Spring 2023

SWCON253: Machine Learning

Lecture 06
Perceptron

Jinwoo Choi
Assistant Professor
CSE, Kyung Hee University

Contents

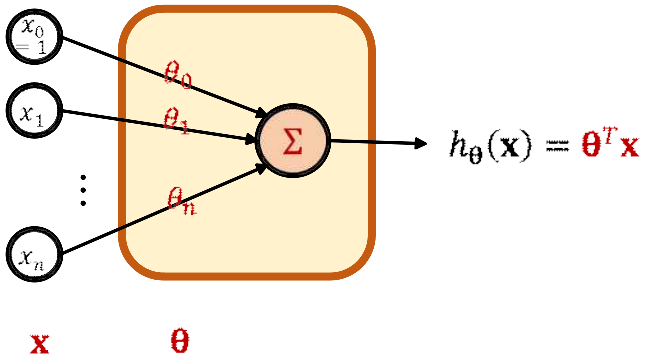
References

- *패턴 인식* by 오일석, *기계 학습* by 오일석
- *Intro to Deep Learning & Generative Models* by Sebastian Raschka (http://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2020/)

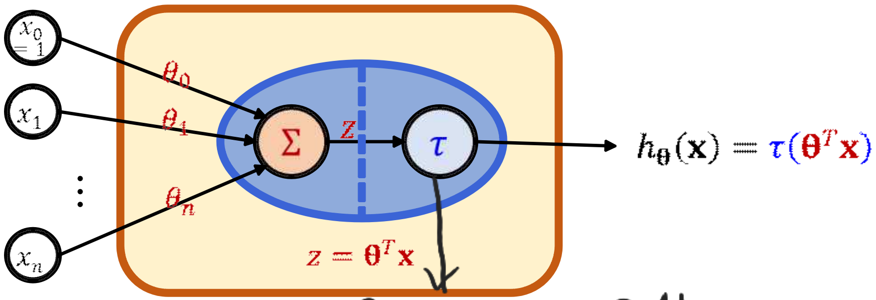
(Recap.) Extending Linear Regression for Binary Classification

- For *Binary classification*, we can extend *Linear regression* by adding some *Activation function* $\tau()$.

Linear Regression

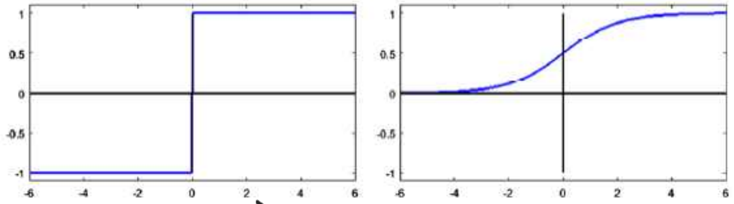


Linear Regression + Threshold function



R에서 [0, 1]로 Mapping

- Examples of $\boldsymbol{\tau}(z)$
  - ★ *step* function: ***hard*** threshold
  - ★ *sigmoid* function: ***soft*** threshold



(a) 계단 함수(Sign) (b) 로지스틱 시그모이드

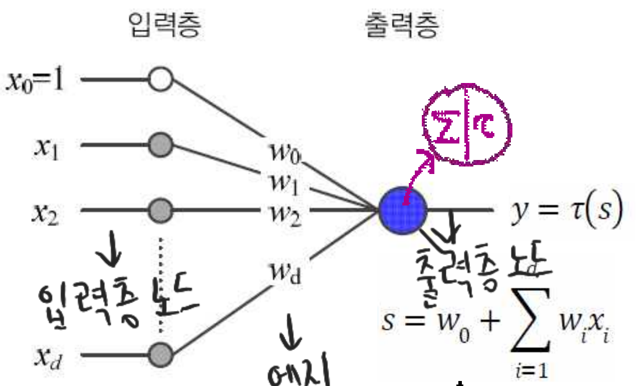
~> Perceptron ~> ~~Logistic Regression~~ Logistic Regression

