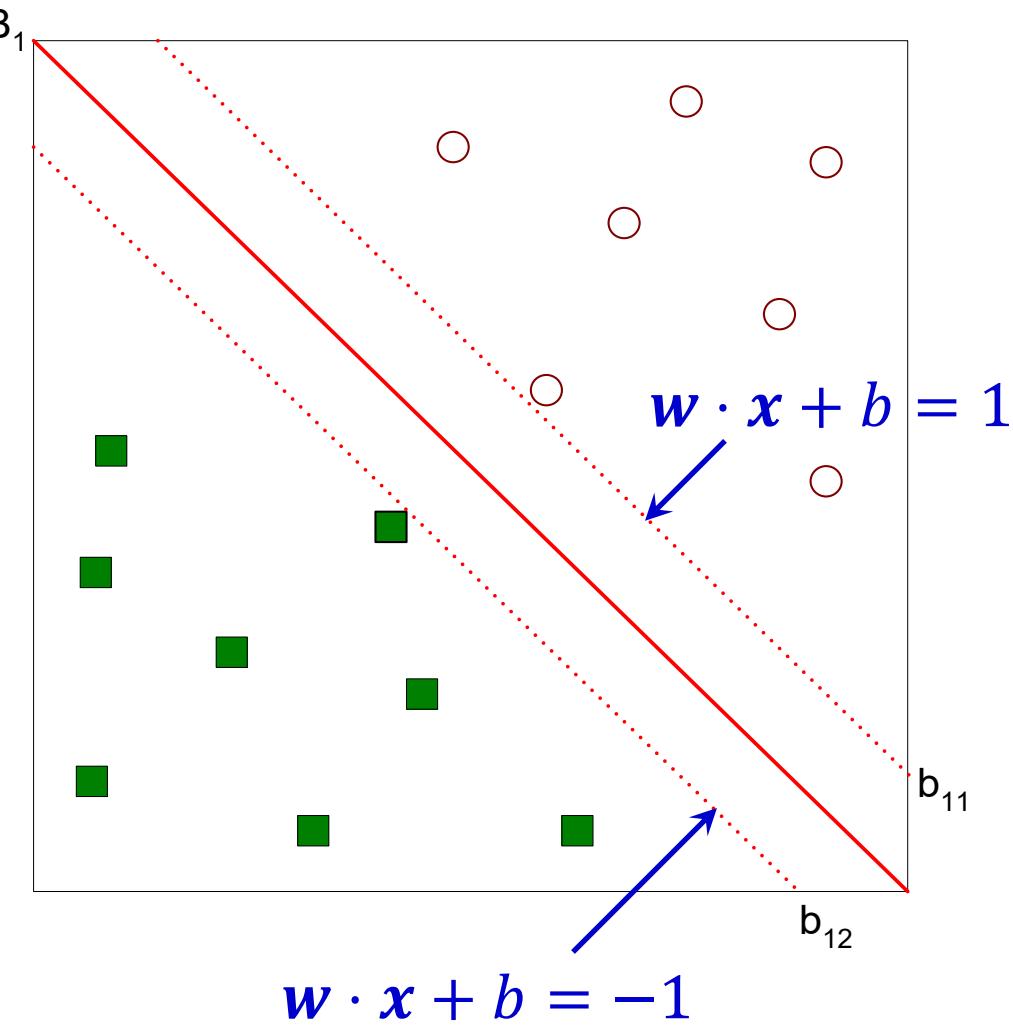
$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = -\bar{k}$$

$$\mathbf{w} = \frac{\mathbf{w}}{\bar{k}} \quad b = \frac{\bar{b}}{\bar{k}}$$

After rescaling  $\mathbf{w}$  and  $b$ , the two parallel hyperplanes can be further rewritten as

$$\mathbf{w} \cdot \mathbf{x} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x} + b = -1$$



# Margin – Induction (cont.)

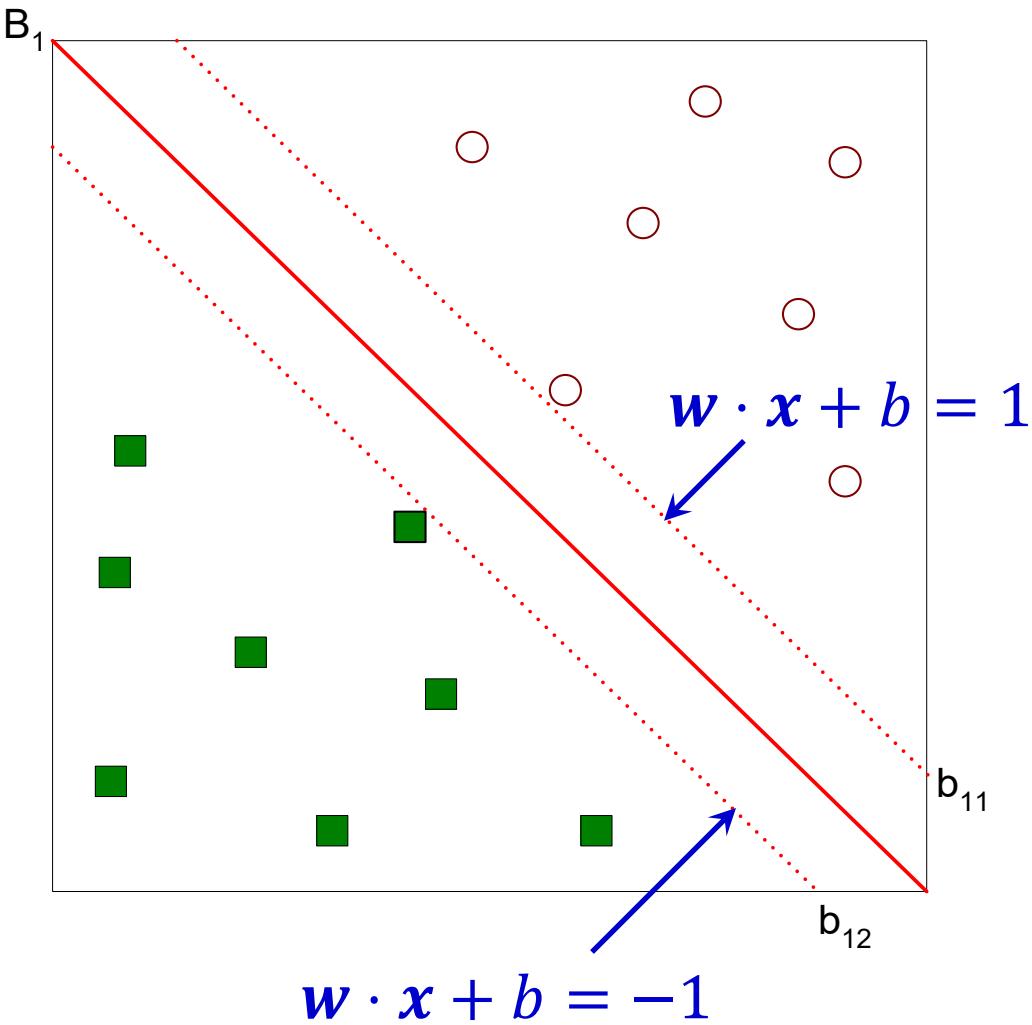
$$\mathbf{w} = \frac{\mathbf{w}}{\bar{k}} \quad b = \frac{\bar{b}}{\bar{k}}$$

After rescaling  $\mathbf{w}$  and  $b$ , the two parallel hyperplanes can be further rewritten as

$$\mathbf{w} \cdot \mathbf{x} + b = 1$$

$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

We should've used different letters for the rescaled  $\mathbf{w}$  and  $b$ , but we abuse notations for simplicity.



