인공지능

- 신경 회로망 상세 설명 및 실험 -

부산대학교

LG Electronics

목차

인공지능 개요 (신경 회로망 상세 설명)

I. 신경 회로망 상세 설명

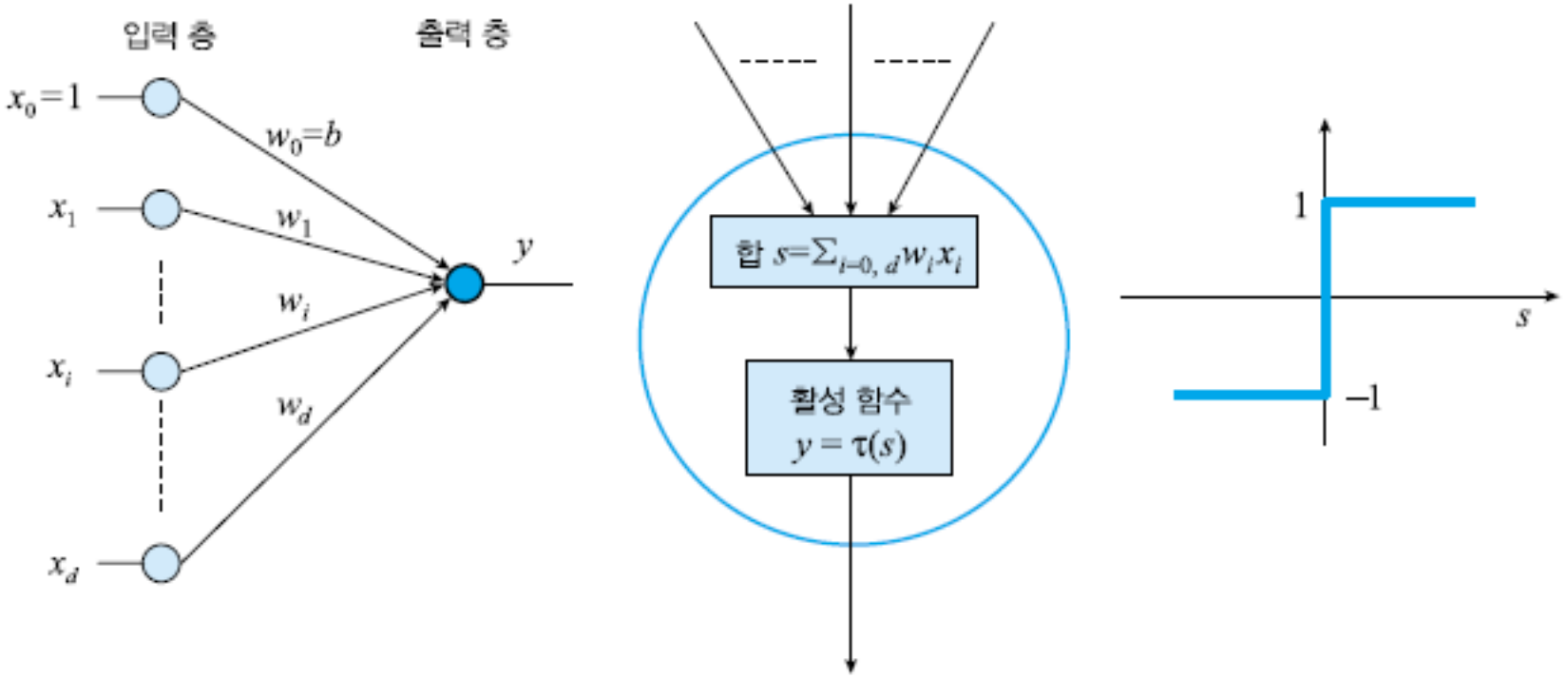
퍼셉트론

- 새로운 개념들 등장
  - 층
  - 노드와 가중치
  - 학습
  - 활성 함수
- 비록 분명한 한계를 가지지만 MLP의 초석이 됨

구조와 원리

구조

- 입력층: d+1개의 노드 (특징 벡터 $\mathbf{x}=(x_1,\cdots,x_d)^T$)
- 출력층: 한 개의 노드 (따라서 2-부류 분류기)
- 에지와 가중치



(a) 전체 구조 (b) 출력 노드의 연산 (c) 활성 함수

< 퍼셉트론의 구조 >

구조와 원리

노드의 연산

- 입력 노드: 받은 신호를 단순히 전달
- 출력 노드: 합 계산과 활성 함수 계산

$$\left.\begin{aligned} y = \tau(s) = \tau\left(\sum_{i=1}^{d} w_i x_i + b\right) = \tau(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \\ \text{이때 } \tau(s) = \begin{cases} +1, s \geq 0 \\ -1, s < 0 \end{cases} \end{aligned}\right\}$$

퍼셉트론은 선형 분류기

$$\left.\begin{aligned} d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b > 0 \text{이면} \quad \mathbf{x} \in \omega_1 \\ d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0 \text{이면} \quad \mathbf{x} \in \omega_2 \end{aligned}\right\}$$

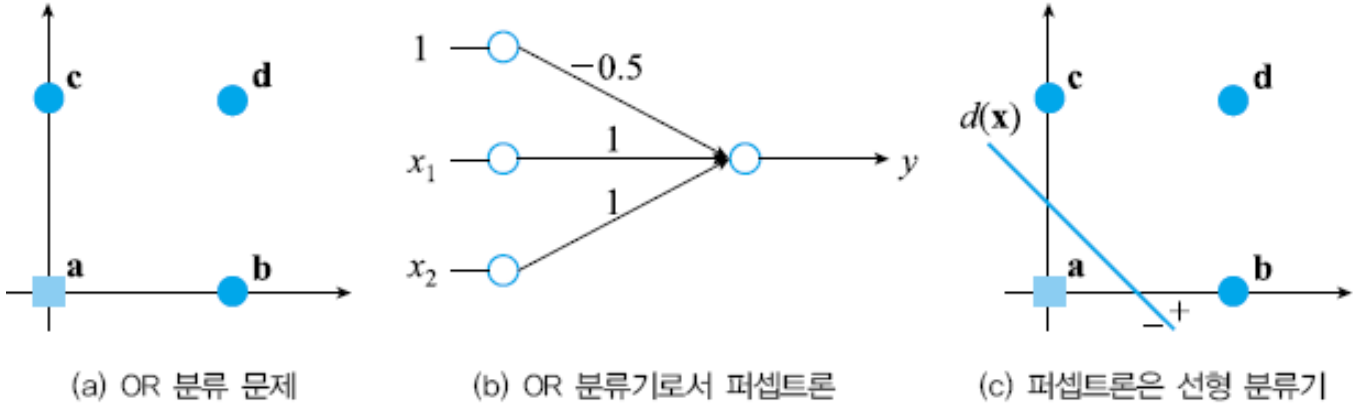
예) OR

$\mathbf{a} = (0,0)^T,\ t_a = -1$

$\mathbf{b} = (1,0)^T,\ t_b = 1$

$\mathbf{c} = (0,1)^T,\ t_c = 1$

$\mathbf{d} = (1,1)^T,\ t_d = 1$



(a) OR 분류 문제 (b) OR 분류기로서 퍼셉트론 (c) 퍼셉트론은 선형 분류기

< 퍼셉트론의 예 >

이 퍼셉트론은 $\mathbf{w}=(1,1)^T$, b= - 0.5

따라서 결정 직선은 $d(\mathbf{x}) = x_1 + x_2 - 0.5$

이 경우, point $c=(0,1)^T$를 제대로 분류함

$$y = \tau(\mathbf{w}^T\mathbf{c} + b) = \tau((1,1)\begin{pmatrix}0\\1\end{pmatrix} - 0.5) = \tau(0.5) = 1$$

$$y = \tau(s) = \tau(\sum_{i=1}^{d} w_i x_i + b) = \tau(\mathbf{w}^T\mathbf{x} + b)$$

$$\text{이때 } \tau(s) = \begin{cases} +1, s \geq 0 \\ -1, s < 0 \end{cases}$$

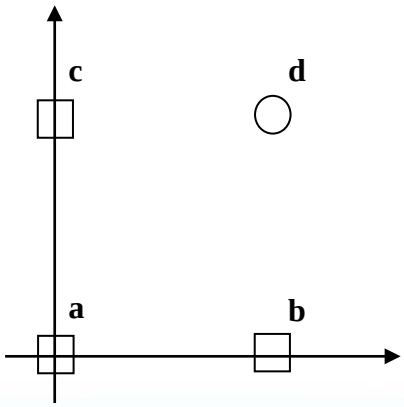
퍼셉트론 학습이란?

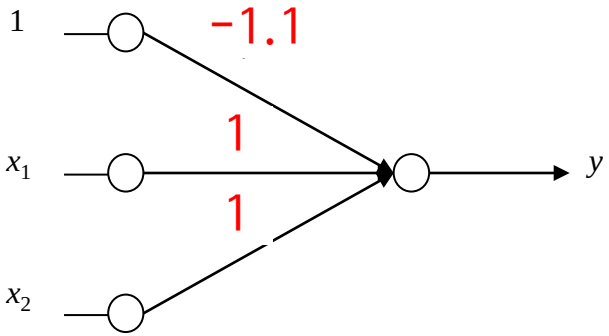
퍼셉트론 학습이란? 훈련 집합 $X = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$이 주어졌을 때 이들을 모두 옳게 분류하는 퍼셉트론 (즉 $\mathbf{w}$와 b)을 찾아라. 샘플 $(\mathbf{x}_i, t_i)$에서 $\mathbf{x}_i$는 특징 벡터이고 t_i는 부류 표지로서 $\mathbf{x}_i \in \omega_1$이면 $t_i = 1$이고 $\mathbf{x}_i \in \omega_2$이면 $t_i = -1$이다. X는 선형 분리 가능하다고 가정한다.[3]

예) AND 분류 문제

$\mathbf{a}=(0,0)^T$ $\mathbf{b}=(1,0)^T$ $\mathbf{c}=(0,1)^T$ $\mathbf{d}=(1,1)^T$

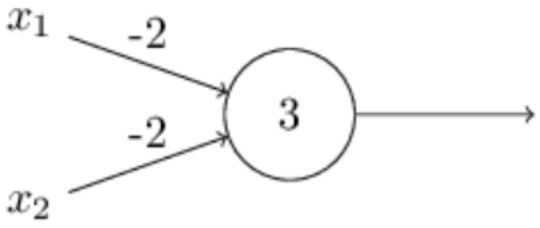
$t_a = -1$ $t_b = -1$ $t_c = -1$ $t_d = 1$





예) NAND

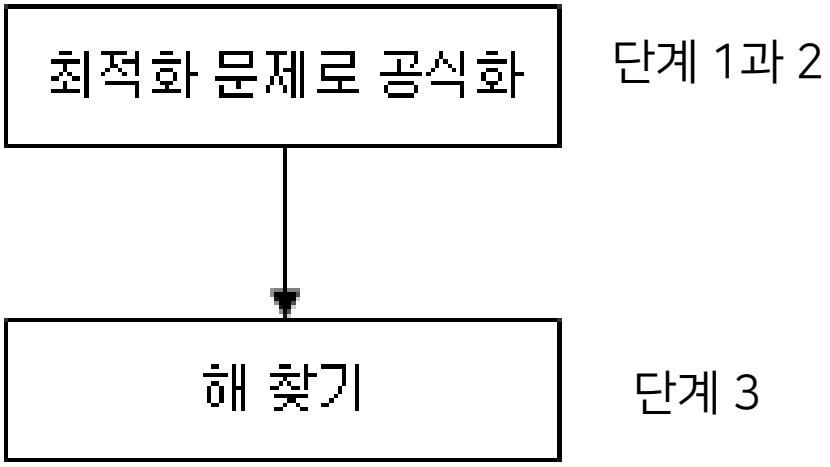
x1	x2	t
0	0	1
0	1	1
1	0	1
1	1	-1



학습과 인식

패턴 인식에서 일반적인 학습 알고리즘 설계 과정

- 단계 1: 분류기 구조 정의와 분류 과정의 수학식 정의
- 단계 2: 분류기 품질 측정용 비용 함수 $J(\Theta)$ 정의
- 단계 3: $J(\Theta)$를 최적화하는 parameter Θ를 찾는 알고리즘 설계