1. Perceptron

1. Model & Terminology
2. Learning Problem
3. Cost Function
4. Gradient
5. Learning Algorithm

Perceptron – *Model & Terminology*

- 퍼셉트론의 구조와 동작
  - **입력층**의 i번째 **노드**는 특징 벡터 $\mathbf{x} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}}$의 i번째 요소 x_i를 담당
  - **출력층**은 한 개의 노드
  - i번째 입력층 노드와 출력층을 연결하는 **에지**는 **가중치** w_i를 가짐
  - 해당하는 특징값과 가중치를 곱한 결과를 모두 더하여 s를 구하고, **활성함수** τ를 적용함

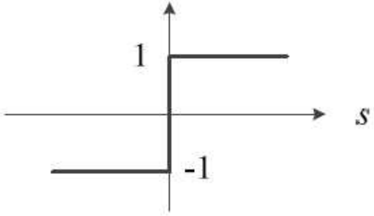


$= \sum_{i=0}^{d} w_i x_i \quad (x_0 = 1)$

(a) 퍼셉트론의 구조

$$\tau(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$$

불연속 함수

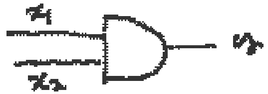


(b) 계단함수를 활성함수 $\tau(s)$로 이용함

Perceptron – *Example*

◆ Logical OR Gate

-※ OR Gate



x_1	x_2	y	
0	0	0	(-1)
0	1	1	
1	0	1	
1	1	1	

2차원 특징 벡터로 표현되는 샘플을 4개 가진 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, $\mathbb{Y} = \{y_1, y_2, y_3, y_4\}$를 생각하자. [그림 3-4(a)]는 이 데이터를 보여준다.

$$\mathbf{x}_1 = \begin{pmatrix}0\\0\end{pmatrix},\ y_1 = -1,\quad \mathbf{x}_2 = \begin{pmatrix}1\\0\end{pmatrix},\ y_2 = 1,\quad \mathbf{x}_3 = \begin{pmatrix}0\\1\end{pmatrix},\ y_3 = 1,\quad \mathbf{x}_4 = \begin{pmatrix}1\\1\end{pmatrix},\ y_4 = 1$$

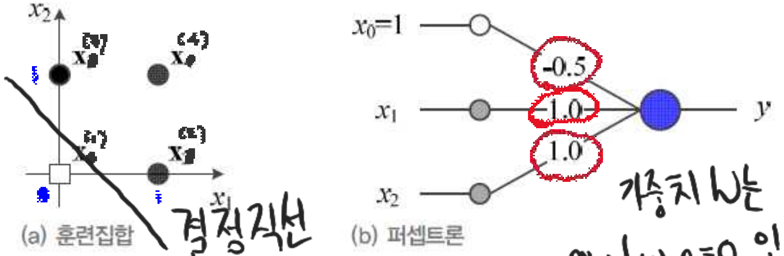


그림 3-4 OR 논리 게이트를 이용한 퍼셉트론의 동작 예시

가중치 w는 $ax+by+c=0$ 인 활성함수의 계수

$\rightarrow x_2 = -x_1 + 0.5$

샘플 4개를 하나씩 입력하여 제대로 분류하는지 확인해 보자.

$$\begin{aligned}
&\mathbf{x}_1:\ s = -0.5 + 0 * 1.0 + 0 * 1.0 = -0.5, && \tau(-0.5) = -1\\
&\mathbf{x}_2:\ s = -0.5 + 1 * 1.0 + 0 * 1.0 = 0.5, && \tau(0.5) = 1\\
&\mathbf{x}_3:\ s = -0.5 + 0 * 1.0 + 1 * 1.0 = 0.5, && \tau(0.5) = 1\\
&\mathbf{x}_4:\ s = -0.5 + 1 * 1.0 + 1 * 1.0 = 1.5, && \tau(1.5) = 1
\end{aligned}$$

결국 [그림 3-4(b)]의 퍼셉트론은 샘플 4개를 모두 맞추었다. 이 퍼셉트론은 훈련집합을 100% 성능으로 분류한다고 말할 수 있다.

