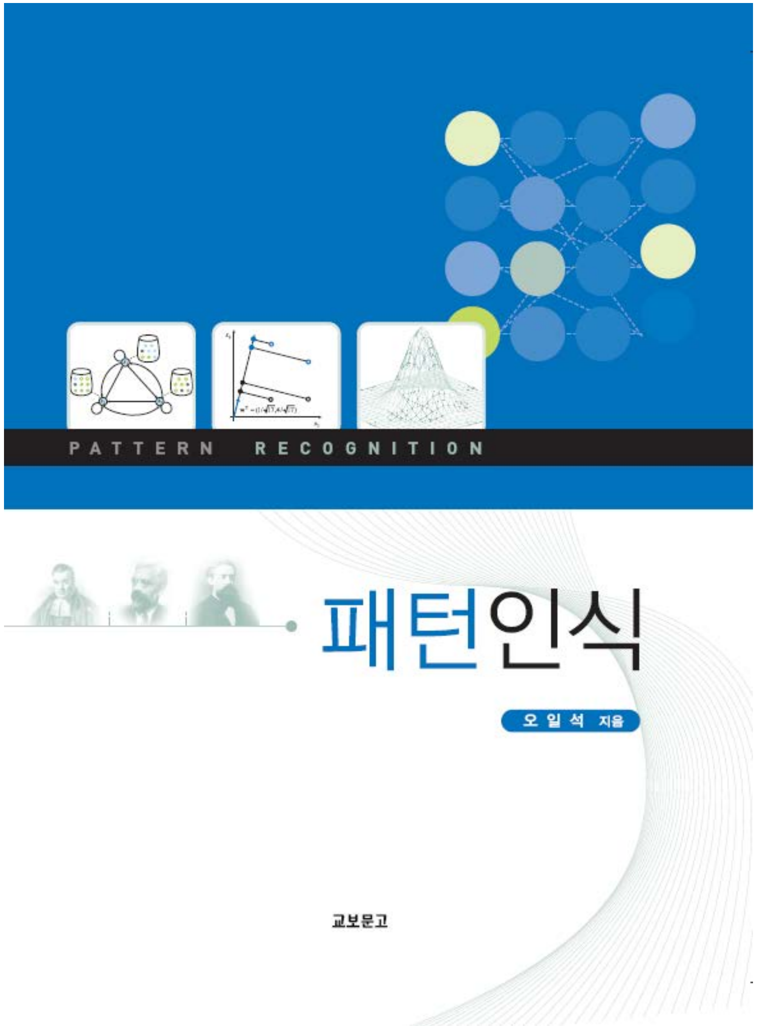
4 장. 신경망

오일석, 패턴인식, 2008. 교보문고.



들어가는 말

- 신경망
  - 1940년대 개발 (디지털 컴퓨터와 탄생 시기 비슷)
  - 인간 지능에 필적하는 컴퓨터 개발이 목표
- 4.1 절
  - 일반적 관점에서 간략히 소개
- 4.2-4.3 절
  - 패턴 인식의 분류 알고리즘으로서 구체적으로 설명
  - 4.2 절: 선형 분류기로서 퍼셉트론
  - 4.3 절: 비선형 분류기로서 다층 퍼셉트론

4.1.1 발상과 전개

- 두 줄기 연구의 시너지
  - 컴퓨터 과학
    - 계산 능력의 획기적 발전으로 지능 처리에 대한 욕구
  - 의학
    - 두뇌의 정보처리 방식 연구 → 얼마간의 성과 (뉴런의 동작 이해 등)

- 뇌의 정보처리 모방하여 인간에 필적하는 지능 컴퓨터에 도전
  - 인공 신경망 (ANN; Artificial Neural Network)이 대표적

4.1.1 발상과 전개

- 컴퓨터와 두뇌의 비교
  - 폰 노이만 컴퓨터
    - 순차 명령어 처리기
  - 두뇌
    - 뉴런으로 구성 (약 10^{11}개, 약 10^{14} 연결 (시냅스))
    - 고도의 병렬 명령어 처리기