# Margin – Induction (cont.)

$$b_{11}: \mathbf{w} \cdot \mathbf{x}_1 + b = 1$$

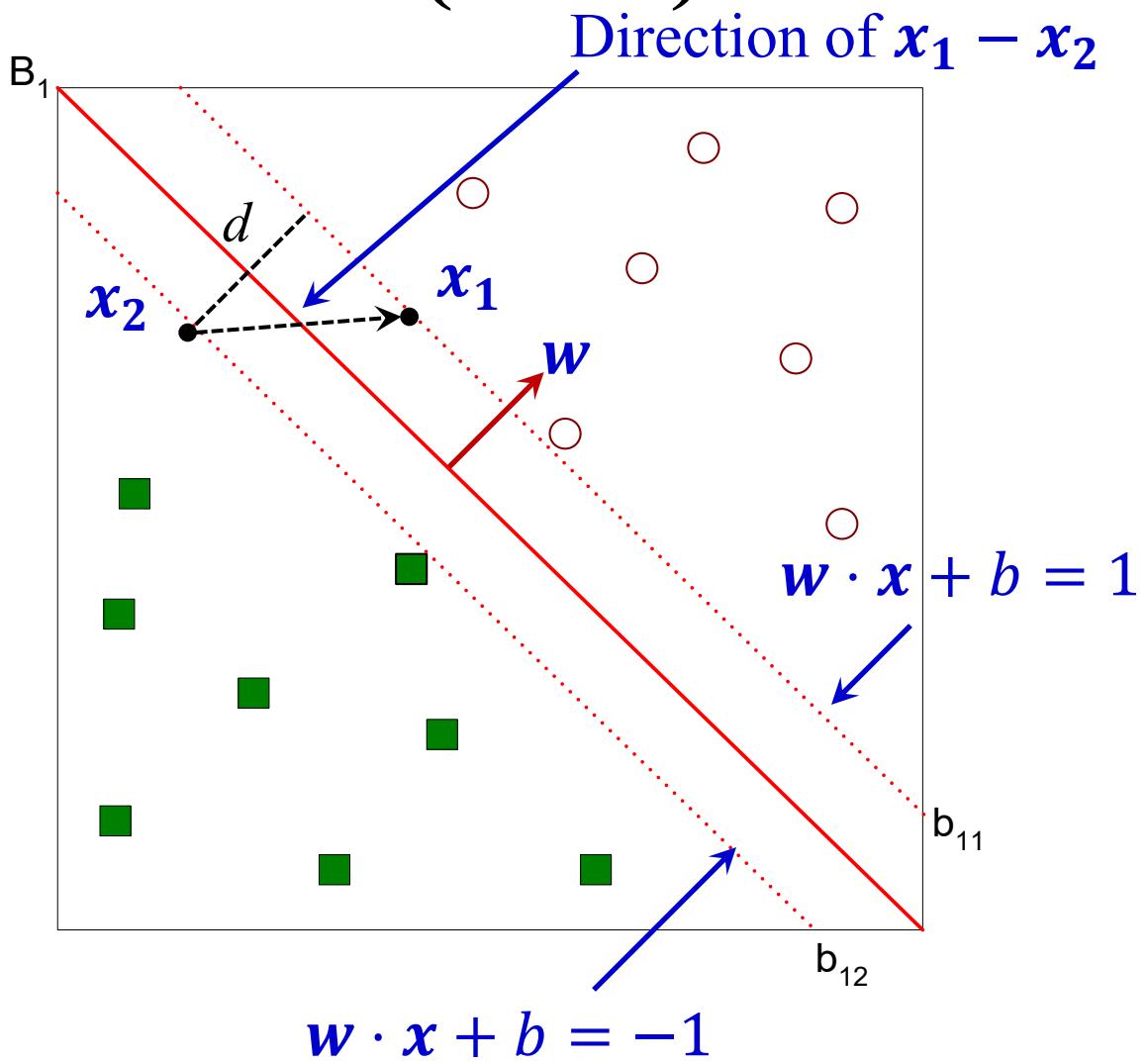
$$b_{12}: \mathbf{w} \cdot \mathbf{x}_2 + b = -1$$

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

Based on definition  
of inner product

$$\|\mathbf{w}\|_2 d = 2$$

Margin



# Margin – Induction (cont.)

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

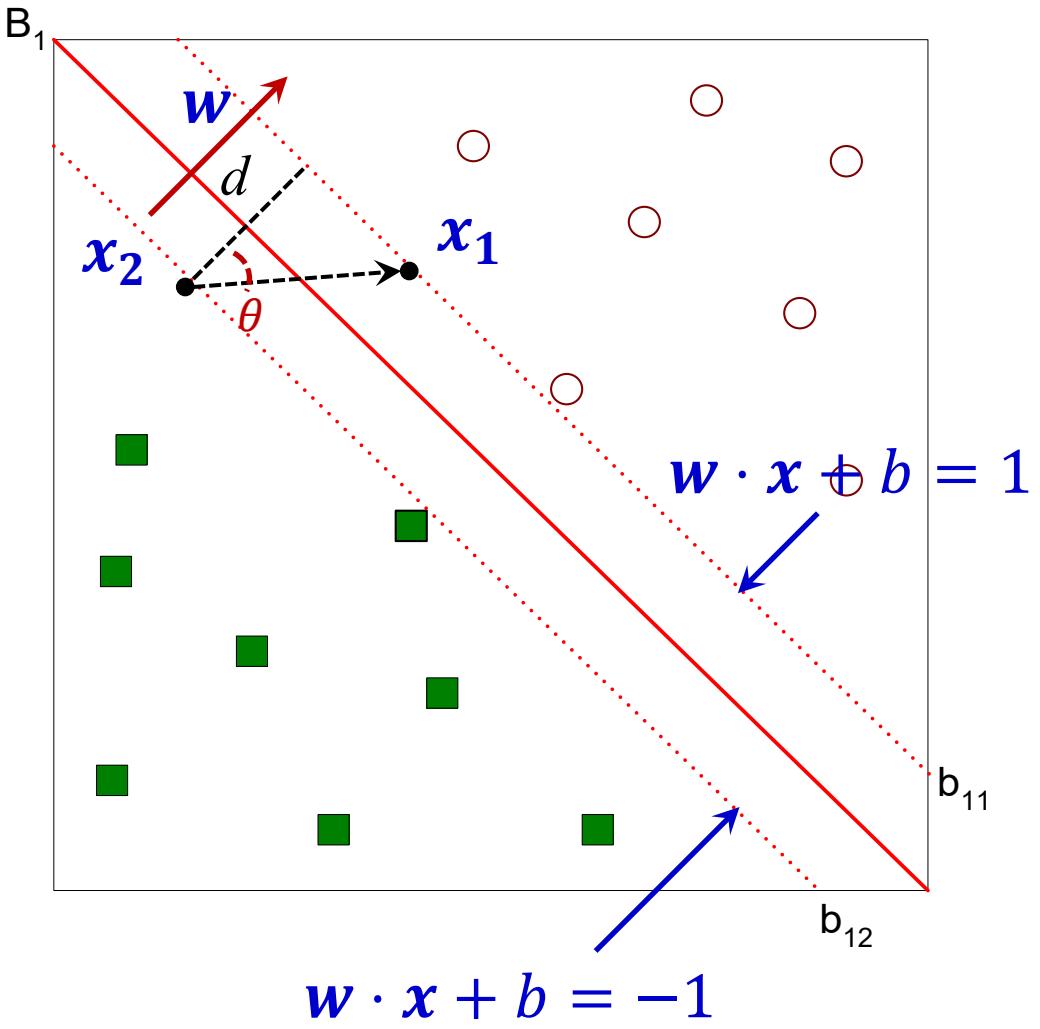
↓ Based on definition  
of inner product

$$\|\mathbf{w}\|_2 \times \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \times \cos(\theta) = 2$$

Based on definition  
of  $\cos(\cdot)$  =  $d$

$$\|\mathbf{w}\|_2 \times d = 2$$

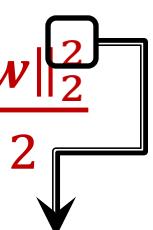
Important: this holds only  
for the rescaled  $\mathbf{w}$ .