여기서 $\tau(y)$는 계단함수

Notation

2차원 특징 벡터로 표현되는 샘플을 4개 가진 훈련집합 $\mathbb{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}\}$, $\mathbb{Y} = \{y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}\}$를 생각하자. [그림 3-4(a)]는 이 데이터를 보여준다.

$$\mathbf{x}^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\ y^{(1)} = -1,\ \ \mathbf{x}^{(2)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ y^{(2)} = 1,\ \ \mathbf{x}^{(3)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},\ y^{(3)} = 1,\ \ \mathbf{x}^{(4)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ y^{(4)} = 1$$

☆ Notation 이렇게 쓸거라는 소리

- $\underline{x}_k = \begin{bmatrix} x_{k1} \\ \vdots \\ x_{kd} \end{bmatrix} \rightarrow \underline{x}^{(k)} = \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_d^{(k)} \end{bmatrix}$
  특징벡터
- $y_k \rightarrow y^{(k)}$

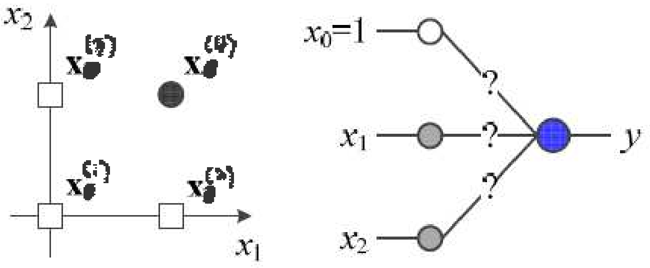
"기계학습" 교재

"Coursera ML" & 김성훈

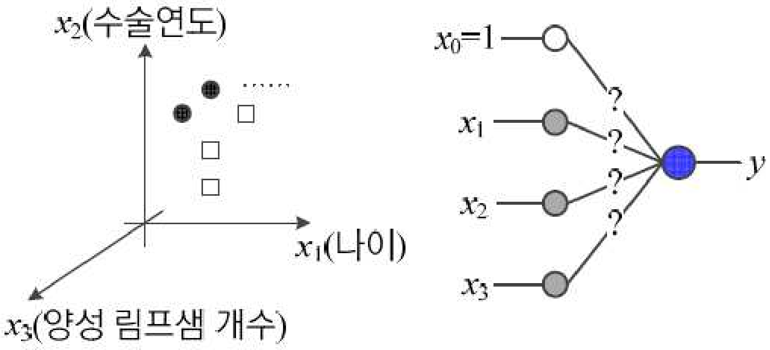
Perceptron – *Learning Problem*

◆ 학습 문제

- w_1과 w_2, w_0이 어떤 값을 가져야 100% 옳게 분류할까?
- 현실 세계는 d차원 공간에 수백~수만 개의 샘플이 존재
  - ★ 예, MNIST는 784차원에 6만개 샘플



(a) AND 분류 문제



(b) Haberman survival 분류 문제

Perceptron – *Cost Function*

- ◆ Cost Function
  - Parameters (가중치): $\mathbf{w} = (w_0, w_1, w_2, \cdots, w_d)^T$
  - Cost Function의 조건:
    - $J(\mathbf{w}) \geq 0$이다.
    - $\mathbf{w}$가 최적이면, 즉 모든 샘플을 맞히면 $J(\mathbf{w}) = 0$이다.
    - 틀리는 샘플이 많은 $\mathbf{w}$일수록 $J(\mathbf{w})$는 큰 값을 가진다.
  - Cost Function for Perceptron: 틀린 샘플에 대해서만 정리
    - ★ Y를 오분류된 샘플의 집합이라 할 때,

$$\boxed{J(\mathbf{w}) = \sum_{\mathbf{x}^{(k)} \in Y} -y^{(k)}\left(\mathbf{w}^T \mathbf{x}^{(k)}\right)}$$