(a) 폰 노이만 컴퓨터의 구조 (b) 사람 뇌의 정보 처리 단위인 뉴런

그림 4.1 컴퓨터와 사람

4.1.1 발상과 전개

- 간략한 역사
  - 1943, McCulloch과 Pitts 최초 신경망 제안
  - 1949, Hebb의 학습 알고리즘
  - 1958, Rosenblatt 퍼셉트론
  - Widrow와 Hoff, Adaline과 Madaline
  - 1960대, 신경망의 과대 포장
  - 1969, Minsky와 Papert, Perceptrons라는 저서에서 퍼셉트론 한계 지적
    - 퍼셉트론은 선형 분류기에 불과하고 XOR도 해결 못함
    - 이후 신경망 연구 퇴조
  - 1986, Rumelhart, Hinton, 그리고 Williams, 다층 퍼셉트론과 오류 역전파 학습 알고리즘
    - 필기 숫자 인식같은 복잡하고 실용적인 문제에 높은 성능
    - 신경망 연구 다시 활기 찾음
    - 현재 가장 널리 활용되는 문제 해결 도구

4.1.2 수학적 모델로서의 신경망

- 신경망 특성
  - 학습 가능
  - 뛰어난 일반화 능력
  - 병렬 처리 가능
  - 현실적 문제에서 우수한 성능
  - 다양한 문제 해결 도구 (분류, 예측, 함수 근사화, 합성, 평가, ...)

- 절반의 성공
  - 인간 지능에 필적하는 컴퓨터 만들지 못함
  - 제한된 환경에서 실용적인 시스템 만드는데 크게 기여 (실용적인 수학적 모델로서 자리매김)

4.2 퍼셉트론

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

Imagination is more important than knowledge.

Albert Einstein (1879-1955)

4.2.1 구조와 원리

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

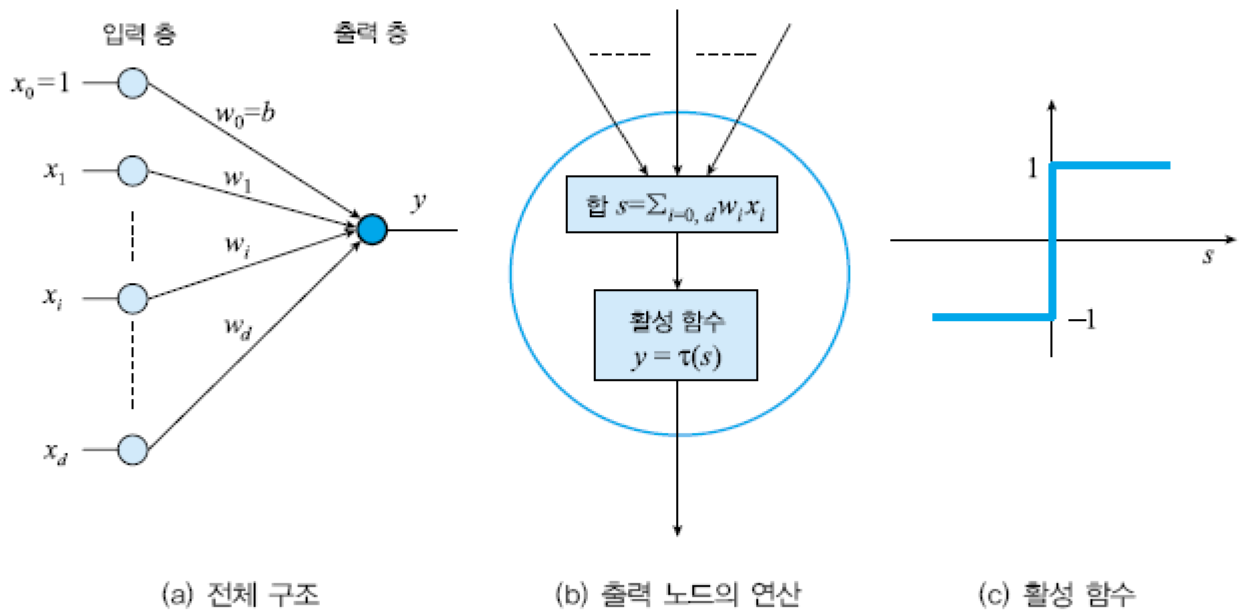


그림 4.2 퍼셉트론의 구조

4.2.1 구조와 원리

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

$$\left.\begin{aligned} &y = \tau(s) = \tau(\sum_{i=1}^{d} w_i x_i + b) = \tau(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \\ &\text{이때}\ \ \tau(s) = \begin{cases} +1, & s \geq 0 \\ -1, & s < 0 \end{cases} \end{aligned}\right\} \qquad (4.2)$$