# Margin Maximization

$$\|\mathbf{w}\|_2 \times d = 2 \implies d = \frac{2}{\|\mathbf{w}\|_2} \quad \|\mathbf{w}\|_2^2 = \sum_{i=1}^d (w_i \times w_i)$$

Maximize margin  $d = \frac{2}{\|\mathbf{w}\|_2}$   $\longrightarrow$  Minimize  $\frac{\|\mathbf{w}\|_2^2}{2}$



Constraints:  $\mathbf{w} \cdot \mathbf{x}_i + b \geq 1$ , if  $y_i = 1$

$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1$ , if  $y_i = -1$

OR

$y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$

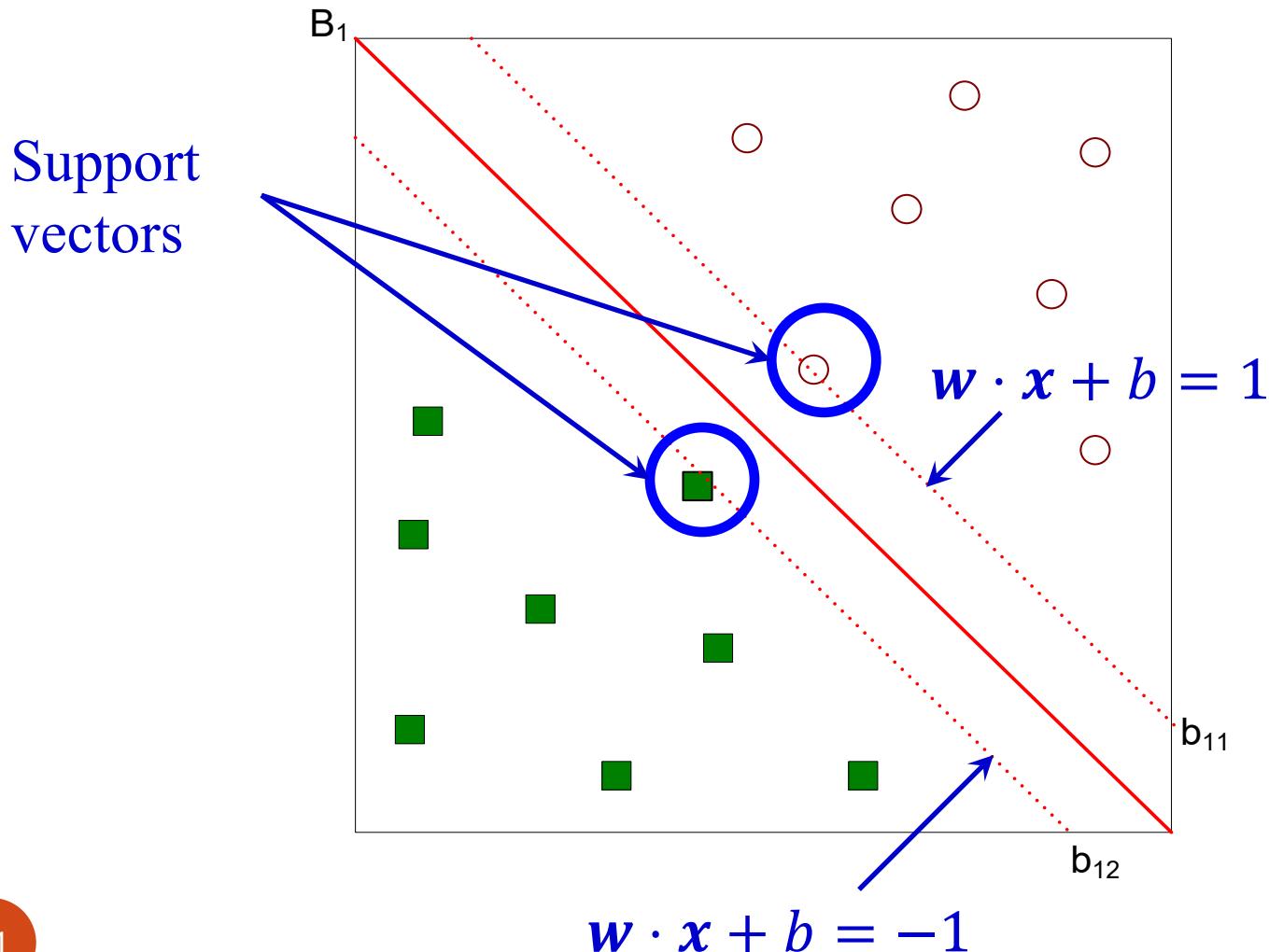
# Optimization Problem for SVMs

- Optimization problem of linear SVMs

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{\|\mathbf{w}\|_2^2}{2} \\ \text{s.t.} \quad & y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, N \end{aligned}$$

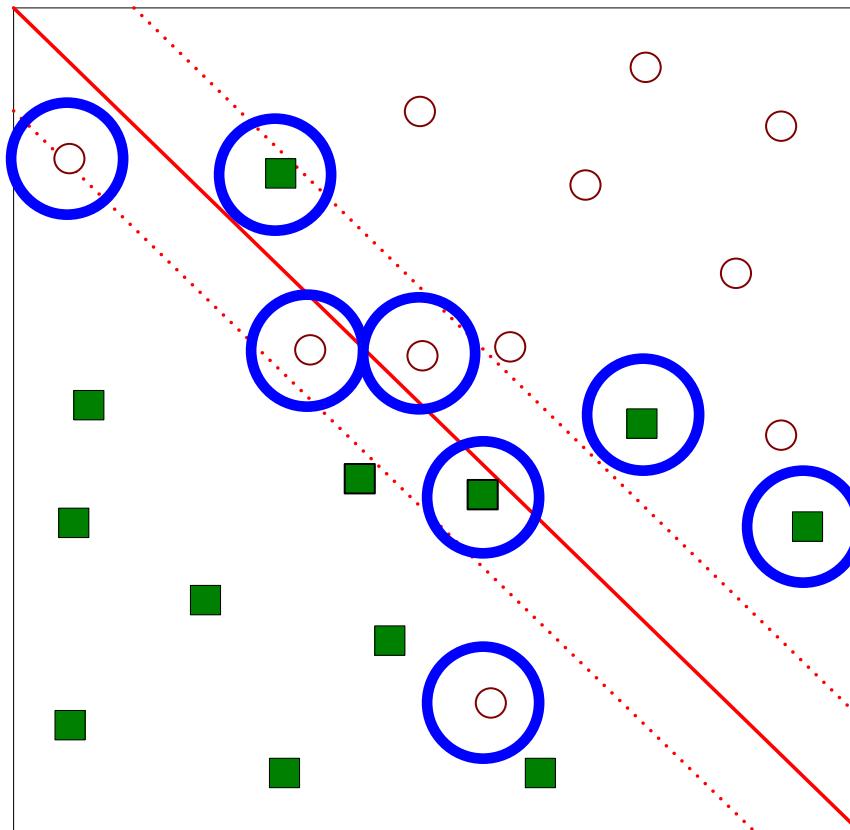
- The optimization is convex
  - Many numerical approaches can be applied to find the solution
  - Convex is easy. Non-convex is hard.
  - The exact optimization algorithms are beyond the scope of this course.