GT

prediction

GT(Ground Truth): 1

$y^{(k)} = 1$: $-(\mathbf{w}^T \mathbf{x}^{(k)})$

- 오분류: $\mathbf{w}^T \mathbf{x}^{(k)} < 0 \Rightarrow -(\mathbf{w}^T \mathbf{x}^{(k)}) > 0$
- 정분류: $> 0 \Rightarrow < \;)$

틀리면 어쨌든 양수

GT = −1

$y^{(k)} = -1$: $(\mathbf{w}^T \mathbf{x}^{(k)})$

- 오분류: $\mathbf{w}^T \mathbf{x}^{(k)} > 0 \Rightarrow (\mathbf{w}^T \mathbf{x}^{(k)}) > 0$
- 정분류: $< 0 \Rightarrow < 0 \;)$

Perceptron – *Gradient*

Gradient

$$J(\mathbf{w}) = \sum_{\mathbf{x}_k \in Y} -y_k\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_k\right)$$

내적

$$\frac{\partial J(\mathbf{w})}{\partial w_i} = \sum_{\mathbf{x}_k \in Y} \frac{\partial(-y_k(w_0 x_{k0} + w_1 x_{k1} + \cdots + w_i x_{ki} + \cdots + w_d x_{kd}))}{\partial w_i} = \sum_{\mathbf{x}_k \in Y} -y_k x_{ki}$$

$$\boxed{\frac{\partial J(\mathbf{w})}{\partial w_i} = \sum_{\mathbf{x}_k \in Y} -y_k x_{ki}, \qquad i = 0,1,\cdots,d}$$

$$\boxed{w_i = w_i + \rho \sum_{\mathbf{x}_k \in Y} y_k x_{ki}, \qquad i = 0,1,\cdots,d}$$

Perceptron – *Learning Algorithms*

◆ Batch mode:

↱ 오분류 집합

- 훈련집합의 샘플을 모두 맞출(즉 $Y = \emptyset$) 때까지 세대epoch(라인 3~9)를 반복함

알고리즘 3-1 퍼셉트론 학습(배치 버전)

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ

출력: 최적 가중치 $\hat{\mathbf{w}}$

1 난수를 생성하여 초기해 $\mathbf{w}$를 설정한다.

2 repeat

3 $\quad Y = \emptyset$ // 틀린 샘플 집합

epoch

4 $\quad$ for j=1 to n // 모든 training sample에 대해

5 $\quad\quad y = \tau(\mathbf{w}^{\mathrm{T}}\mathbf{x}_j)$ // 식 (3.4) : 출력값 계산 (forward)

6 $\quad\quad$ if$(y \neq y_j)$ $Y = Y \cup \mathbf{x}_j$ // 틀린 샘플을 집합에 추가한다. → Y

오분류

7 $\quad$ if$(Y \neq \emptyset)$ // 오분류 샘플이 하나라도 있으면

8 $\quad\quad$ for i=0 to d // 식 (3.9)

9 $\quad\quad\quad w_i = w_i + \rho \sum_{\mathbf{x}_k \in Y} y_k x_{ki}$

parameter update (iteration)

10 until $(Y = \emptyset)$

11 $\hat{\mathbf{w}} = \mathbf{w}$

↳ 모든 샘플을 본 후 업데이트

행렬 표기

$$
\left.
\begin{array}{l}
8.\quad \text{for } i = 0 \text{ to } d \\
9.\quad\quad w_i = w_i + \rho \sum_{\mathbf{x}_k \in Y} y_k x_{ki}
\end{array}
\right\}
$$

$$\rightarrow \quad 8.\quad \mathbf{w} = \mathbf{w} + \rho \sum_{\mathbf{x}_k \in Y} y_k \mathbf{x}_k$$

Perceptron – *Learning Algorithms (cont'd)*

Stochastic mode:

- 샘플 순서를 섞음. 틀린 샘플이 발생하면 즉시 갱신

알고리즘 3-2 퍼셉트론 학습(스토캐스틱 버전)

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력: 최적 가중치 $\hat{\mathbf{w}}$

1 난수를 생성하여 초기해 $\mathbf{w}$을 설정한다.
2 repeat
3    $\mathbb{X}$의 샘플 순서를 섞는다. // stochastic
4   quit=true
5   for j=1 to n // 모든 training sample에 대해
6       $y = \tau(\mathbf{w}^{\mathrm{T}}\mathbf{x}_j)$ // 식 (3.4) : 출력값 계산
7   if($y \neq y_j$) // 이 샘플의 출력이 잘못됐으면
8   quit=false
9   for i=0 to d
10             $w_i = w_i + \rho y_j x_{ji}$
11 until(quit) // 틀린 샘플이 없을 때까지
12 $\hat{\mathbf{w}} = \mathbf{w}$

epoch

parameter update (iteration)

Online Mode

<u>행렬 표기</u>

9. for $i = 0$ to d
10. $\quad w_i = w_i + \rho y_j x_{ji}$

$\rightarrow$ 9. $\mathbf{w} = \mathbf{w} + \rho y_j \mathbf{x}_j$

[주의] 선형분리 불가능한 경우에는 무한 반복함
→ until($Y = \emptyset$) 또는 until(quit)를
until(더 이상 개선이 없다면)으로 수정해야 함

→Batch Mode도 동일함

Perceptron – *Learning Algorithms (cont'd)*

◆ **Stochastic Mini-batch mode:**

Mini-Batch만큼 업데이트

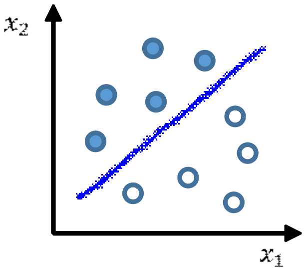
2. Decision Boundary

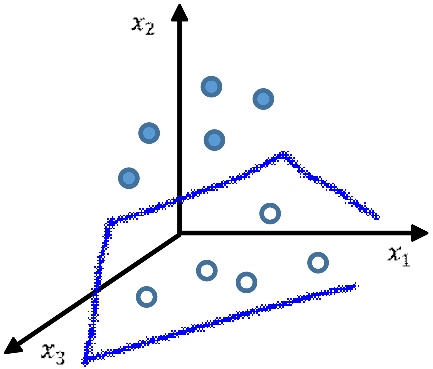
Decision Boundary

- **결정 경계 (Decision Boundary)**
  - The boundary in the ***feature space*** that separates the area of each class: $d(\mathbf{x}) = 0$
    - ★ 결정 경계는 전체 특징 공간을 두 부분공간으로 분할하는 분류기 역할

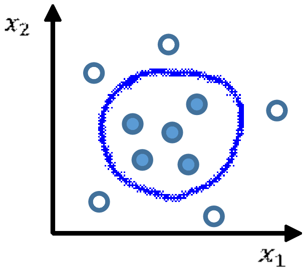
- **Examples (*for Binary Classifications*)**

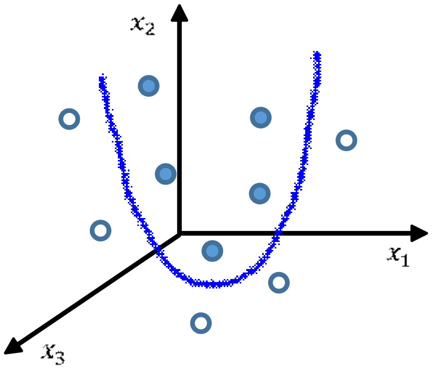
Linear Decision Boundary → 선형 분류기





Non-linear Decision Boundary → 비선형 분류기





Linear Decision Boundary

- Equation for *Linear Decision Boundary* (선형 결정 경계)

$$\boxed{d(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_dx_d + w_0 = 0}$$