학습 알고리즘의 구조패턴 모드와 배치 모드

- **배치 모드: 오분류된 모든 샘플을 모은 다음, 이를 사용하여 한꺼번에 가중치를 갱신함**
- **패턴 모드: 샘플을 하나 입력한 후, 잘못 인식하면 곧 바로 가중치를 갱신함**

부류 표지로서 $\mathbf{x}_i \in \omega_1$이면 $t_i = 1$이고 $\mathbf{x}_i \in \omega_2$이면 $t_i =$ -1

단계 1: 분류기 구조 정의와 분류 과정의 수학식 정의

- 식 (4.2)
- 매개변수 집합 $\Theta = \{\mathbf{w}, b\}$

$$y = \tau(s) = \tau\left(\sum_{i=1}^{d} w_i x_i + b\right) = \tau(\mathbf{w}^T\mathbf{x} + b)$$

$$\text{이때 } \tau(s) = \begin{cases} +1, & s \geq 0 \\ -1, & s < 0 \end{cases} \qquad (4.2)$$

단계 2: 분류기 품질 측정용 비용함수(cost function) $J(\Theta)$ 정의

- 분류기 품질을 측정하는 $J(\Theta)$를 어떻게 정의할 것인가?

$$J(\Theta) = \sum_{\mathbf{x}_k \in Y} (-t_k)(\mathbf{w}^T\mathbf{x}_k + b) \qquad (4.4)$$

- Y: 오분류된 샘플 집합
- $J(\Theta)$는 항상 양수
- Y가 공집합이면 $J(\Theta)=0$
- $|Y|$가 클수록 $J(\Theta)$ 큼

▪오분류된 x_k가 ω_1에 속한다면, t_k=1이며, 오분류 되었기때문에 $\mathbf{w}^T\mathbf{x}_k+b$<0이 되어, 최종 비용함수 $J(\Theta)$ 값은 양수가됨

▪또한, 오분류된 x_k가 ω_2에 속하더라도, t_k=−1이며, 오분류 되었기때문에 $d\mathbf{w}^T\mathbf{x}_k+b$ 값이 음수가 아닌 양수가되어, 결국 비용함수 $J(\Theta)$ 값은 양수값이 됨.

▪즉, 오분류되면 비용함수는 증가함

학습과 인식

- 단계 3: $J(\Theta)$를 최소화하는 Θ를 찾는 알고리즘 설계
  - $J(\Theta)=0$인 Θ를 찾아라.
  - 내리막 경사법 (Gradient descent method)
    - 현재 해를 $-\partial/\partial\Theta$ 방향으로 이동
    - 학습률 ρ (learning rate)를 곱하여 어떤 특정 값 크기만큼 이동

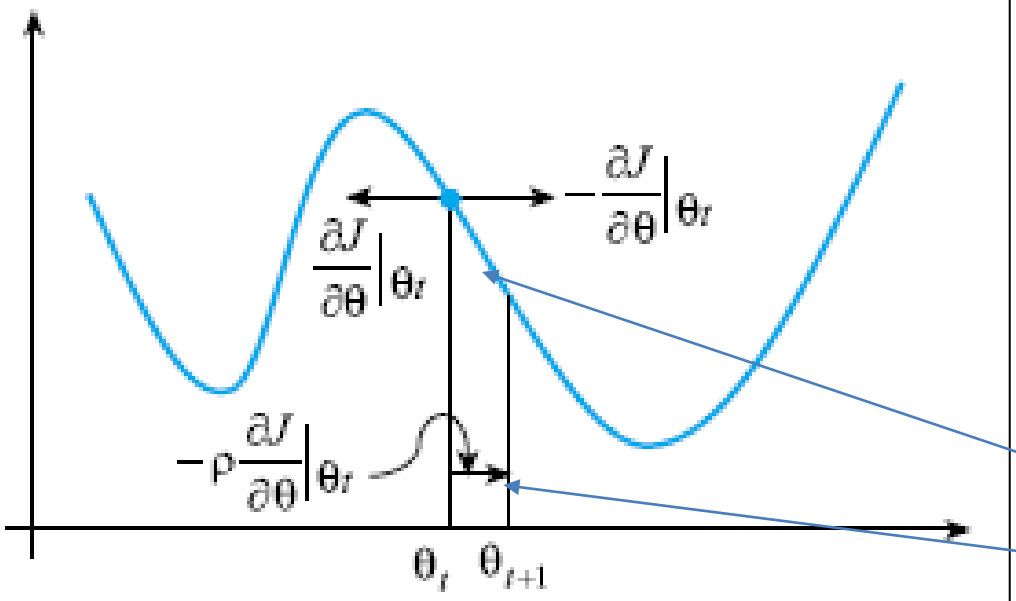


그림 11.9 내리막 경사법

$\frac{\partial J}{\partial \theta}$는 음의 값을 가짐

θ값이 더큰 곳에 최적점이 존재하므로

$\Delta\theta$를 양의 값으로 하기위해선 $-\frac{\partial J}{\partial \theta}$로 해야함

학습과 인식

$$J(\Theta) = \sum_{\mathbf{x}_k \in Y} (-t_k)(\mathbf{w}^T \mathbf{x}_k + b) \quad (4.4)$$

매개변수 집합 Θ={**w**, *b*}

- **알고리즘 스케치**
  - 초기해를 설정한다.
  - 멈춤 조건이 만족될 때까지 현재 해를 $-\partial/\partial\Theta$ 방향으로 조금씩 이동시킨다.
- **알고리즘 수식**

$$\Theta(h+1) = \Theta(h) - \rho(h)\frac{\partial J(\Theta)}{\partial \Theta} \quad (4.5)$$

$$\left.\begin{aligned} \frac{\partial J(\Theta)}{\partial \mathbf{w}} &= \sum_{\mathbf{x}_k \in Y} (-t_k)\mathbf{x}_k \\ \frac{\partial J(\Theta)}{\partial b} &= \sum_{\mathbf{x}_k \in Y} (-t_k) \end{aligned}\right\} \quad (4.6)$$

$$\left.\begin{aligned} \mathbf{w}(h+1) &= \mathbf{w}(h) + \rho(h)\sum_{\mathbf{x}_k \in Y} t_k \mathbf{x}_k \\ b(h+1) &= b(h) + \rho(h)\sum_{\mathbf{x}_k \in Y} t_k \end{aligned}\right\} \quad (4.7)$$

또는

$$\hat{\mathbf{w}}(h+1) = \hat{\mathbf{w}}(h) + \rho(h)\sum_{\mathbf{x}_k \in Y} t_k \hat{\mathbf{x}}_k$$

←퍼셉트론 학습 규칙
(델타 규칙)

학습과 인식

- 배치 모드: 오분류된 모든 샘플을 모은 다음, 이를 사용하여 한꺼번에 가중치를 갱신함
- 패턴 모드: 샘플을 하나 입력한 후, 잘못 인식하면 곧 바로 가중치를 갱신함

알고리즘 [4.1] 퍼셉트론 학습 (배치 모드 batch mode)

입력: 훈련 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 학습률 ρ

출력: 퍼셉트론 가중치 $\mathbf{w}$, b

알고리즘:

1. $\mathbf{w}$와 b를 초기화한다.
2. **repeat** {
3. $Y = \emptyset$;
4. **for** ($i = 1$ **to** N) {
5. $y = \tau(\mathbf{w}^T\mathbf{x}_i + b)$; // (4.2)로 분류를 수행함
6. **if** ($y \neq t_i$) $Y = Y \cup \mathbf{x}_i$; // 오분류된 샘플 수집
7. }
8. $\mathbf{w} = \mathbf{w} + \rho \sum_{\mathbf{x}_k \in Y} t_k \mathbf{x}_k$; // (4.7)로 가중치 갱신
9. $b = b + \rho \sum_{\mathbf{x}_k \in Y} t_k$;
10. } **until** ($Y = \emptyset$);
11. $\mathbf{w}$와 b를 저장한다.

오분류 샘플을 모은 후에, 나중에 한꺼번에 가중치를 갱신함

알고리즘 [4.2] 퍼셉트론 학습 (패턴 모드)

입력: 훈련 집합 $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 학습률 ρ

출력: 퍼셉트론 가중치 $\mathbf{w}$, b

알고리즘:

1. $\mathbf{w}$와 b를 초기화한다.
2. **repeat** {
3. QUIT = true;
4. **for** ($i = 1$ **to** N) {
5. $y = \tau(\mathbf{w}^T\mathbf{x}_i + b)$; // (4.2)로 분류를 수행함
6. **if** ($y \neq t_i$) { QUIT = false; $\mathbf{w} = \mathbf{w} + \rho t_i \mathbf{x}_i$; $b = b + \rho t_i$;}
7. }
8. } **until** (QUIT);
9. $\mathbf{w}$와 b를 저장한다.

오분류 상황, 해당 w,b를 바로 update함

Batch mode 사용 퍼셉트론 학습 예제

$\mathbf{w}(0)=(-0.5,0.75)^T$, $b(0)=0.375$ ⟶ $d(\mathbf{x})= -0.5x_1+0.75x_2+0.375$

$$\mathbf{w} = \mathbf{w} + \rho \sum_{\mathbf{x}_k \in Y} t_k \mathbf{x}_k \; ;$$

$$b = b + \rho \sum_{\mathbf{x}_k \in Y} t_k \; ;$$

points	target class	d값	분류class(오분류)
a(0,0)	ta = -1	0.375	1 (오분류)
b(1,0)	tb = 1	-0.125	-1 (오분류)
c(0,1)	tc = 1	1.125	1
d(1,1)	td = 1	0.625	1