- 퍼셉트론은 선형 분류기

$$\left.\begin{aligned} d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b > 0 \text{ 이면 } \quad \mathbf{x} \in \omega_1 \\ d(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0 \text{ 이면 } \quad \mathbf{x} \in \omega_2 \end{aligned}\right\} \qquad (4.3)$$

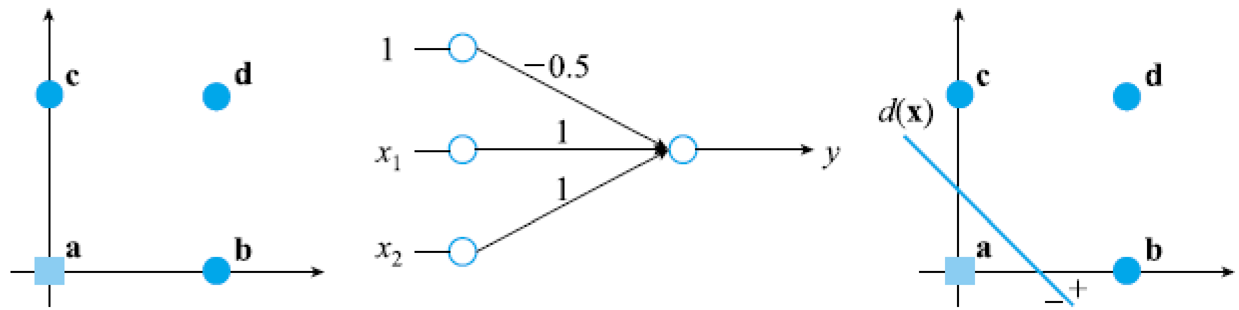
예제 4.1

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

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

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

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



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

그림 4.3 퍼셉트론의 예

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

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

- 샘플 a를 인식해 보자. 맞추나? $y=\tau(\mathbf{w}^{\mathrm{T}}\mathbf{c}+b)=\tau((1,1)\begin{pmatrix}0\\1\end{pmatrix}-0.5)=\tau(0.5)=1$
- 나머지 b, c, d는?

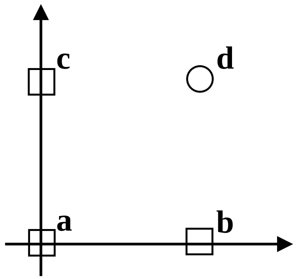
4.2.2 학습과 인식

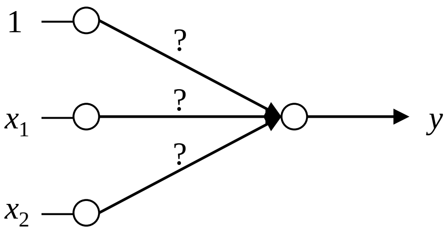
- 퍼셉트론 학습이란?
  - 퍼셉트론 학습이란? 훈련 집합 $X = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$이 주어졌을 때 이들을 모두 옳게 분류하는 퍼셉트론 (즉 $\mathbf{w}$와 b)을 찾아라. 샘플 $(\mathbf{x}_i, t_i)$에서 $\mathbf{x}_i$는 특징 벡터이고 t_i는 부류 표지로서 $\mathbf{x}_i \in \omega_1$이면 $t_i = 1$이고 $\mathbf{x}_i \in \omega_2$이면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

  - 예) AND 분류 문제

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

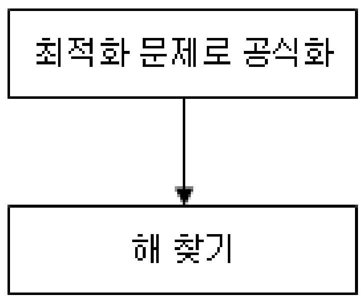
$t_a= -1$ $t_b= -1$ $t_c= -1$ $t_d=1$





4.2.2 학습과 인식

- 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



단계 1과 2

단계 3

4.2.2 학습과 인식

- 단계 1
  - 식 (4.2)
  - 매개변수 집합 $\Theta=\{\mathbf{w}, b\}$

- 단계 2
  - 분류기 품질을 측정하는 $J(\Theta)$를 어떻게 정의할 것인가?