★ class 1 if $d(\mathbf{x}) > 0$, class 2 if $d(\mathbf{x}) < 0$

- Two types of vector representation:

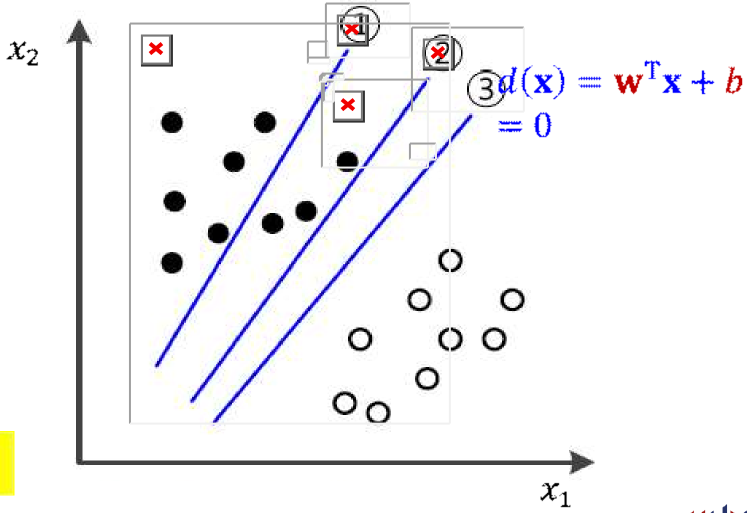
★ Let $\mathbf{x} = [x_0\ x_1\ \ldots x_d], \mathbf{w} = [w_0\ w_1\ \ldots w_d]$: $d(\mathbf{x}) = \mathbf{w}^\mathrm{T}\mathbf{x} = 0$

★ Let $\mathbf{x} = [x_1\ \ldots x_d], \mathbf{w} = [w_1\ \ldots w_d]$: $d(\mathbf{x}) = \mathbf{w}^\mathrm{T}\mathbf{x} + b = 0$

- Geometric Interpretation
  - $d(\mathbf{x}) = 0$ is a *hyperplane* in the feature space.
  -