① 아래의 결정 직선 사용시, 분류한 결과 **a, b**가 오분류됨…

$d(\mathbf{x})= -0.5x_1+0.75x_2+0.375$

$Y=\{\mathbf{a}, \mathbf{b}\}$가 오분류됨

학습율

$$\mathbf{w}(1) = \mathbf{w}(0) + 0.4(t_a \cdot \mathbf{a} + t_b \cdot \mathbf{b}) = \begin{pmatrix} -0.5 \\ 0.75 \end{pmatrix} + 0.4\left[-\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} -0.1 \\ 0.75 \end{pmatrix}$$

$$b(1) = b(0) + 0.4(t_a + t_b) = 0.375 + 0.4 * 0 = 0.375$$

⟶ $d(\mathbf{x})= -0.1x_1+0.75x_2+0.375$ ② 식을 만듦

② $d(\mathbf{x})= -0.1x_1+0.75x_2+0.375$ ② 식으로 다시 분류함…

$Y=\{\mathbf{a}\}$가 오분류됨

$$\mathbf{w}(2) = \mathbf{w}(1) + 0.4(t_a \mathbf{a}) = \begin{pmatrix} -0.1 \\ 0.75 \end{pmatrix} + 0.4\left[-\begin{pmatrix} 0 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} -0.1 \\ 0.75 \end{pmatrix}$$

$$b(2) = b(1) + 0.4(t_a) = 0.375 - 0.4 = -0.025$$

⟶ $d(\mathbf{x})= -0.1x_1+0.75x_2 - 0.025$

③ $d(\mathbf{x})= -0.1x_1+0.75x_2 - 0.025$

$Y=\{\mathbf{b}\}$가 오분류됨

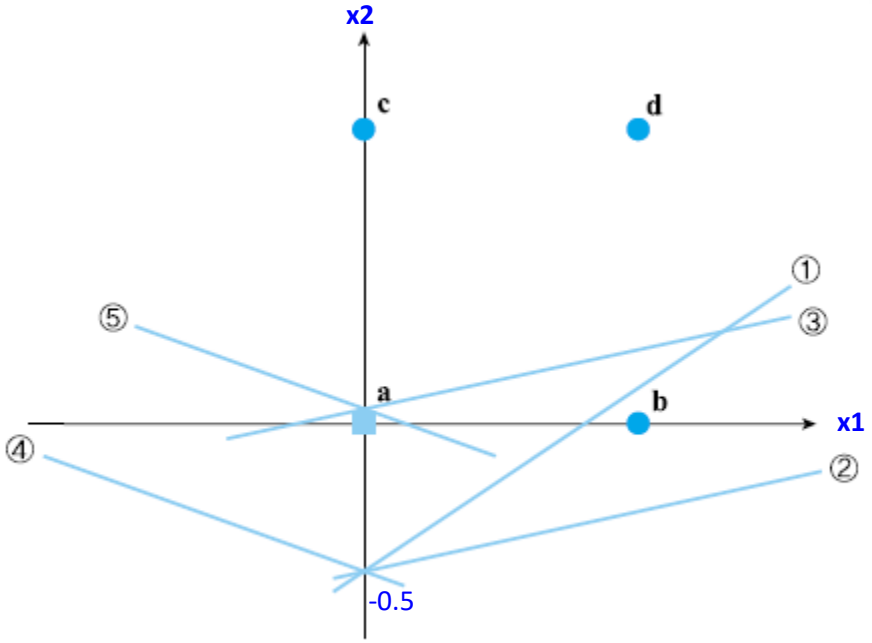


그림 4.4 예제 4.2의 퍼셉트론 학습 과정의 시각화

학습과 인식

예제 (계속)

④ $d(x) = 0.3x_1 + 0.75x_2 + 0.375$를 사용시, 오분류 points는 $Y=\{a\}$가 오분류됨

$$\mathbf{w}(4) = \mathbf{w}(3) + 0.4(t_a \mathbf{a}) = \begin{pmatrix} 0.3 \\ 0.75 \end{pmatrix} + 0.4\left[-\begin{pmatrix} 0 \\ 0 \end{pmatrix}\right] = \begin{pmatrix} 0.3 \\ 0.75 \end{pmatrix}$$

$$b(4) = b(3) + 0.4(t_a) = 0.375 - 0.4 = -0.025$$

$$d(\mathbf{x}) = 0.3x_1 + 0.75x_2 - 0.025$$

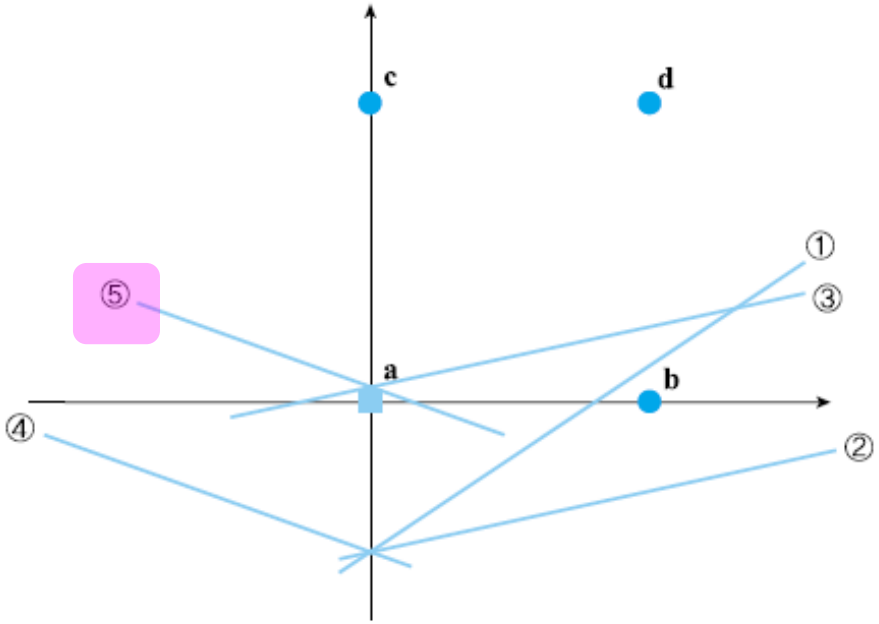


그림 4.4 예제 4.2의 퍼셉트론 학습 과정의 시각화

⑤ $d(x) = 0.3x_1 + 0.75x_2 - 0.025$는 모든 point 입력을 옳게 분류함

네 번째 세대의 결정 직선은 ⑤에 해당하는데 이 결정 직선은 모든 샘플을 옳게 분류하여 $Y = \varnothing$이 된다. 따라서 라인 2~10의 repeat 루프를 빠져 나와 $\mathbf{w} = (0.3, 0.75)^T$와 $b = -0.025$를 저장하고 마친다.

학습과 인식

인식 알고리즘

- 학습이 끝났으며, 이로써, **w**와 *b* 값이 결정됨.
- 미지의 샘플 **x**가 입력되면, 학습된 퍼셉트론으로 x를 인식할 수 있음.
- 인식에 사용되는 알고리즘은 아래와 같음. (퍼셉트론 인식 알고리즘은 매우 단순)

$$\left.\begin{array}{l} \mathbf{w}^T\mathbf{x}+b>0 \text{이면}, \mathbf{x}\in\omega_1 \\ \mathbf{w}^T\mathbf{x}+b<0 \text{이면}, \mathbf{x}\in\omega_2 \end{array}\right\} \quad (4.8)$$

학습과 인식

- **구현**
  - 그런데, 초기값 어떻게 정하는가 ?
    - **w와 b 값의 초기값은 일반적으로 작은 난수값으로 설정함**
  - 학습률은 어떻게 정하는가 ?
    - **고정된 학습율 사용**
    - **세대 수에 따라 적응적 학습율 사용**

$$\rho(h) = \rho_s - (\rho_s - \rho_e) * h / H$$

ρ_s : 시작 학습율

ρ_e : 종료시 학습율

h : 세대 수

H : 최대 세대수

Review: 패턴 모드 학습

- 샘플을 하나 입력한 후, 잘못 인식하면 곧 바로 가중치를 갱신함

알고리즘 [4.2] 퍼셉트론 학습 (패턴 모드)

입력: 훈련 집합 $X = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 학습률 ρ

출력: 퍼셉트론 가중치 $\mathbf{w}$, b

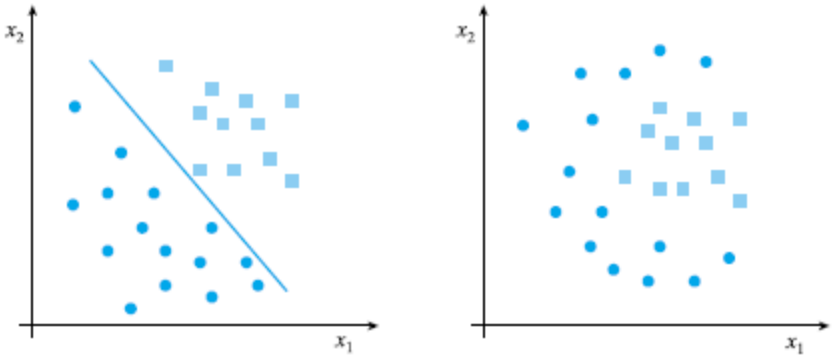
알고리즘:

1. $\mathbf{w}$와 b를 초기화한다.
2. **repeat** {
3. QUIT = true;
4. **for** ($i = 1$ **to** N) {
5. $y = \tau(\mathbf{w}^T\mathbf{x}_i + b)$; // (4.2)로 분류를 수행함
6. **if** ($y \neq t_i$) { QUIT = false; $\mathbf{w} = \mathbf{w} + \rho t_i \mathbf{x}_i$; $b = b + \rho t_i$;}
7. }
8. } **until** (QUIT);
9. $\mathbf{w}$와 b를 저장한다.

학습과 인식

만약 선형 분리 불가능할 경우에는 ?

- (b)는 선형 분리 불가능한 상황임
- 이 경우, 모든 샘플을 올바르게 분류하고자하는 목표(즉, $J(\Theta)$=0으로 만들자는 목표)를 버리고, $J(\Theta)$를 최소화하는 목표로 수정
- 새롭게 계산한 w 값이 이전 것보다 좋은지를 확인함 → 만약 더 좋으면 이를 선택 사용함 (사례: pocket algorithm)



(a) 선형 분리 가능 (b) 선형 분리 불가능

그림 4.5 선형 분리 가능과 불가능

알고리즘 [4.3] 포켓 알고리즘 (패턴 모드)

입력: 훈련 집합 집합 $X = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, 학습률 ρ

출력: 퍼셉트론 가중치 $\mathbf{w}$, b

알고리즘:

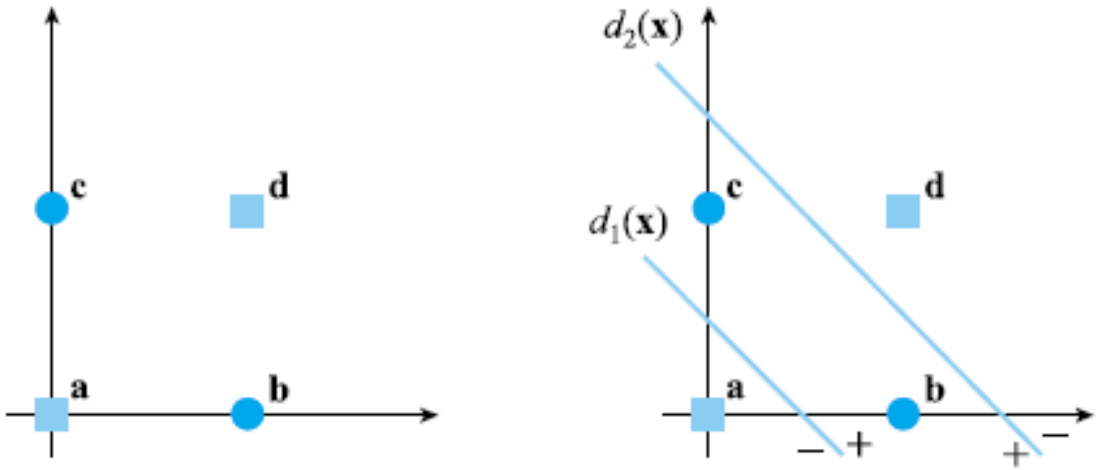
1. $\mathbf{w}$와 b를 초기화하고, 이들을 $\mathbf{w}_{best}$와 b_{best}에 저장한다.
2. $q_{best} = 0$; // 품질을 0으로 초기화
3. $h = 0$; // 세대 수
4. **repeat** {
5. **for** ($i = 1$ **to** N) {
6. $y = \tau(\mathbf{w}^T\mathbf{x}_i + b)$; // 식 (4.2)
7. **if** ($y \neq t_i$) { $\mathbf{w} = \mathbf{w} + \rho t_i \mathbf{x}_i$; $b = b + \rho t_i$;} // $\mathbf{w}$와 b 갱신
8. }
9. $\mathbf{w}$와 b로 N 개의 샘플을 인식하여 정인식률 q를 구한다.
10. **if** ($q > q_{best}$) {$\mathbf{w}_{best} = \mathbf{w}$; $b_{best} = b$; $q_{best} = q$;} // 더 좋은 가중치 발견함
11. $h = h + 1$;
12. } **until** (stop-condition);
13. $\mathbf{w} = \mathbf{w}_{best}$; $b = b_{best}$;
14. $\mathbf{w}$와 b를 저장한다.