# Support Vectors



# Non-separable Case

- What if data of two classes cannot be perfectly separated?



Slack variables  
need to be  
introduced to  
absorb errors

# Implementation Example

```
>>> from sklearn import svm
```

• • •

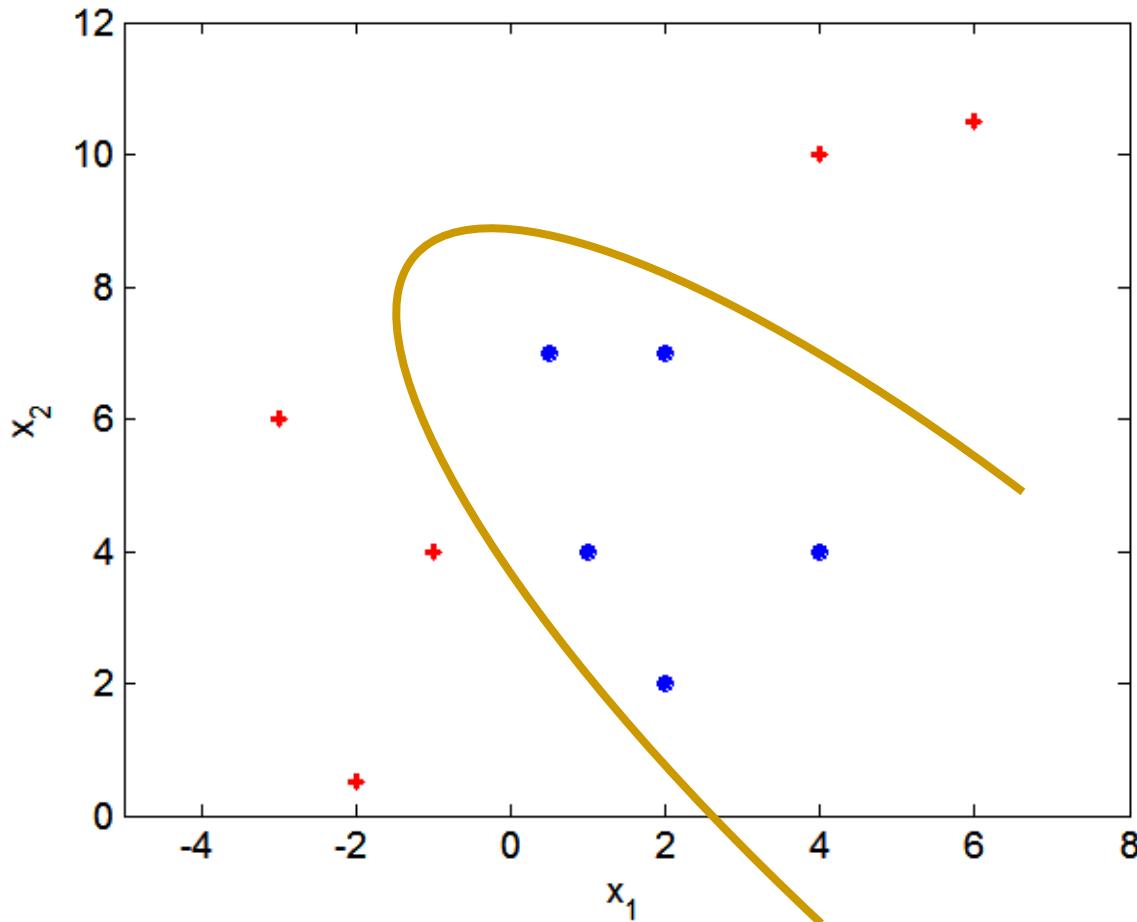
```
>>> svmC = svm.LinearSVC()
```

```
>>> svmC.fit(X, y)
```

```
>>> pred= svmC.predict(X)
```

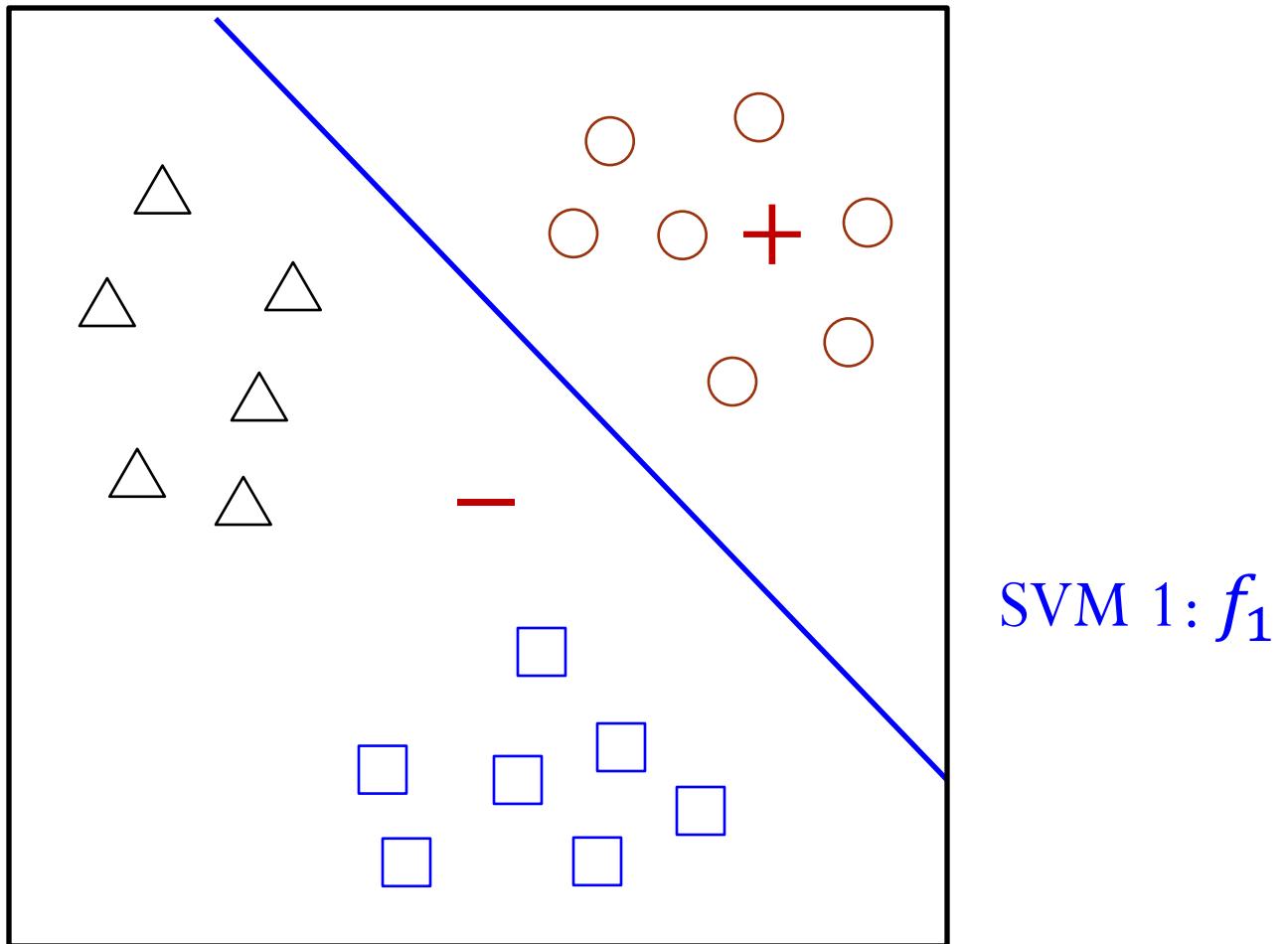
# Nonlinear SVMs

- What if decision boundary is not linear?