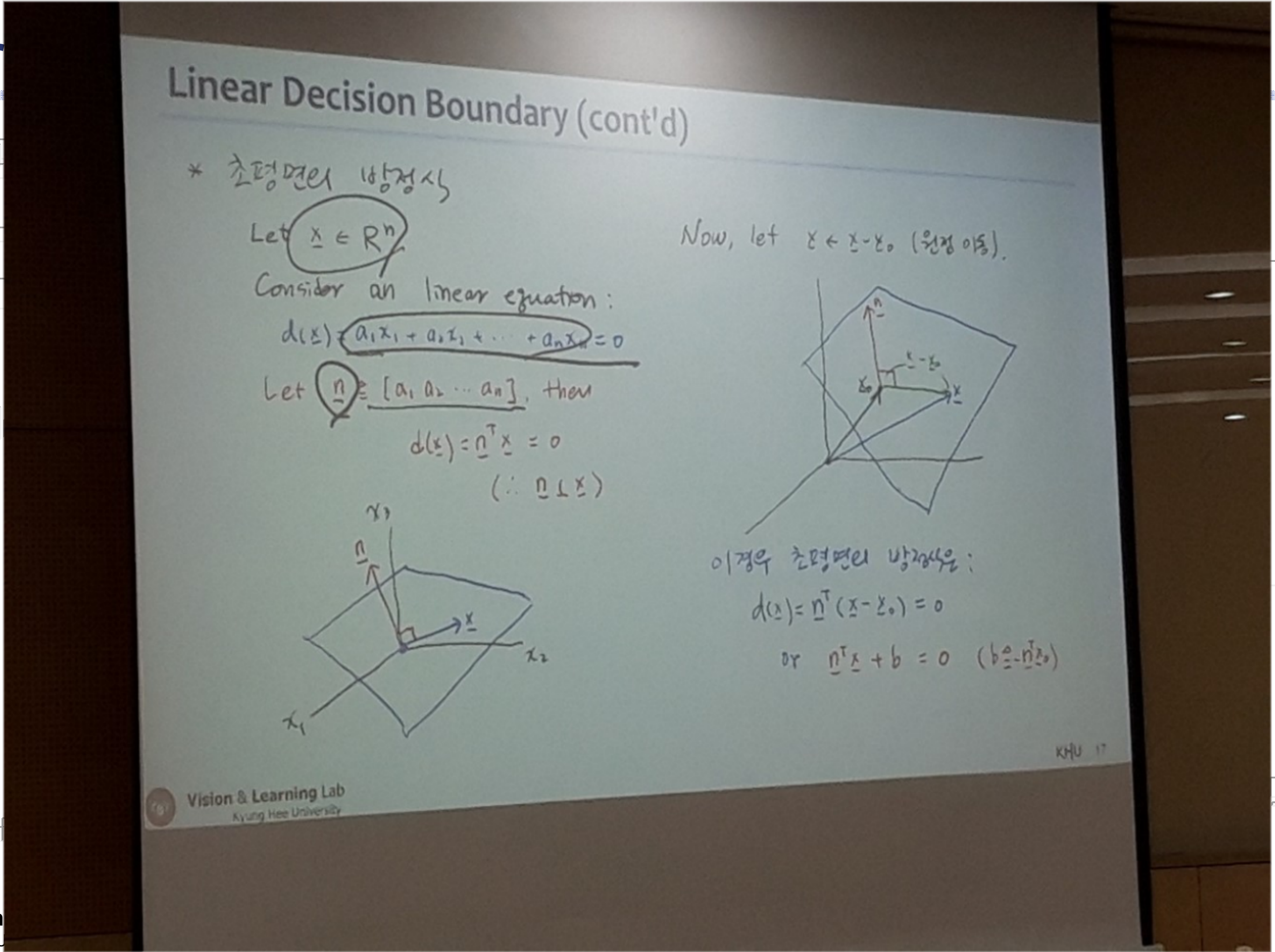
$d(\mathbf{x}) = \mathbf{w}^\mathrm{T}\mathbf{x} + b = 0$ *normal vector* of the hyperplane.

★ b determines the *position* (i.e., the displacement from the origin) of the hyperplane.



w는 결정경계의 방향을 결정하고, *b*는 위치를 결정한다.

Linear



Linear Decision Boundary (cont'd)
* 초평면의 방정식
Let $\underline{x} \in R^n$
Consider an linear equation:
$d(\underline{x}) = a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$
Let $\underline{n} = [a_1\ a_2 \cdots a_n]$, then
$d(\underline{x}) = \underline{n}^T\underline{x} = 0$
$(\because \underline{n} \perp \underline{x})$
x_3
$\underline{n}$
$\underline{x}$
x_2
x_1
Now, let $\underline{x} \leftarrow \underline{x} - \underline{x}_0$ (원점 이동).
$\underline{n}$
$\underline{x} - \underline{x}_0$
$\underline{x}_0$
$\underline{x}$
이경우 초평면의 방정식은:
$d(\underline{x}) = \underline{n}^T(\underline{x} - \underline{x}_0) = 0$
or $\underline{n}^T\underline{x} + b = 0$ (b는 $-\underline{n}^T\underline{x}_0$)
KHU 17
Vision & Learning Lab
Kyung Hee University

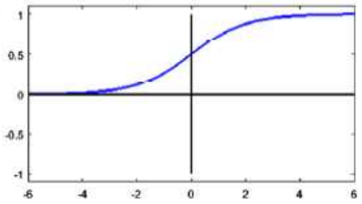
Linear Decision Boundary (cont'd)