옳게 인식하는 비율 값(q)이 높아지는 weight와 bias 값을 선택함

다층퍼셉트론 - 구조와 원리

단일 퍼셉트론의 한계: XOR 문제

- XOR인 경우, 단일 퍼셉트론으로는 75% 정인식률 한계가 있음
- 이 한계를 어떻게 극복할 것인가?
  - 두 개의 퍼셉트론 (결정 직선)을 사용하여 분류함



(a) XOR 분류 문제 (b) 두 개의 직선으로 해결

그림 4.6 XOR 분류 문제의 해결

$$\left.\begin{array}{l}\mathbf{w}_1^T\mathbf{x}+b_1>0 \text{ 이고 } \mathbf{w}_2^T\mathbf{x}+b_2>0 \text{ 이면, } \mathbf{x}\in\omega_1\\ \mathbf{w}_1^T\mathbf{x}+b_1<0 \text{ 이거나 } \mathbf{w}_2^T\mathbf{x}+b_2<0 \text{ 이면, } \mathbf{x}\in\omega_2\end{array}\right\} \quad (4.9)$$

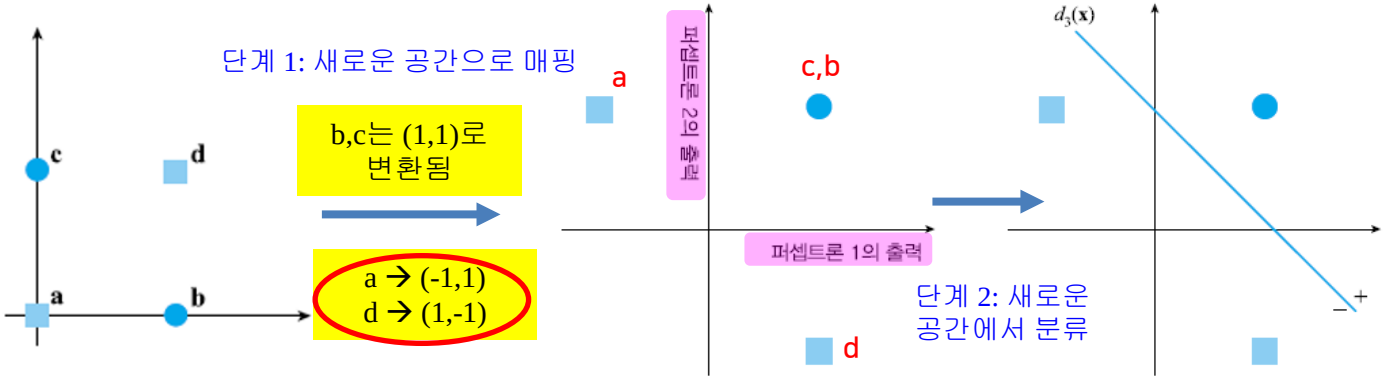
다층퍼셉트론 - 구조와 원리

두 단계에 걸쳐 문제 해결

- 단계 1: 원래 특징 공간을 새로운 공간으로 매핑 (perceptron 1, 2)
- 단계 2: 새로운 공간에서 분류 (perceptron 3에 의한 결정 직선)

표 4.1 두 단계로 XOR 문제 해결

샘플	특징 벡터 (x)		첫 번째 단계		두 번째 단계
	x_1	x_2	퍼셉트론1	퍼셉트론2	퍼셉트론3
a	0	0	−1	+1	−1
b	1	0	+1	+1	+1
c	0	1	+1	+1	+1
d	1	1	+1	−1	−1



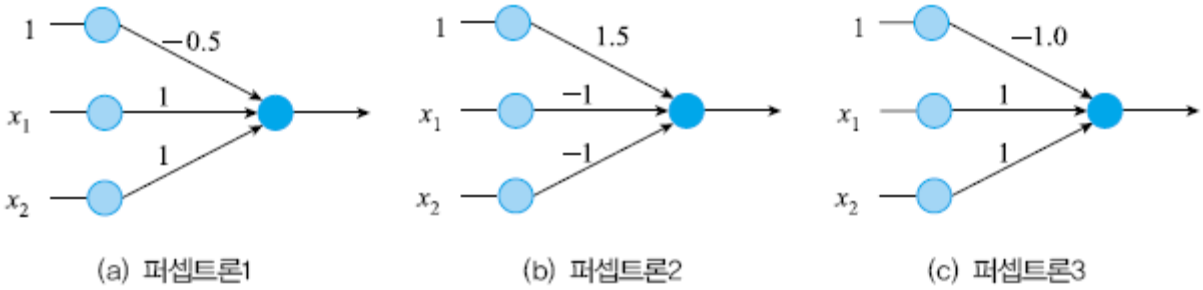
(a) XOR 분류 문제

(a) 매핑 공간에서의 샘플 분포 (b) 퍼셉트론3에 의한 영역 분할

그림 4.7 새로운 공간에서의 샘플 분포와 영역 분할

다층퍼셉트론 - 구조와 원리

다층 퍼셉트론 (MLP: Multi-layer perceptron)



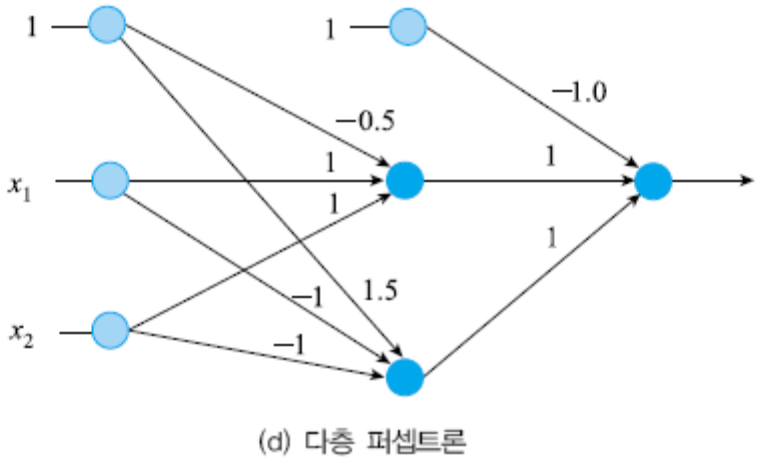


그림 4.8 세 개의 퍼셉트론과 이들을 연결하여 만든 다층 퍼셉트론

두개의 퍼셉트론1,2를 사용하여,
특징 벡터를 새로운 공간으로 매핑함

이후, 새로운 공간에서
하나의 퍼셉트론 (페셉트론 3)을 사용하여,
최종 분류함

다층퍼셉트론 - 구조와 원리

다층 퍼셉트론 아키텍처

- 입력층, 은닉층, 출력층을 가짐
- 입력을 위한 d개의 노드, 1개의 bias를 위한 노드 (총 d+1)개의 노드
- 한 개의 은닉층 및 P+1개 은닉층 노드 수(+1은 bias 값)
- 가중치: **u**와 **v**

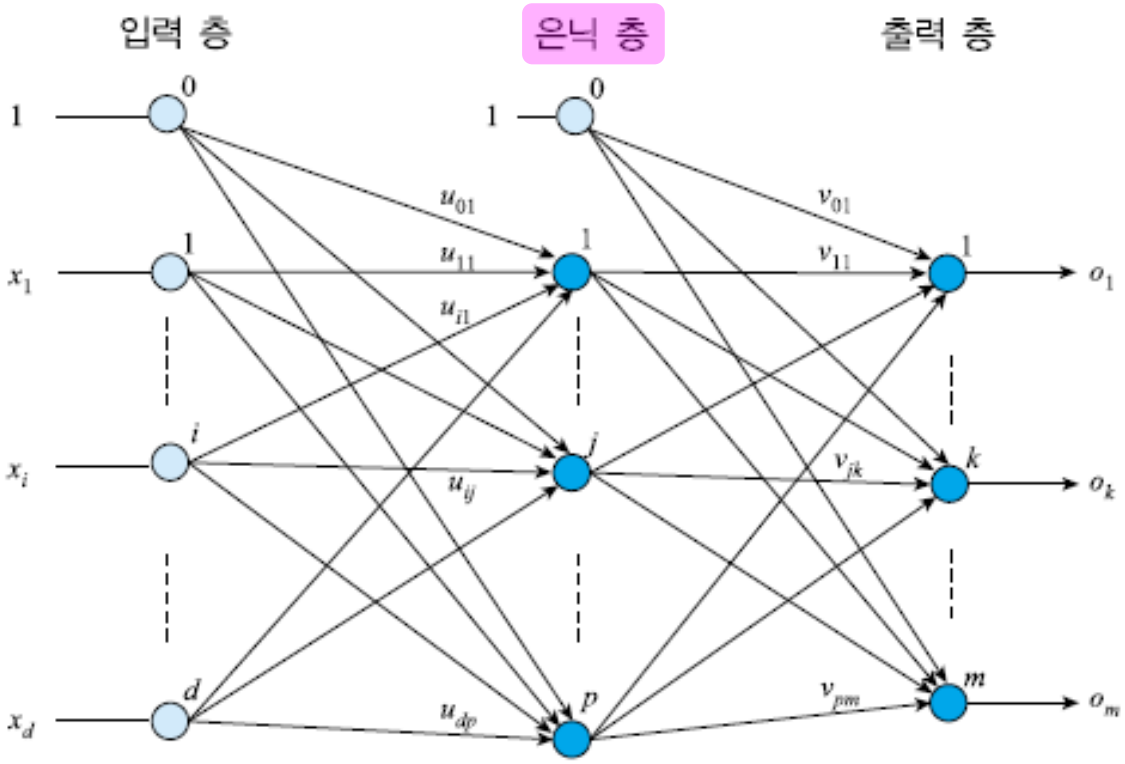


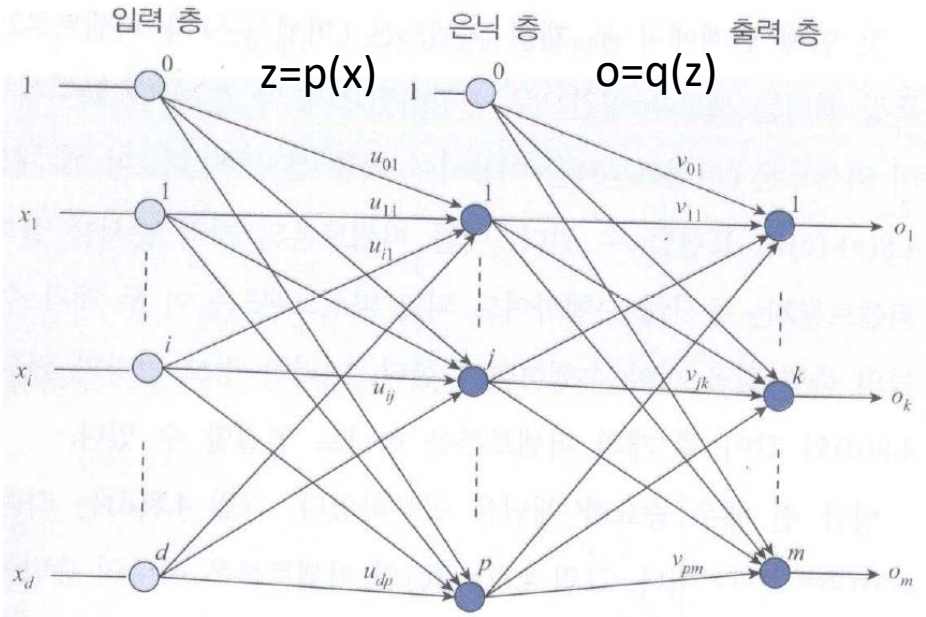
그림 4.9 다층 퍼셉트론의 구조와 표기

다층퍼셉트론 - 구조와 원리

신경망은 일종의 함수

$$\mathbf{o} = f(\mathbf{x}) \qquad (4.10)$$

$$\left.\begin{array}{l} \mathbf{z} = p(\mathbf{x}) \\ \mathbf{o} = q(\mathbf{z}) \\ \\ \text{또는} \\ \\ \mathbf{o} = q(p(\mathbf{x})) \end{array}\right\} \qquad (4.11)$$



입력 x　　은닉층 출력z　　출력벡터 o

입력층에서 은닉층 매핑 함수(p(.))