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

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

4.2.2 학습과 인식

- 단계 3
  - $J(\Theta)=0$인 Θ를 찾아라.
  - 내리막 경사법 (Gradient descent method)
    - 현재 해를 $-\partial/\partial\Theta$ 방향으로 이동
    - 학습률 ρ를 곱하여 조금씩 이동

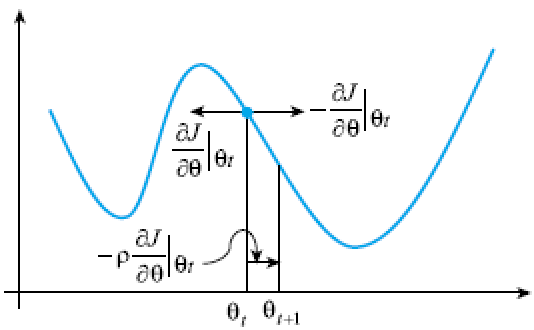


그림 11.9 내리막 경사법

4.2.2 학습과 인식

- 알고리즘 스케치
  - 초기해를 설정한다.
  - 멈춤조건이 만족될 때까지 현재 해를 $-\partial/\partial\Theta$ 방향으로 조금씩 이동시킨다.
- 알고리즘에 필요한 수식들

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

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

$$\downarrow$$

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

←퍼셉트론 학습 규칙 (델타 규칙)

4.2.2 학습과 인식

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

4.2.2 학습과 인식

- 예제 4.2

w(0)=(-0.5,0.75)^T, b(0)=0.375

① $d(\mathbf{x})= -0.5x_1+0.75x_2+0.375$

$Y=\{\mathbf{a}, \mathbf{b}\}$

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

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

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

$Y=\{\mathbf{a}\}$

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

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

⋮

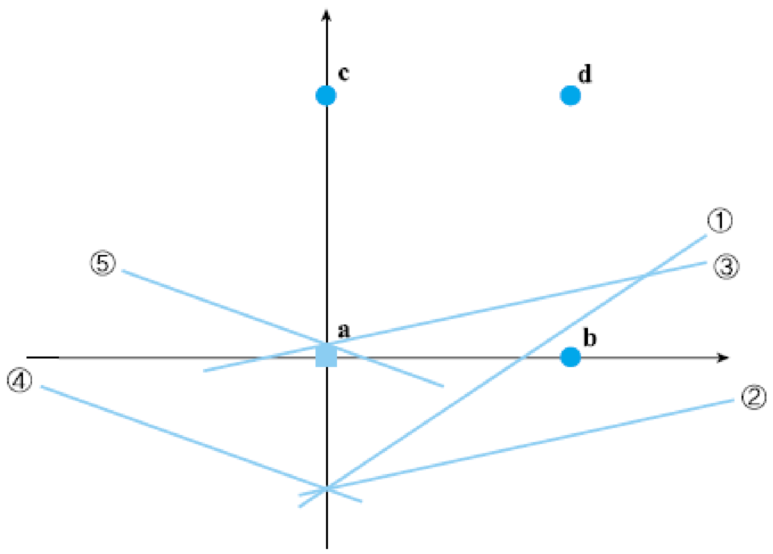


그림 4.4 예제 4.2의 퍼셉트론 학습 과정의 시각화

4.2.2 학습과 인식

- 인식 알고리즘

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

4.2.2 학습과 인식

- 구현
  - 초기값 어떻게?
  - 학습률 어떻게?
  - 패턴 모드와 배치 모드

- 패턴 모드 학습 알고리즘

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

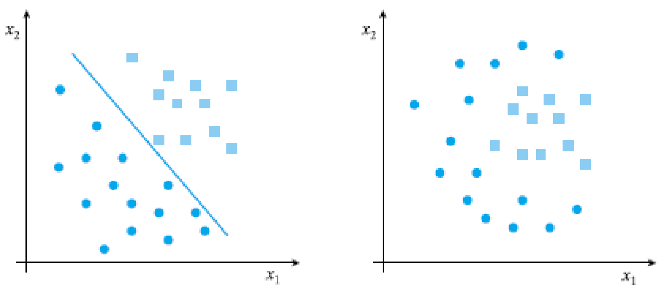
출력: 퍼셉트론 가중치 $\mathbf{w}$, b

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.