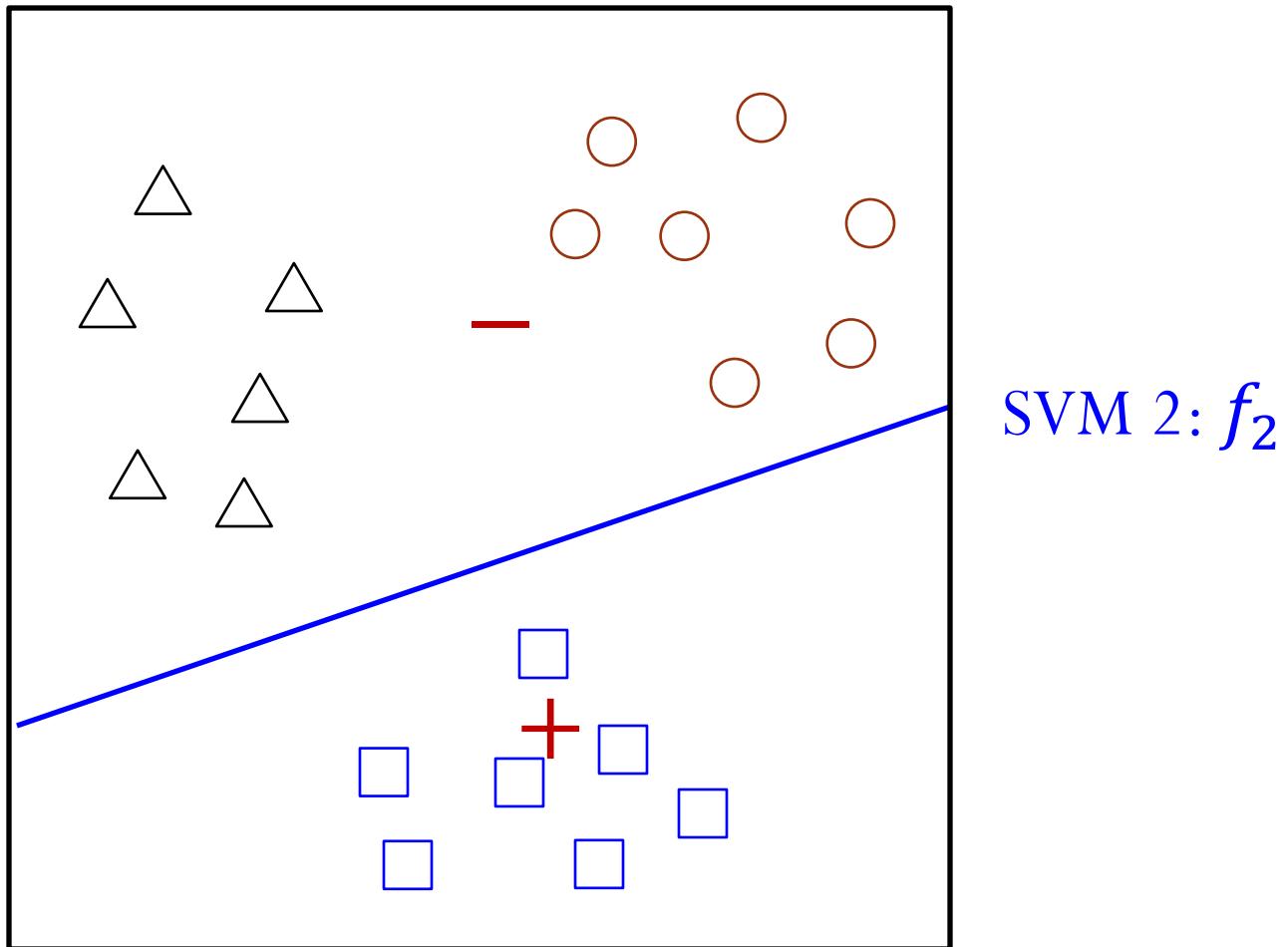


Kernel trick in  
the dual form

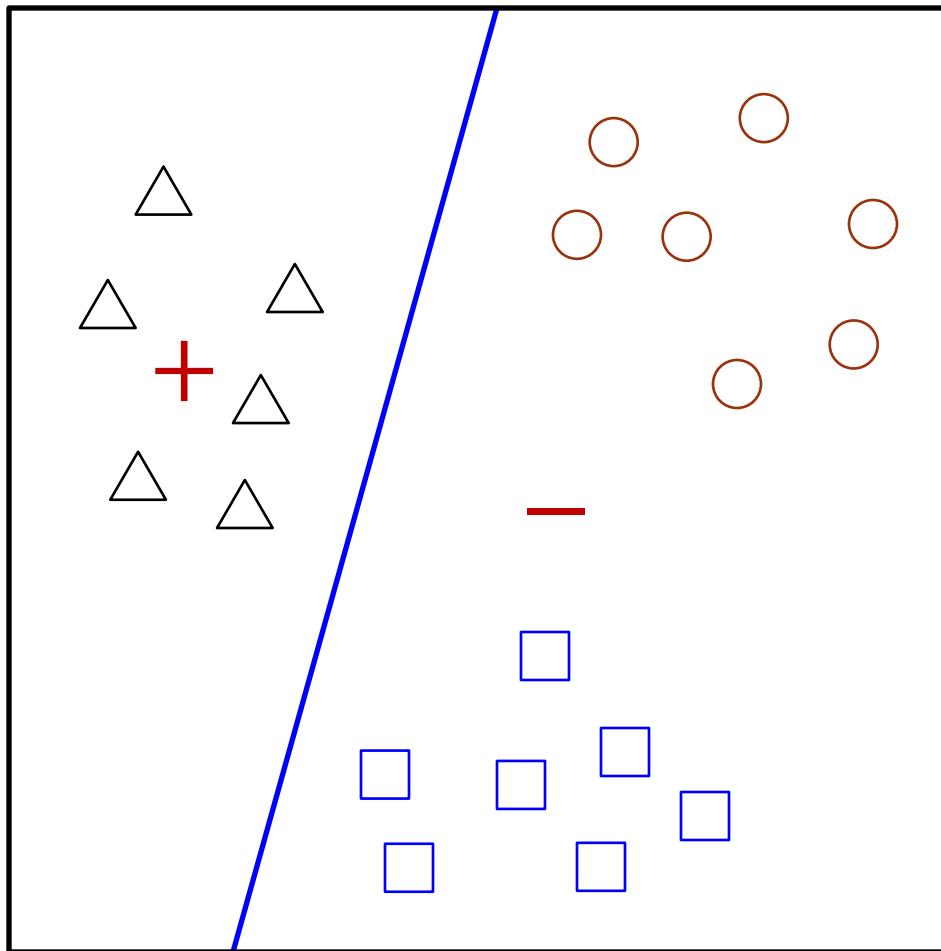
# Multi-Class Classification



# Multi-Class Classification (cont.)



# Multi-Class Classification (cont.)



# Multi-Class Classification (cont.)

- Give a 3-class classification problem:  $C_1$ ,  $C_2$  &  $C_3$
- General approaches: 1 v.s. rest
  - Binary classification 1: positive ( $C_1$ ) v.s. negative ( $C_2 \& C_3$ )
  - Binary classification 2: positive ( $C_2$ ) v.s. negative ( $C_1 \& C_3$ )
  - Binary classification 3: positive ( $C_3$ ) v.s. negative ( $C_1 \& C_2$ )
  - For a test instance  $\mathbf{x}^*$ , apply binary classifier  $f_1$ ,  $f_2$ , and  $f_3$  to make predictions on  $\mathbf{x}^*$
  - Combine predicted results of  $f_1(\mathbf{x}^*)$ ,  $f_2(\mathbf{x}^*)$ , and  $f_3(\mathbf{x}^*)$  to make a final prediction