은닉층에서 출력층 매핑 함수(q(.))

다층퍼셉트론 - 구조와 원리

전방 계산 (forward computation)

- 신경회로망의 왼쪽에서 오른쪽으로 계산이 이뤄짐

은닉 층의 j번째 노드, $1 \le j \le p$:

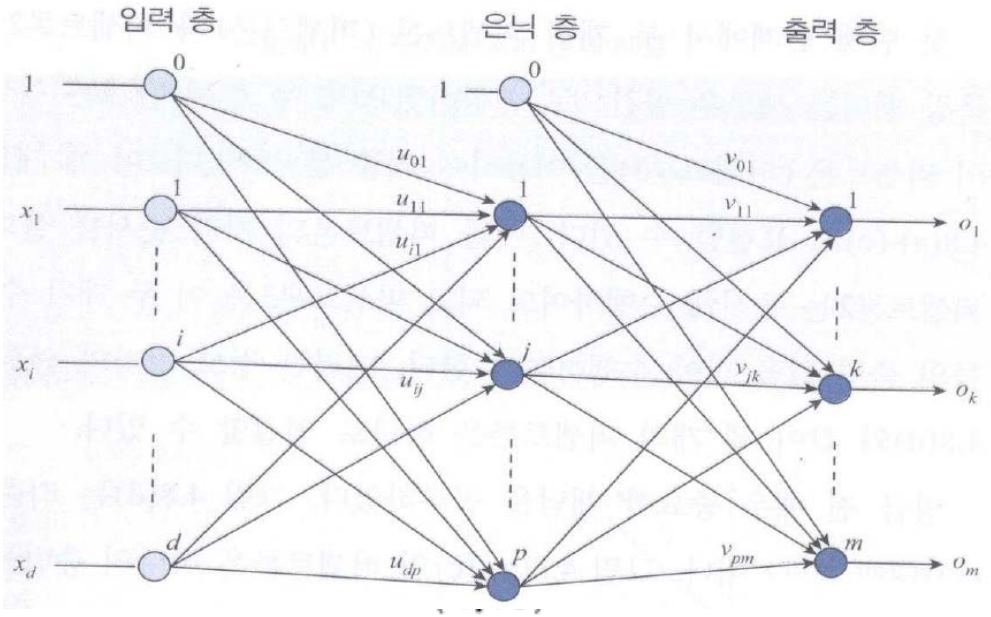
$$\left.\begin{aligned} z\_sum_j &= \sum_{i=1}^{n} x_i u_{ij} + u_{0j} \\ z_j &= \tau(z\_sum_j) \end{aligned}\right\} \quad (4.12)$$

활성함수
(activation function)

출력 층의 k번째 노드, $1 \le k \le m$:

$$\left.\begin{aligned} o\_sum_k &= \sum_{j=1}^{p} z_j v_{jk} + v_{0k} \\ o_k &= \tau(o\_sum_k) \end{aligned}\right\} \quad (4.13)$$

활성함수
(activation function)



입력 **x**　　은닉층 출력**z**　　출력벡터 **o**

입력층에서 은닉층
매핑 함수(p(.))

은닉층에서 출력층
매핑 함수(q(.))

다층퍼셉트론 - 구조와 원리

활성 함수 (activation function)

– 다양한 비선형 함수 사용 가능 : 예) 시그모이드 함수

- 이진 시그모이드 함수:

$$\left.\begin{aligned}\tau_1(x) &= \frac{1}{1+e^{-\alpha x}} \\ \tau_1'(x) &= \alpha\tau_1(x)(1-\tau_1(x))\end{aligned}\right\} \quad (4.14)$$

Derivative of Sigmoid:
http://www.ai.mit.edu/courses/6.892/lecture8-html/sld015.htm

- 양극 시그모이드 함수:

$$\left.\begin{aligned}\tau_2(x) &= \frac{2}{1+e^{-\alpha x}} - 1 \\ \tau_2'(x) &= \frac{\alpha}{2}(1+\tau_2(x))(1-\tau_2(x))\end{aligned}\right\} \quad (4.15)$$

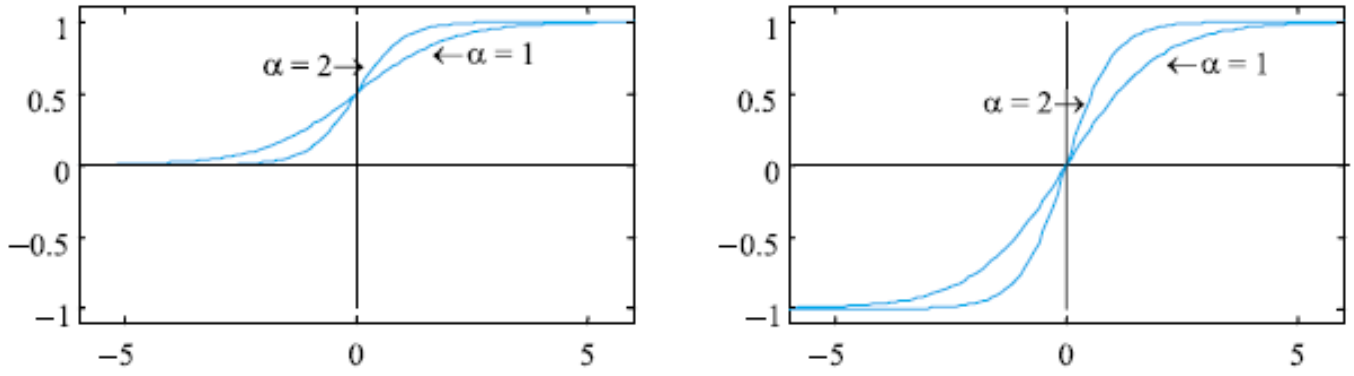
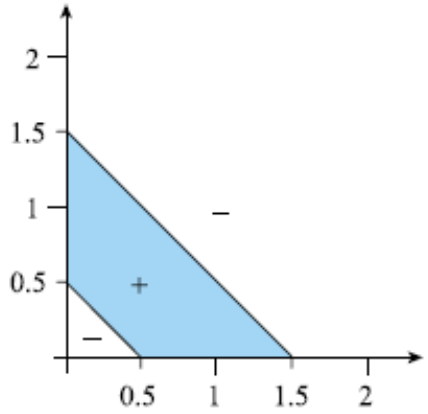


그림 4.10 활성 함수로 널리 사용되는 두 가지 시그모이드 함수

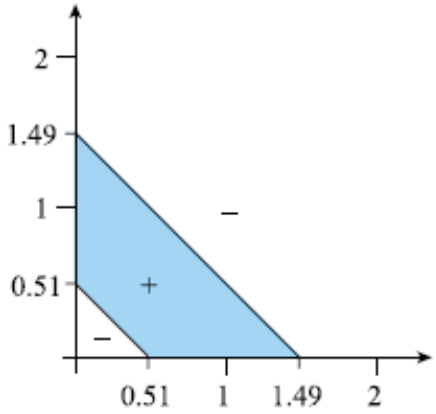
다층퍼셉트론 - 구조와 원리

다층 퍼셉트론의 공간 분할 능력

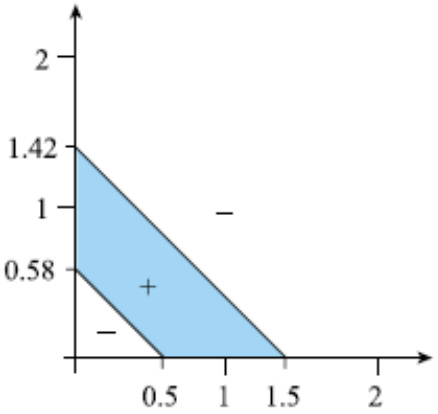
– 활성 함수에 따른 공간 분할



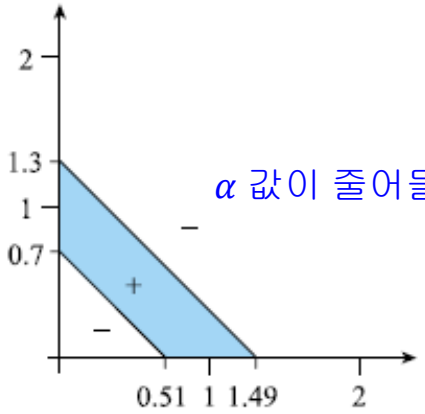
(a) 계단 함수 (양극 시그모이드 α=∞)



(b) 양극 시그모이드 α=5



(c) 양극 시그모이드 α=3



(d) 양극 시그모이드 α=2.5

α 값이 줄어들 수록 w1 class 영역이 줄어들고 있음

그림 4.11 활성 함수에 따른 다층 퍼셉트론의 공간 분할 능력

Feed-Forward MLP

- **FFMLP (Feed-Forward MLP) 의 아키텍처**
  - 은닉층은 몇 개로?
  - 층간의 연결은 어떻게?
  - 각 층의 노드는 몇 개로?
  - 어떤 활성 함수 사용할까?

학습

MLP의 학습이란?

- MLP 학습이란? 훈련 집합 $X = \{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots, (\mathbf{x}_N, \mathbf{t}_N)\}$이 주어졌을 때 이들을 분류하는 다층 퍼셉트론 (즉 $\mathbf{u}$와 $\mathbf{v}$)을 찾아라. $(\mathbf{x}_i, \mathbf{t}_i)$에서 $\mathbf{x}_i$는 특징 벡터이고 $\mathbf{t}_i$는 부류 표지 벡터로서 class label vector (또는 목적 벡터라고도 target vector 함) $\mathbf{x}_i \in \omega_j$이면 $\mathbf{t}_i = (0, \cdots, \overset{j}{1}, \cdots, 0)^T$이다. 즉 j 번째 요소만 1이고 나머지 요소는 모두 0을 갖는다. 이것은 이진 모드를 사용할 때의 값이고 만일 양극 모드를 사용한다면 $\mathbf{t}_i = (-1, \cdots, \overset{j}{1}, \cdots, -1)^T$로 하면 된다.