- For *Logistic Regression*

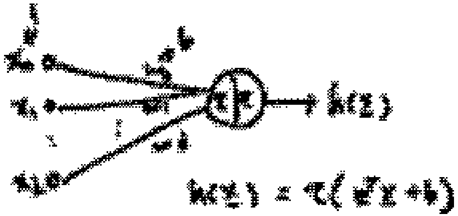
$$h(\mathbf{x}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b}\right) \lessgtr Th = 0.5$$

$$\Longrightarrow \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} \lessgtr 0$$

$$\Longrightarrow d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



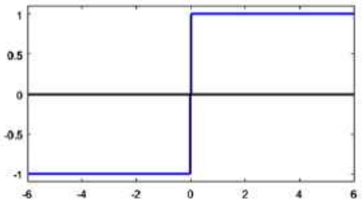
- For *Perceptron*



$$h(\mathbf{x}) = \tau\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b}\right) \lessgtr Th = 0$$

$$\Longrightarrow \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b} \lessgtr 0$$

$$\Longrightarrow d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



Linear Decision Boundary (cont'd)

- Example: Binary Classification using Linear Regression

$$\left.\begin{array}{l} h_\theta(x) = \theta^T x \ge 0 \Rightarrow y = 1 \\ \theta^T x < 0 \Rightarrow y = 0 \end{array}\right\}$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \theta^T x$$

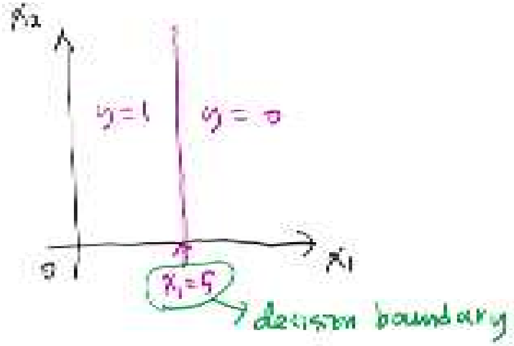
$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\theta^* = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix} \rightarrow \theta^T x = 5 - 1 \cdot x_1 + 0 \cdot x_2 = 5 - x_1$$

$$\Rightarrow \begin{cases} 5 - x_1 \ge 0 \Rightarrow y = 1 \quad (x_1 \le 5) \\ 5 - x_1 < 0 \Rightarrow y = 0 \quad (x_1 > 5) \end{cases}$$

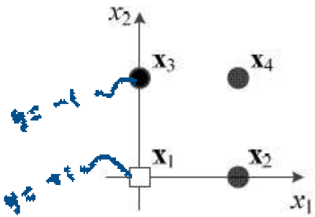
$h_\theta(x) = \theta^T x = 0$

← decision boundary

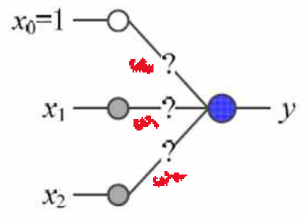


Perceptron – *Quiz*

◆ 아래 그림의 OR Gate를 Perceptron으로 구현하시오.



(a) 훈련집합



(b) 퍼셉트론

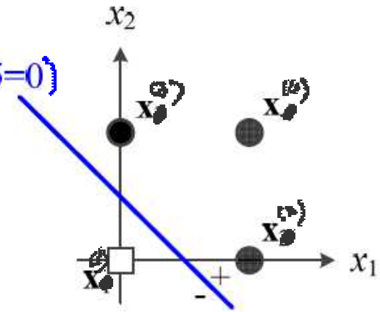
$$w_0 + w_1 x_1 + w_2 x_2 \gtrless 0 \quad \begin{cases} y=1 \\ y=-1 \end{cases}$$

$$d(\mathbf{x}) = d(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_0 = 0$$

● 결정 직선 구하기:

$d(\underline{x}) = 0$

$(x_1 + x_2 - 0.5 = 0)$



$d(\underline{x}) = 0 \Rightarrow x_2 = -x_1 + 0.5$

$$\begin{cases} \bullet\ (y=1) : d(\underline{x}) > 0 \\ \square\ (y=-1) : d(\underline{x}) < 0 \end{cases}$$

$\Rightarrow [w_0\ w_1\ w_2] = [-0.5,\ 1,\ 1]$