4.2.2 학습과 인식

- 포켓 알고리즘
  - 선형 분리 불가능한 상황
  - $J(\Theta)=0$이라는 목표를 버리고, $J(\Theta)$를 최소화하는 목표로 수정



(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:

```
w와 b를 초기화하고, 이들을 w_best와 b_best에 저장한다.
q_best = 0;     // 품질을 0으로 초기화
h = 0;          // 세대 수
repeat {
    for (i = 1 to N) {
        y = τ(w^T x_i + b);   // 식 (4.2)
        if (y ≠ t_i) { w = w + ρ t_i x_i; b = b + ρ t_i; }   // w와 b 갱신
    }
    w와 b로 N 개의 샘플을 인식하여 정인식률 q를 구한다.
    if (q > q_best) {w_best = w; b_best = b; q_best = q;}   // 더 좋은 가중치 발견함
    h = h + 1;
} until (stop-condition);
w = w_best; b = b_best;
w와 b를 저장한다.
```

Frank Rosenblatt

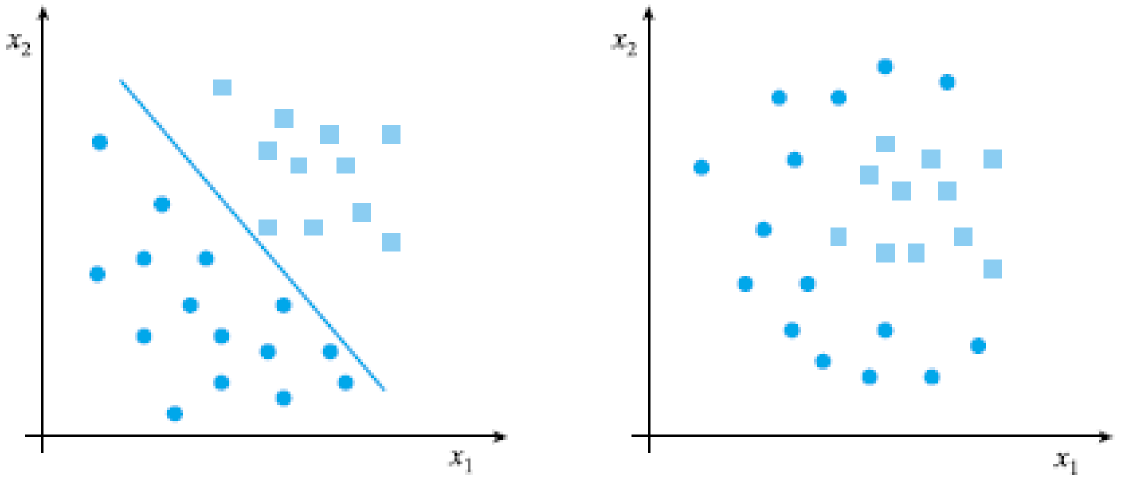
(1928년 7월 11일 ~ 1971년) 미국

Rosenblatt은 초기 신경망 연구에서 독보적인 위치를 차지하고 있다. 그는 Cornell 대학의 교수로 근무하며 퍼셉트론이라는 초기 신경망 모델을 개발하였다. 처음에는 IBM 704에서 시뮬레이션 하였지만 1960년대 초에는 Mark I 퍼셉트론이라는 특수 컴퓨터를 제작하였다. 그는 '뇌 작용 이론 theory of brain mechanism'이라는 학제간 강좌를 여러 해 개설하였고 강의 노트를 정리하여 책으로 출판하였다 [Rosenblatt62]. 이 책에 그의 아이디어가 결집되어 있다. 퍼셉트론은 발표 당시 과다할 정도로 매스컴의 주목을 받았다. 하지만 1969년의 Marvin Minsky와 Seymour Papert의 Perceptrons라는 책의 반론으로 신경망 연구가 침체기에 빠졌고 1980년대 중반에 다층 퍼셉트론이 나올 때까지 소강 상태에 머물러 있었다. 퍼셉트론은 신경망 연구에 한 획을 그은 것으로 평가된다. 현재는 신경망의 역사를 기술할 때 McCulloch-Pitts 모델 → 퍼셉트론 → Adaline → MLP로 이어지는 것으로 보고 있다. Rosenblatt은 스포츠 카를 즐기는 멋쟁이였다고 알려져 있다. 1971년 보트 사고로 사망하였다.