# Linear SVMs for Multi-Class

- $f_i$  only generates  $-1/1$ :  
1: belong to  $C_i$ , and  $-1$ : not belong to  $C_i$
- Given a test data  $\boldsymbol{x}^*$ , suppose

	$C_1$	$C_2$	$C_3$
$f_1(\boldsymbol{x}^*) = -1$	0	1	1
$f_2(\boldsymbol{x}^*) = 1$	0	1	0
$f_3(\boldsymbol{x}^*) = -1$	1	1	0
<b>Total Votes:</b>	1	3	1



# CZ4041/SC4000: Machine Learning

## Lesson 8b: Linear Regression

LI Boyang, Albert

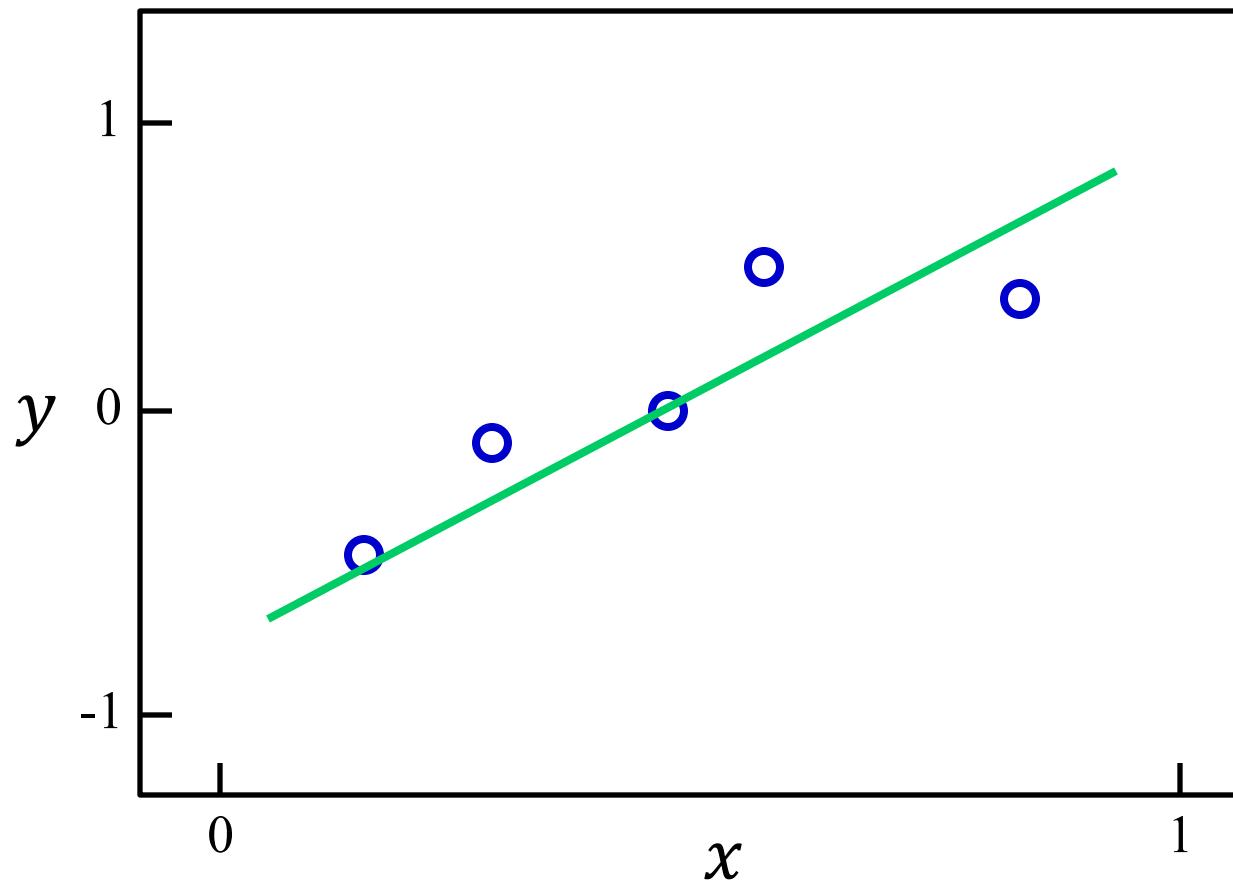
School of Computer Science and Engineering,  
NTU, Singapore

Acknowledgements: Slides are modified from the version prepared by Dr. Sinno Pan.

# Regression

- What is the stock price tomorrow?
- What is the position of my phone?
- How severe is a patient's heart problem?
  - no disease (0), very mild (1), mild (2), severe (3), immediate danger (4)
- Difference with classification: Error metric