패턴 인식에서 일반적인 학습 알고리즘 설계 과정

- 단계 1: 분류기 구조 정의와 분류 과정의 수학식 정의
- 단계 2: 분류기 품질 측정용 비용함수 $J(\Theta)$ 정의
- 단계 3: $J(\Theta)$를 최적화하는 Θ를 찾는 알고리즘 설계

학습

- **단계 1:** 분류기 구조 정의와 분류 과정의 수학식 정의
  - (4.12)와 (4.13)의 전방 계산이 분류기의 식
  - 매개변수 집합 Θ={u, v}

은닉 층의 j번째 노드, $1 \le j \le p$:

$$\left.\begin{aligned} z\_sum_j &= \sum_{i=1}^{n} x_i u_{ij} + u_{0j} \\ z_j &= \tau(z\_sum_j) \end{aligned}\right\} \quad (4.12)$$

출력 층의 k번째 노드, $1 \le k \le m$:

$$\left.\begin{aligned} o\_sum_k &= \sum_{j=1}^{p} z_j v_{jk} + v_{0k} \\ o_k &= \tau(o\_sum_k) \end{aligned}\right\} \quad (4.13)$$

- **단계 2:** 비용 함수 정의

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2 \quad (4.16)$$

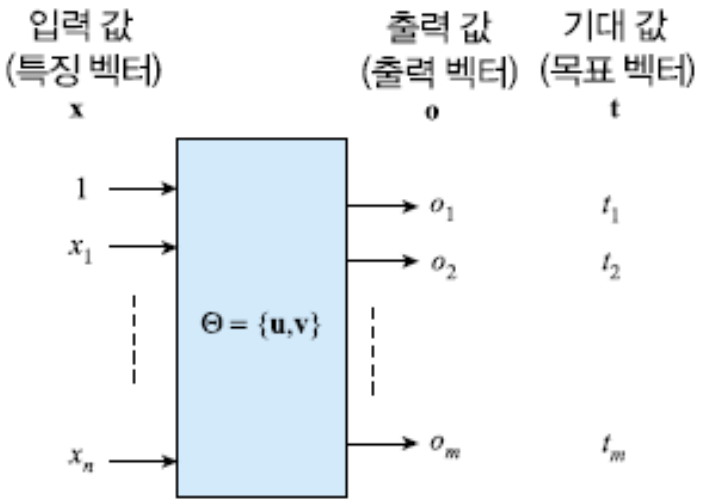


그림 4.12 다층 퍼셉트론의 입력, 출력, 그리고 기대값

단계 3: (최적 해 찾기) J(Θ)를 최적화하는 Θ를 찾는 알고리즘 설계

– 비용함수의 오류를 줄이는 방향으로 Θ를 수정해 나감

$$\left.\begin{aligned}\mathbf{v}(h+1) &= \mathbf{v}(h) + \Delta\mathbf{v} = \mathbf{v}(h) - \rho\frac{\partial E}{\partial \mathbf{v}} \\ \mathbf{u}(h+1) &= \mathbf{u}(h) + \Delta\mathbf{u} = \mathbf{u}(h) - \rho\frac{\partial E}{\partial \mathbf{u}}\end{aligned}\right\} \quad (4.17)$$

알고리즘 [4.4] 다층 퍼셉트론 (MLP) 학습

입력: 훈련 집합 $X = \{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots, (\mathbf{x}_N, \mathbf{t}_N)\}$, 학습률 ρ

출력: 가중치 $\mathbf{u}$와 $\mathbf{v}$

알고리즘:

1. $\mathbf{u}$와 $\mathbf{v}$를 초기화한다.
2. **repeat** {
3. **for** (X의 샘플 각각에 대해) {
4. (4.12)와 (4.13)으로 전방 계산을 한다.
5. $\frac{\partial E}{\partial \mathbf{v}}$와 $\frac{\partial E}{\partial \mathbf{u}}$를 계산한다.
6. (4.17)로 새로운 $\mathbf{u}$와 $\mathbf{v}$를 계산한다.
7. }
8. } **until** (stop-condition);

라인 5를 어떻게?

학습

v_{jk} 갱신값 Δv_{jk} 의 유도

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2$$

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial(0.5\sum_{r=1}^{m}(t_r - o_r)^2)}{\partial v_{jk}}$$

$$= \frac{\partial(0.5(t_k - o_k)^2)}{\partial v_{jk}}$$

특정 출력노드 k에서

$$= -(t_k - o_k)\frac{\partial o_k}{\partial v_{jk}}$$

t_k는 상수임

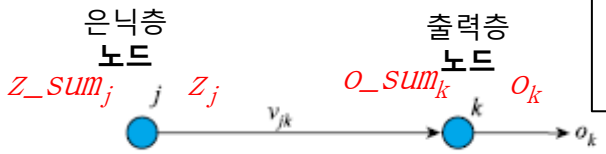
출력 $o_k = \tau(o\_sum_k)$

$$= -(t_k - o_k)\frac{\partial \tau(o\_sum_k)}{\partial v_{jk}}$$

$$= -(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial v_{jk}}$$

$$= -(t_k - o_k)\tau'(o\_sum_k)z_j$$

$$o\_sum_k = \sum_{j=1}^{p} z_j v_{jk} + v_{0k}$$



은닉노드 j와 출력 노드 k 사이의 weight 값(v_{jk})의 변화가 error에 미치는 영향

$$\delta_k = (t_k - o_k)\tau'(o\_sum_k), 1 \le k \le m \quad (4.18)$$

$$\Delta v_{jk} = -\rho\frac{\partial E}{\partial v_{jk}} = \rho\delta_k z_j, 0 \le j \le p, 1 \le k \le m \quad (4.19)$$

$$\left.\begin{aligned}\mathbf{v}(h+1) &= \mathbf{v}(h) + \Delta\mathbf{v} = \mathbf{v}(h) - \rho\frac{\partial E}{\partial \mathbf{v}} \\ \mathbf{u}(h+1) &= \mathbf{u}(h) + \Delta\mathbf{u} = \mathbf{u}(h) - \rho\frac{\partial E}{\partial \mathbf{u}}\end{aligned}\right\}$$

은닉 층의 j번째 노드, $1 \le j \le p$:

$$\left.\begin{aligned}z\_sum_j &= \sum_{i=1}^{n} x_i u_{ij} + u_{0j} \\ z_j &= \tau(z\_sum_j)\end{aligned}\right\} \quad (4.12)$$

출력 층의 k번째 노드, $1 \le k \le m$:

$$\left.\begin{aligned}o\_sum_k &= \sum_{j=1}^{p} z_j v_{jk} + v_{0k} \\ o_k &= \tau(o\_sum_k)\end{aligned}\right\} \quad (4.13)$$

u_{ij}를 위한 갱신값 Δu_{ij}의 유도

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2$$

$$\frac{\partial E}{\partial u_{ij}} = \frac{\partial(0.5\sum_{k=1}^{m}(t_k - o_k)^2)}{\partial u_{ij}}$$

$$= -\sum_{k=1}^{m}(t_k - o_k)\frac{\partial o_k}{\partial u_{ij}}$$

$$= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial u_{ij}} \quad \text{출력 } o_k = \tau(o\_sum_k)$$

$$= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial z_j}\frac{\partial z_j}{\partial u_{ij}} \quad v_{jk} = \frac{\partial o\_sum_k}{\partial z_j}$$

$$= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)v_{jk}\frac{\partial z_j}{\partial u_{ij}}$$

$$= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)v_{jk}\tau'(z\_sum_j)x_i \quad \frac{\partial z_j}{\partial u_{ij}} = \frac{\partial(\tau(z\_sumj))}{\partial u_{ij}} = \tau'(\,) * \frac{\partial(z\_sumj)}{\partial u_{ij}}$$

$$= -\sum_{k=1}^{m}\delta_k v_{jk}\tau'(z\_sum_j)x_i$$

$$\delta_k = (t_k - o_k)\tau'(o\_sum_k)$$

은닉 층의 j번째 노드, $1 \le j \le p$:

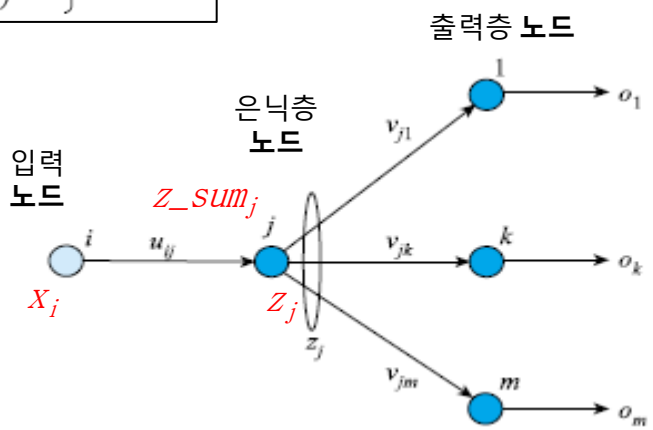
$$z\_sum_j = \sum_{i=1}^{n} x_i u_{ij} + u_{0j}$$
$$z_j = \tau(z\_sum_j) \qquad (4.12)$$

출력 층의 k번째 노드, $1 \le k \le m$:

$$o\_sum_k = \sum_{j=1}^{p} z_j v_{jk} + v_{0k}$$
$$o_k = \tau(o\_sum_k) \qquad (4.13)$$

(4.13)

(4.12)



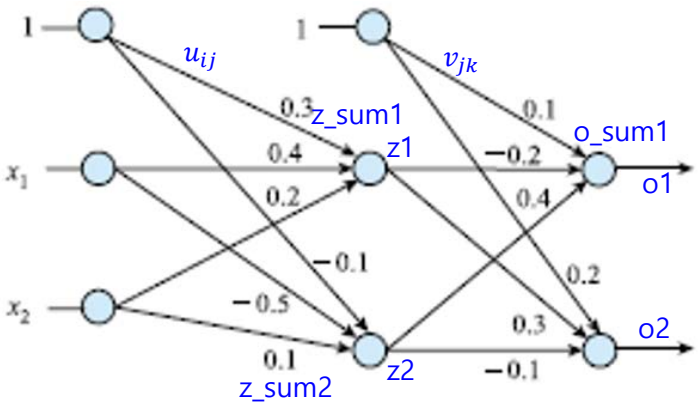
u_{ij}가 미치는 영향

$$\eta_j = \tau'(z\_sum_j)\sum_{k=1}^{m}\delta_k v_{jk}, 1 \le j \le p \qquad (4.20)$$

$$\Delta u_{ij} = -\rho\frac{\partial E}{\partial u_{ij}} = \rho\eta_j x_i, 0 \le i \le d, 1 \le j \le p \qquad (4.21)$$

학습

[알고리즘 4.5]
다중퍼셉트론 학습을위한 오류 역전파 알고리즘 (패턴 모드)



은닉층 노드 수 d개 (2개)

은닉층 노드 수 p개 (2개)

출력층 노드 수 m개 (2개)

입력: 훈련 집합 $X = \{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), \cdots, (\mathbf{x}_N, \mathbf{t}_N)\}$, 학습률 ρ

출력: 가중치 $\mathbf{u}$와 $\mathbf{v}$

알고리즘:

// 초기화

1. $\mathbf{u}$와 $\mathbf{v}$를 초기화한다.
2. $x_0 = z_0 = 1$; // 바이어스
3. **repeat** {
4. **for** (X의 샘플 각각에 대해) {
5. 현재 샘플을 $\mathbf{x} = (x_1, x_2, \cdots, x_d)^T$와 $\mathbf{t} = (t_1, t_2, \cdots, t_m)^T$으로 표기한다.