Mark I 퍼셉트론의 영상 입력 장치를 조작하고 있는 Rosenblatt (20*20 영상을 획득)

4.3 다층 퍼셉트론

- 선형 분리 불가능한 상황
  - 퍼셉트론의 한계
  - 그림 4.5(b)에서 퍼셉트론으로 최대 몇 개까지 맞출 수 있을까?



(a) 선형 분리 가능 (b) 선형 분리 불가능

그림 4.5 선형 분리 가능과 불가능

4.3.1 구조와 원리

- XOR 문제
  - 퍼셉트론은 75% 정인식률이 한계
  - 이 한계를 어떻게 극복?
    - 두 개의 퍼셉트론 (결정 직선) 사용

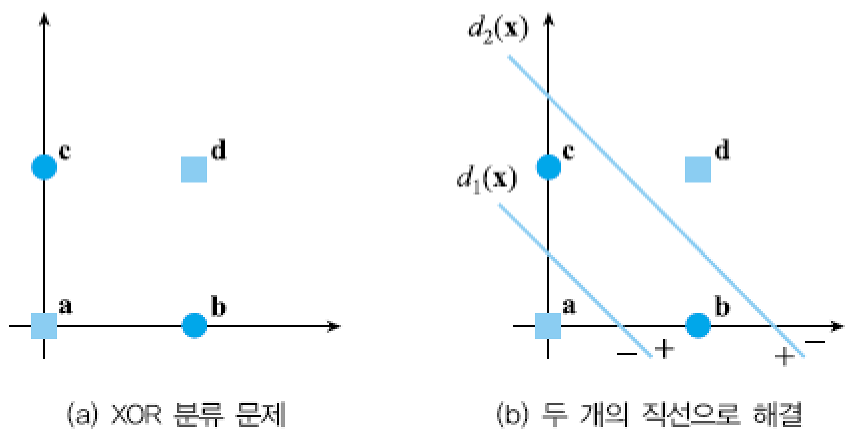


그림 4.6 XOR 분류 문제의 해결

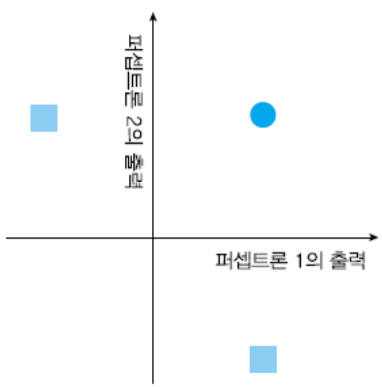
$$\left.\begin{array}{l}\mathbf{w}_1{}^{\mathrm{T}}\mathbf{x}+b_1>0 \text{ 이고 } \mathbf{w}_2{}^{\mathrm{T}}\mathbf{x}+b_2>0 \text{ 이면, } \mathbf{x}\in\omega_1 \\ \mathbf{w}_1{}^{\mathrm{T}}\mathbf{x}+b_1<0 \text{ 이거나 } \mathbf{w}_2{}^{\mathrm{T}}\mathbf{x}+b_2<0 \text{ 이면, } \mathbf{x}\in\omega_2\end{array}\right\}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

4.3.1 구조와 원리

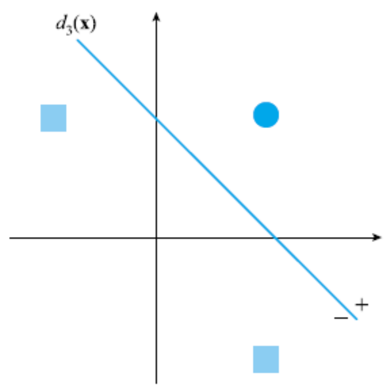
- 두 단계에 걸쳐 문제 해결
  - 단계 1: 원래 특징 공간을 새로운 공간으로 매핑
  - 단계 2: 새로운 공간에서 분류