# Linear Fitting



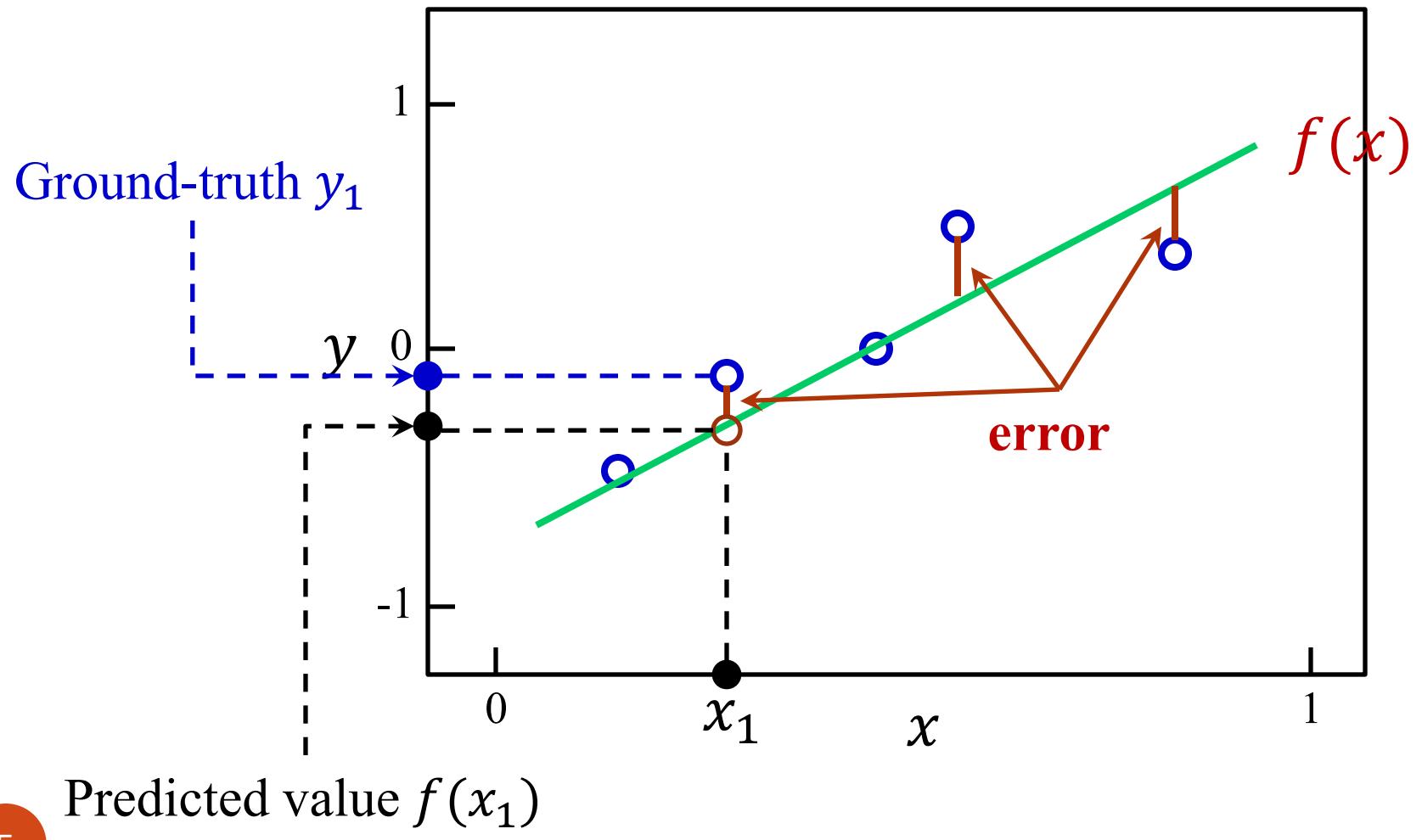
# Linear Regression: 1D Case

- A special case, an instance is represented by one input feature
- To learn a linear function  $f(x)$  in terms of  $w$  (drop bias term  $b$  for simplicity) from  $\{x_i, y_i\}, i = 1, \dots, N$

$$f(x) = w \cdot x$$

- Such that the difference (i.e., error) between the predicted values  $f(x_i)$ 's and the ground-truth values  $y_i$ 's is as small as possible

# Linear Regression Model (cont.)



# Linear Regression Model (cont.)

- Suppose sum-of-squares error is used

$$\mathcal{L}(w) = \frac{1}{2} \sum_{i=1}^N (f(x_i) - y_i)^2 = \frac{1}{2} \sum_{i=1}^N (wx_i - y_i)^2$$

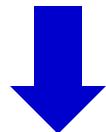
- Learn the linear model in terms of  $w$  by minimizing the error

$$w^* = \arg \min_w \mathcal{L}(w)$$

# Linear Regression Model (cont.)

- To solve the unconstrained minimization problem, we can set the derivative of  $E(w)$  w.r.t.  $w$  to zero

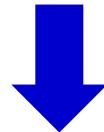
$$\frac{\partial \mathcal{L}(w)}{\partial w} = \frac{\partial \left( \frac{1}{2} \sum_{i=1}^N (wx_i - y_i)^2 \right)}{\partial w} = 0$$



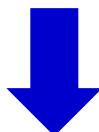
$$\sum_{i=1}^N (wx_i - y_i) x_i = 0$$

# Linear Regression Model (cont.)

$$\sum_{i=1}^N (wx_i - y_i) x_i = 0$$



$$w \sum_{i=1}^N x_i^2 - \sum_{i=1}^N y_i x_i = 0$$



$$w = \frac{\sum_{i=1}^N y_i x_i}{\sum_{i=1}^N x_i^2}$$

# Linear Regression Model (cont.)

- The derivative  $\frac{\partial \mathcal{L}(w)}{\partial w} = 0$  is a necessary, but not sufficient, condition for  $\mathcal{L}(w)$  to take the minimum.
- The second-order derivative must be greater than 0.
- What is the second-order derivative?

$$\frac{\partial \left( \frac{1}{2} \sum_{i=1}^N (wx_i - y_i)^2 \right)}{\partial w} = w \sum_{i=1}^N x_i^2 - \sum_{i=1}^N y_i x_i$$

$$\frac{\partial^2 \left( \frac{1}{2} \sum_{i=1}^N (wx_i - y_i)^2 \right)}{\partial w^2} =$$

# Linear Regression Model (cont.)

- What is the second-order derivative?

$$\frac{\partial \left( \frac{1}{2} \sum_{i=1}^N (wx_i - y_i)^2 \right)}{\partial w} = w \sum_{i=1}^N x_i^2 - \sum_{i=1}^N y_i x_i$$



Go to [wooclap.com](https://wooclap.com) and enter the event code in the top banner:

Event code  
**NESKSR**

# Linear Regression Model (cont.)

- To learn a linear function  $f(\mathbf{x})$  in more general form in terms of  $\mathbf{w}$  and  $b$ ,

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

- By defining  $w_0 = b$ , and  $X_0 = 1$ ,  $\mathbf{w}$  and  $\mathbf{x}$  are of  $d + 1$  dimensions

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$$

# Linear Regression Model (cont.)

- Suppose sum-of-squares error is used

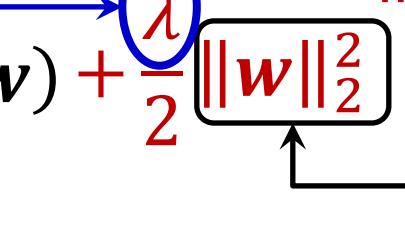
$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2 = \frac{1}{2} \sum_{i=1}^N (\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$$

- Learn the linear model in terms of  $\mathbf{w}$  by minimizing the error

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Positive tradeoff parameter

$$\|\mathbf{w}\|_2^2 = \mathbf{w} \cdot \mathbf{w}$$



A regularization term to control the complexity of the model



# Regularized Linear Regression

- To solve the unconstrained minimization problem, we can set the derivative of  $\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$  w.r.t.  $\mathbf{w}$  to zero

$$\frac{\partial \left( \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)}{\partial \mathbf{w}} = \mathbf{0}$$

- We can obtain a closed-form solution for  $\mathbf{w}$  by the above equations

# Closed-Form Solution

- Denote by  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ x_{10} & \cdots & x_{N0} \\ \vdots & \ddots & \vdots \\ x_{1d} & \cdots & x_{Nd} \end{pmatrix}^T = \begin{pmatrix} \mathbf{x}_1 \\ x_{10} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{Nd} \end{pmatrix}$$

- And by  $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$

- The closed-form solution for  $\mathbf{w}$ :