// 전방 계산

6. **for** ($j = 1$ **to** p) $\{ z\_sum_j = \sum_{i=0}^{d} x_i u_{ij}; \quad z_j = \tau(z\_sum_j); \}$ // (4.12)
7. **for** ($k = 1$ **to** m) $\{ o\_sum_k = \sum_{j=0}^{p} z_j v_{jk}; \quad o_k = \tau(o\_sum_k); \}$ // (4.13)

// 오류 역전파

8. **for** ($k = 1$ **to** m) $\delta_k = (t_k - o_k)\tau'(o\_sum_k)$; // (4.18)
9. **for** (모든 $v_{jk}, 0 \le j \le p, 1 \le k \le m$ 에 대해) $\Delta v_{jk} = \rho \delta_k z_j$; // (4.19)
10. **for** (j=1 **to** p) $\eta_j = \tau'(z\_sum_j) \sum_{k=1}^{m} \delta_k v_{jk}$; // (4.20)
11. **for** (모든 $u_{ij}, 0 \le i \le d, 1 \le j \le p$ 에 대해) $\Delta u_{ij} = \rho \eta_j x_i$; // (4.21)

// 가중치 갱신

12. **for** (모든 $v_{jk}, 0 \le j \le p, 1 \le k \le m$ 에 대해) $v_{jk} = v_{jk} + \Delta v_{jk}$; // (4.17)
13. **for** (모든 $u_{ij}, 0 \le i \le d, 1 \le j \le p$ 에 대해) $u_{ij} = u_{ij} + \Delta u_{ij}$; // (4.17)
14. }
15. } **until** (stop-condition);
16. $\mathbf{u}$와 $\mathbf{v}$를 저장한다.

다층 퍼셉트론의 학습

그림 4.13은 $d=2$, $p=2$, 그리고 $m=2$인 아키텍처를 가진 다층 퍼셉트론이다. 가중치는 그림에서처럼 초기화되어 있다고 하자. 활성 함수로 $\alpha=1$인 양극 시그모이드를 사용하고 학습률은 $\rho=0.2$라 한다. 아래 샘플을 가지고 알고리즘 [4.5]의 학습 과정을 살펴보자.

$$\mathbf{x}=(0.7,\ 0.2)^T,\ \mathbf{t}=(-1,1)^T$$

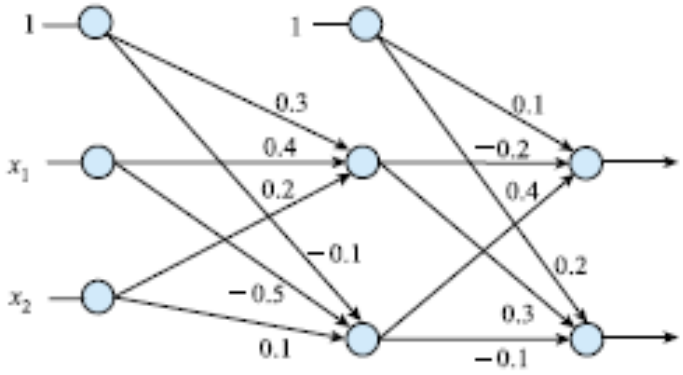
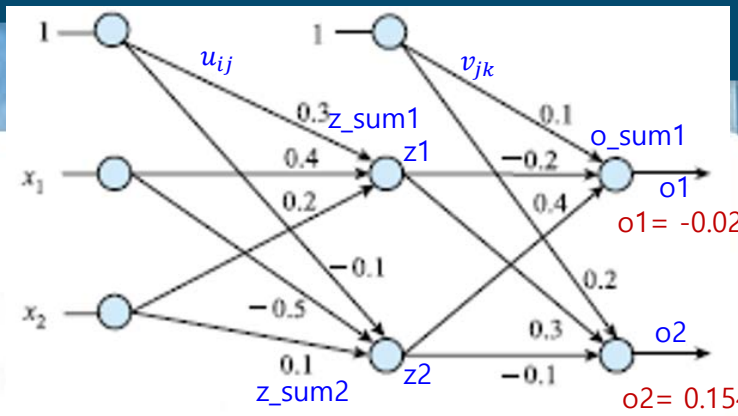


그림 4.13 다층 퍼셉트론 학습 과정의 예시



예제

전방 계산을 해 보자. $\mathbf{x}=(0.7,\ 0.2)^{\mathrm{T}},\ \mathbf{t}=(-1,1)^{\mathrm{T}}$

라인 6:

$z\_sum_1 = 1*0.3+0.7*0.4+0.2*0.2 = 0.62000$

$z\_sum_2 = 1*(-0.1)+0.7*(-0.5)+0.2*0.1 = -0.43000$

$z_1 = \tau_2(0.62000) = 2/(1+e^{-0.62000})-1 = 0.30044$

$z_2 = \tau_2(-0.43000) = 2/(1+e^{0.43000})-1 = -0.21175$

$$\textbf{for } (j=1 \textbf{ to } p)\ \{z\_sum_j = \sum_{i=0}^{n} x_i u_{ij};\quad z_j = \tau(z\_sum_j);\}\quad //\ (4.12)$$

라인 7:

$o\_sum_1 = 1*0.1+0.30044*(-0.2)+(-0.21175)*0.4 = -0.04479$

$o\_sum_2 = 1*0.2+0.30044*0.3+(-0.21175)*(-0.1) = 0.31131$

$o_1 = \tau_2(-0.04479) = -0.02239$

$o_2 = \tau_2(0.31131) = 0.15441$

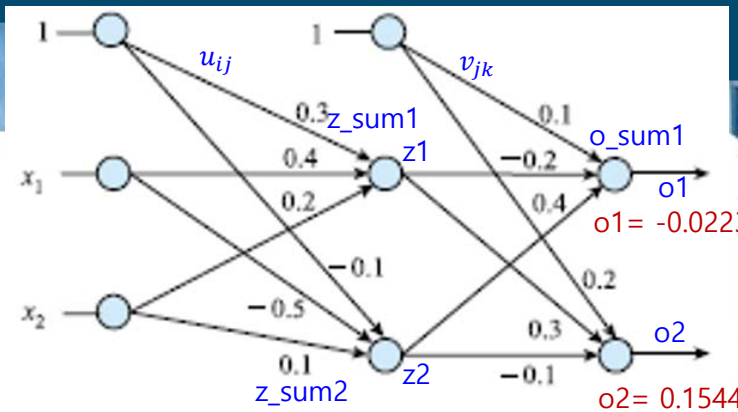
$$\textbf{for } (k=1 \textbf{ to } m)\ \{o\_sum_k = \sum_{j=0}^{p} z_j v_{jk};\quad o_k = \tau(o\_sum_k);\}\ //\ (4.13)$$

이 다층 퍼셉트론은 입력 $\mathbf{x}=(0.7,0.2)^{\mathrm{T}}$에 대해 $\mathbf{o}=(-0.02239,0.15441)^{\mathrm{T}}$을 출력하였다. 기대하는 값 $\mathbf{t}=(-1,1)^{\mathrm{T}}$과의 오류는 아래와 같이 계산할 수 있다.

$$E = 0.5*((-1.0-(-0.02239))^2 + (1.0-0.15441)^2) = 0.83537$$

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2$$

■ 예제 4.4



이제 오류 역전파 단계를 계산해 보자.

라인 8:

$$\delta_1 = (-1.0 + 0.02239)\tau_2'(-0.04479) = -0.97761 * 0.5 * (1 + \tau_2(-0.04479))(1 - \tau_2(-0.04479))$$
$$= -0.48856$$

$$\delta_2 = (1.0 - 0.15441)\tau_2'(0.31131) = 0.84559 * 0.5 * (1 + \tau_2(0.31131))(1 - \tau_2(0.31131))$$
$$= 0.41271$$

라인 9:

$$\Delta v_{01} = 0.2 * (-0.48856) * 1.0 = -0.09771$$

$$\Delta v_{02} = 0.2 * 0.41271 * 1.0 = 0.08254$$

$$\Delta v_{11} = 0.2 * (-0.48856) * 0.30044 = -0.02936$$

$$\Delta v_{12} = 0.2 * 0.41271 * 0.30044 = 0.02480$$

$$\Delta v_{21} = 0.2 * (-0.48856) * (-0.21175) = 0.02069$$

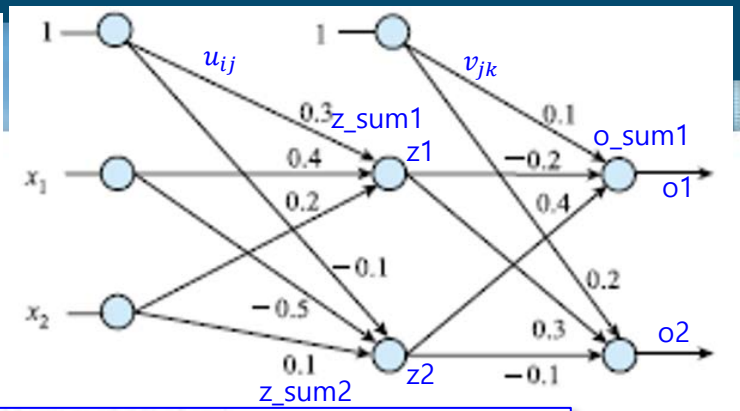
$$\Delta v_{22} = 0.2 * 0.41271 * (-0.21175) = -0.01748$$

```
// 오류 역전파
8.  for (k = 1 to m)  δ_k = (t_k − o_k)τ'(o_sum_k);                          // (4.18)
9.  for (모든 v_jk, 0 ≤ j ≤ p, 1 ≤ k ≤ m 에 대해)  Δv_jk = ρδ_k z_j;          // (4.19)
10. for (j=1 to p)  η_j = τ'(z_sum_j) Σ_{k=1}^{m} δ_k v_jk ;                 // (4.20)
11. for (모든 u_ij, 0 ≤ i ≤ d, 1 ≤ j ≤ p 에 대해)  Δu_ij = ρη_j x_i;          // (4.21)
```

라인 10:

$$\eta_1 = \tau_2'(0.62000)*((-0.48856)*(-0.2)+0.41271*0.3) = 0.10076$$

$$\eta_2 = \tau_2'(-0.43000)*((-0.48856)*(0.4)+0.41271*(-0.1)) = -0.11304$$



라인 11: $\Delta u_{ij} = \rho \eta_j x_i$

$$\Delta u_{01} = 0.2*0.10076*1.0 = 0.02015$$

$$\Delta u_{02} = 0.2*(-0.11304)*1.0 = -0.02261$$

$$\Delta u_{11} = 0.2*0.10076*0.7 = 0.01411$$

$$\Delta u_{12} = 0.2*(-0.11304)*0.7 = -0.01583$$

$$\Delta u_{21} = 0.2*0.10076*0.2 = 0.00403$$

$$\Delta u_{22} = 0.2*(-0.11304)*0.2 = -0.00452$$

// 오류 역전파

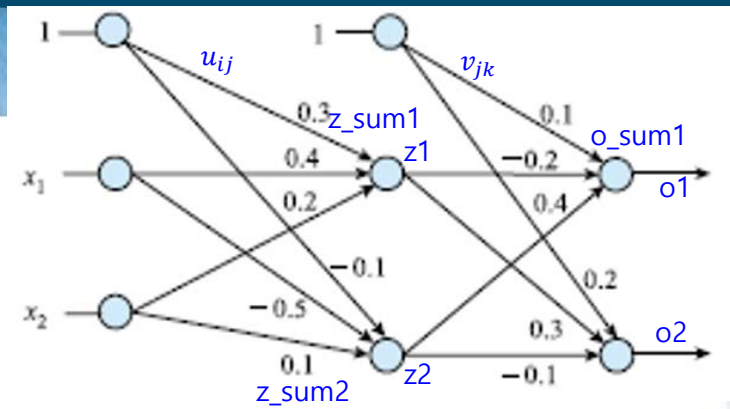
8. **for** $(k = 1$ **to** $m)$ $\delta_k = (t_k - o_k)\tau'(o\_sum_k)$; // (4.18)

9. **for** (모든 $v_{jk}, 0 \le j \le p, 1 \le k \le m$ 에 대해) $\Delta v_{jk} = \rho \delta_k z_j$; // (4.19)

10. **for** $(j=1$ **to** $p)$ $\eta_j = \tau'(z\_sum_j)\sum_{k=1}^{m} \delta_k v_{jk}$; // (4.20)

11. **for** (모든 $u_{ij}, 0 \le i \le d, 1 \le j \le p$ 에 대해) $\Delta u_{ij} = \rho \eta_j x_i$; // (4.21)

예제



이제 가중치 갱신 단계를 수행해 보자.