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

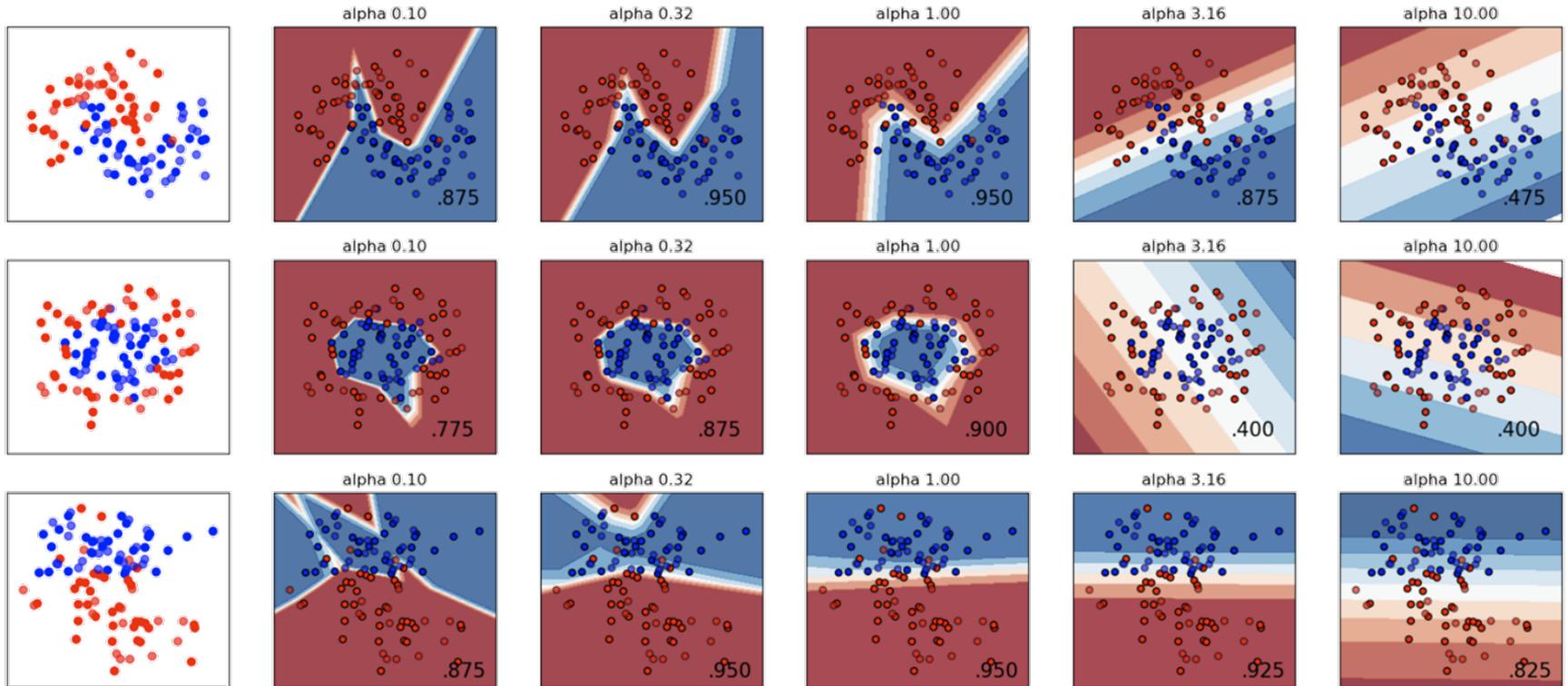
How to induce  
this closed-form  
solution?

Tutorial



# L2 Regularization

- The  $\frac{\lambda}{2} \|\mathbf{w}\|_2^2$  term encourages simpler models with small weight norms and is widely used.



Example in MLP with coefficient  $\alpha$

([https://scikit-learn.org/stable/auto\\_examples/neural\\_networks/plot\\_mlp\\_alpha.html](https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mlp_alpha.html))

# Implementation Example

```
>>> from sklearn.linear_model import Ridge
```

```
...
```

Referred to as  $\lambda$  on  
our lecture notes

```
>>> rlr = Ridge(alpha=0.1)
```

```
>>> rlr.fit(X, y)
```

```
>>> pred_train_rlr= rlr.predict(X)
```

# Evaluation

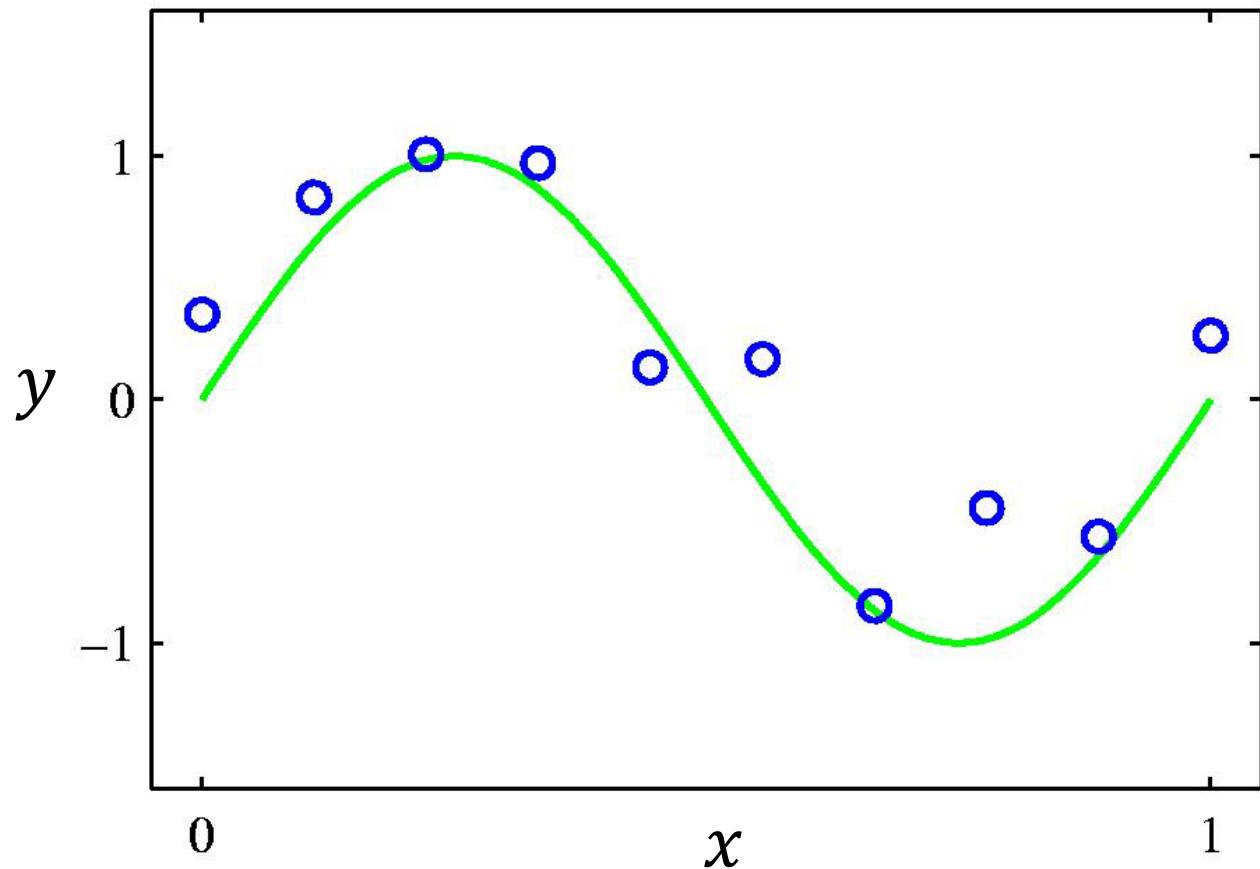
- Root Mean Square Error (RMSE)

$$\sqrt{\frac{1}{N} \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2}$$

- Mean Absolute Error (MAE)

$$\frac{1}{N} \sum_{i=1}^N |f(\mathbf{x}_i) - y_i|$$

# Nonlinear Fitting



Kernel trick