라인 12:

$v_{01} = 0.1 - 0.09771 = 0.00229$

$v_{02} = 0.2 + 0.08254 = 0.28254$

$v_{11} = -0.2 - 0.02936 = -0.22936$

$v_{12} = 0.3 + 0.02480 = 0.32480$

$v_{21} = 0.4 + 0.02069 = 0.42069$

$v_{22} = -0.1 - 0.01748 = -0.11748$

라인 13:

$u_{01} = 0.3 + 0.02015 = 0.32015$

$u_{02} = -0.1 - 0.02261 = -0.12261$

$u_{11} = 0.4 + 0.01411 = 0.41411$

$u_{12} = -0.5 - 0.01583 = -0.51583$

$u_{21} = 0.2 + 0.00403 = 0.20403$

$u_{22} = 0.1 - 0.00452 = 0.09548$

// 가중치 갱신

for (모든 $v_{jk}, 0 \le j \le p, 1 \le k \le m$ 에 대해) $v_{jk} = v_{jk} + \Delta v_{jk}$; // (4.17)

for (모든 $u_{ij}, 0 \le i \le d, 1 \le j \le p$ 에 대해) $u_{ij} = u_{ij} + \Delta u_{ij}$; // (4.17)

이 예제를 마치기 전에 학습한 효과를 확인해 보자. 이 작업은 새로 얻은 **u**와 **v**가 좋아졌는지를 확인하는 것이다. 라인 6과 라인 7로 전방 계산을 해보자.

라인 6과 7: 0.3 → 0.32015

$z\_sum_1 = 1.0*0.32015+0.7*0.41411+0.2*0.20403 = 0.65083$

$z\_sum_2 = 1.0*(-0.12261)+0.7*(-0.51583)+0.2*0.09548 = -0.46460$

$z_1 = 0.31440$

$z_2 = -0.22821$

$o\_sum_1 = 1.0*0.00229+0.31440*(-0.22936)+(-0.22821)*0.42069 = -0.16582$

$o\_sum_2 = 1.0*0.28254+0.31440*(0.32480)+(-0.22821)*(-0.11748) = 0.41147$

$o_1 = \tau_2(-0.04479) = -0.02239$

$o_2 = \tau_2(0.31131) = 0.15441$

$o_1 = -0.08272$

$o_2 = 0.20288$

$\mathbf{o} = (-0.08272, 0.20288)^T$을 얻어 우리가 원하는 $\mathbf{t} = (-1,1)^T$에 가까워졌음을 알 수 있다. 오류도 $E = 0.73840$이 되어 이전보다 줄어들었음을 확인할 수 있다.

학습

- **오류 역전파 알고리즘의 계산 복잡도**
  - $\Theta((d+m)pHN)$
  - H는 세대 수
  - 많은 시간 소요
    - 예) MNIST 필기 숫자 데이터베이스는 N=60000

인식

- 학습된 다층 퍼셉트론을 사용하여, 입력에 대해 인식을 수행
- 인식 알고리즘

$$\left.\begin{array}{l}\mathbf{x}\text{를 } \omega_q\text{로 분류} \\ \text{이때 } q = \arg\max_j o_j, 1 \le j \le m\end{array}\right\}$$

알고리즘 [4.6] 다층 퍼셉트론 (MLP)에 의한 인식

입력: MLP (**u**와 **v**), 미지 패턴 **x**

출력: 부류 ω_q

알고리즘:

1. **u**와 **v**를 읽어 MLP를 설정한다.
2. $x_0 = z_0 = 1$; // 바이어스
3. **for** $(j = 1$ **to** $p)$ $\{z\_sum_j = \sum_{i=0}^{d} x_i u_{ij}$; $z_j = \tau(z\_sum_j)$; $\}$ // 은닉 층
4. **for** $(k=1$ **to** $m)$ $\{o\_sum_k = \sum_{j=0}^{p} z_j v_{jk}$; $o_k = \tau(o\_sum_k)$; $\}$ // 출력 층
5. **x**를 $q = \arg\max_j o_j$ 인 ω_q로 분류한다. // 가장 큰 값을 갖는 부류

- 시간 복잡도 $\Theta((d+m)p)$
  - N에 무관, 빠름

구현과 몇 가지 부연 설명

- **몇 가지 부연 설명**
  - 네트워크 아키텍처 (은닉 노드 개수 등) 관련
  - 가중치 초기화 이슈
  - 언제 종료할 것인가?

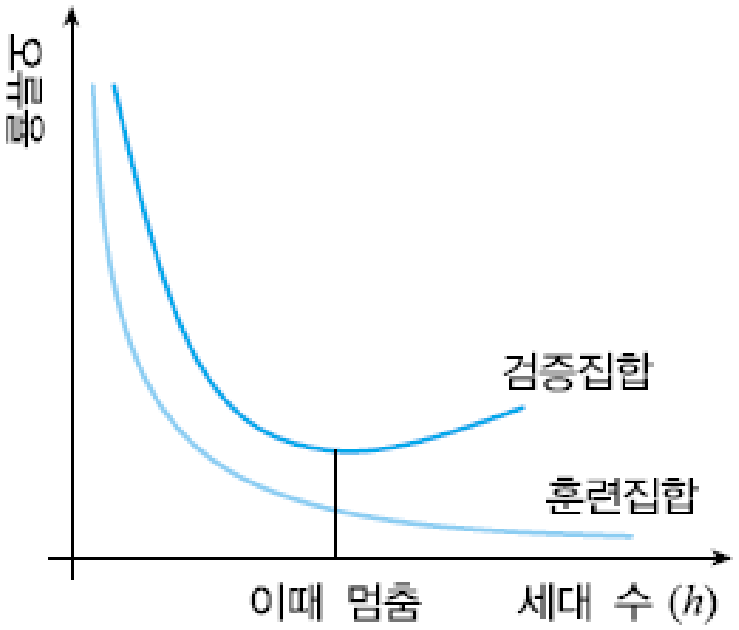


그림 4.15 일반화 기준에 따른 멈춤 조건

구현과 몇 가지 부연 설명

매개변수 설정

- 일반적인 경우에 적용되는 보편적 규칙은 없음
- 경험과 실험을 통해 설정해야 함
- 신경망 성능이 매개변수에 아주 민감하지는 않기 때문에 어느 정도의 실험과 경험을 통해 설정 가능

II. Confusion Matrix

머신러닝 모델 검증 방법

혼동 행렬(Confusion Matrix)

- 혼동 행렬은 훈련된 모델의 성능을 측정하기 위한 Matrix

		True condition	
	Total population	Condition positive	Condition negative
Predicted Condition	Predicted condition positive	True positive	False positive
	Predicted condition negative	False negative	True negative

– TP(True Positive): True를 True로 잘 예측한 것
  - ex) 질병이 있는 사람을 질병이 있다고 예측

– TN(True Negative): False를 False로 잘 예측한 것
  - ex) 질병이 없는 사람을 질병이 없다고 예측

– FP(False Positive): False를 True로 잘못 예측한 것
  - ex) 질병이 없는 사람을 질병이 있다고 잘못 예측

– FN(False Negative): True를 False로 잘못 예측한 것
  - ex) 질병이 있는 사람을 질병이 없다고 잘못 예측

정탐: true positive, true negative제대로 탐지함

- true positive의 의미:
  - "훈련된 모델"에서 positive라고 판단을 내렸을때, 그 판단이 맞는(true) 경우
  - 즉, 질병이 있는 사람이었는데, 입력데이터를 통해, 머신러닝 엔진에서 "질병이 있다(positive)"라고 했고, 그 결과가 맞는(true) 경우임

- true negative의 의미:
  - 머신러닝 엔진에서 "질병이 없다(negative)"라고 했고, 그 결과가 맞는(true) 경우임
  - 즉, 질병이 없는 사람을 잘 맞췄음

오탐: false positive, 엉터리로 탐지함

- false positive의 의미:
  - 머신러닝 엔진에서 "질병이있다(positive)"라고 했는데, 그 결과는 틀린 것임(false)
  - 즉, 질병이 없는 사람을 질병이 있다!라고 틀리게 탐지함

미탐: false negative, 탐지하지 못함

- false negative의 의미:
  - 머신러닝 엔진에서 "질병이 없다(negative)"라고 했는데, 이 결과는 틀렸음
  - 즉, 질병이 있는 사람을 탐지하지 못함 (미탐)

머신러닝 모델 검증 방법

머신러닝 모델 성능 지표

- 정확도(Accuracy): (TP + TN) / Total
  - 전체 데이터 중 True는 True로, False는 False로 잘 예측하는 정도
  - True와 False의 정도가 불균형한 데이터에서는 적절한 평가지표라고 볼 수 없음
    - ex) 신용카드 사기 거래에 대한 예측을 원할 때, 100,000개의 데이터 중 99,900개가 정상 거래고 100개가 사기 거래인 경우, '모든 거래가 정상 거래'라고 판단해도 정확도는 99.9%가 됨

- 정밀도(Precision): TP / (TP + FP)
  - 모델이 "positive다~"라고 한 예측 중에서, 예측한 값이 실제 맞는지(true)에 대한 정도
  - 즉, 탐지했다고 주장하는 것 중에서(즉, positive), 그 예측이 정확히 맞는 경우

- 재현율(Recall) : TP / (TP + FN) (sensitivity라고도 함)
  - 실제로 True인 값들(TP와 FN) 중, 모델이 잘 예측한 값(TP)의 비율

- F1 Score: 2 x (정밀도 x 재현율) / (정밀도 + 재현율)
  - 정밀도와 재현율의 중요성을 동일하게 보고 있음
  - 정밀도와 재현율은 상호 보완할 수 있는 수준에서 적용되어야 함
  - 정밀도와 재현율을 결합한 지표를 F1 Score라고 함

감사합니다

Q & A

부산대학교 전기컴퓨터공학부
부산대학교 사물인터넷 연구센터장
부산대 블록체인 플랫폼 연구센터장
부산대 융합보안대학원 책임교수

김호원
howonkim@pusan.ac.kr