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) 매핑 공간에서의 샘플 분포



(b) 퍼셉트론3에 의한 영역 분할

그림 4.7 새로운 공간에서의 샘플 분포와 영역 분할

4.3.1 구조와 원리

- 다층 퍼셉트론 (MLP; Multi-layer perceptron)

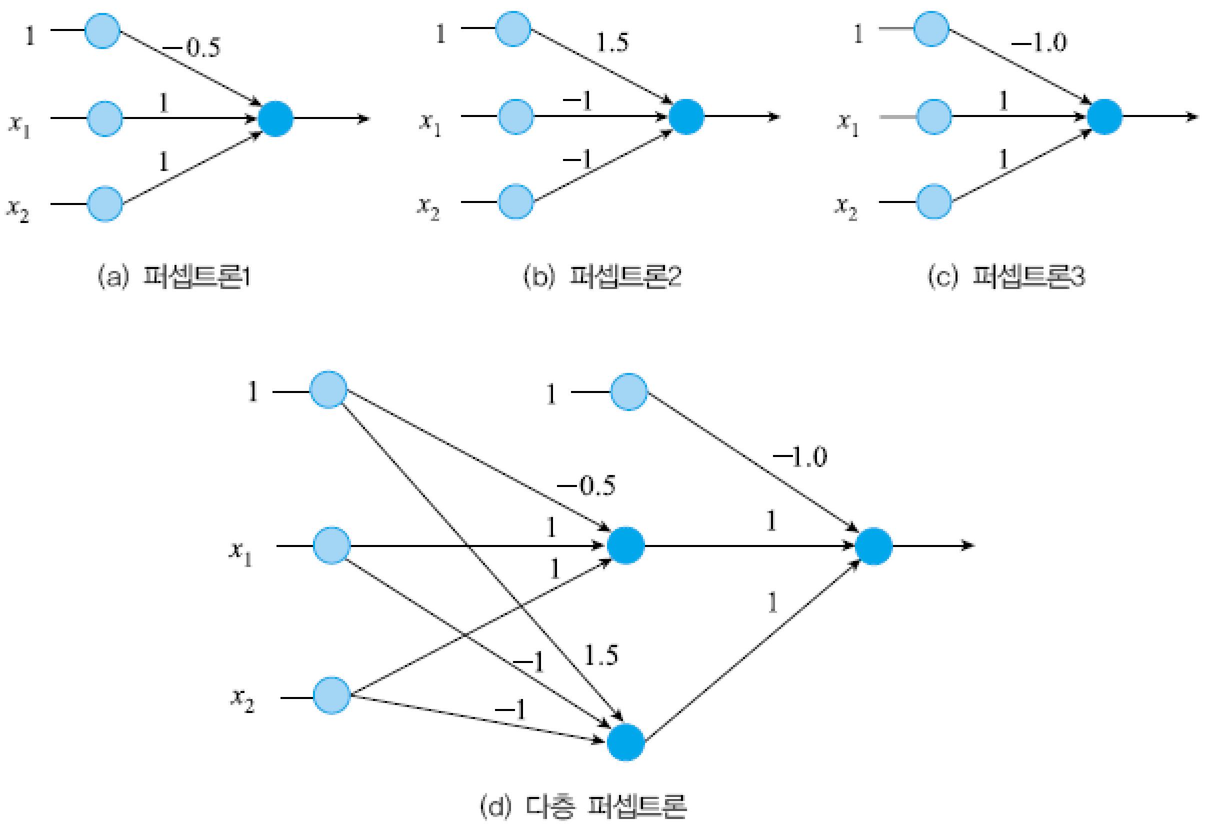


그림 4.8 세 개의 퍼셉트론과 이들을 연결하여 만든 다층 퍼셉트론

4.3.1 구조와 원리

- 다층 퍼셉트론의 아키텍처
  - 입력층, 은닉층, 출력층
  - 가중치: u와 v

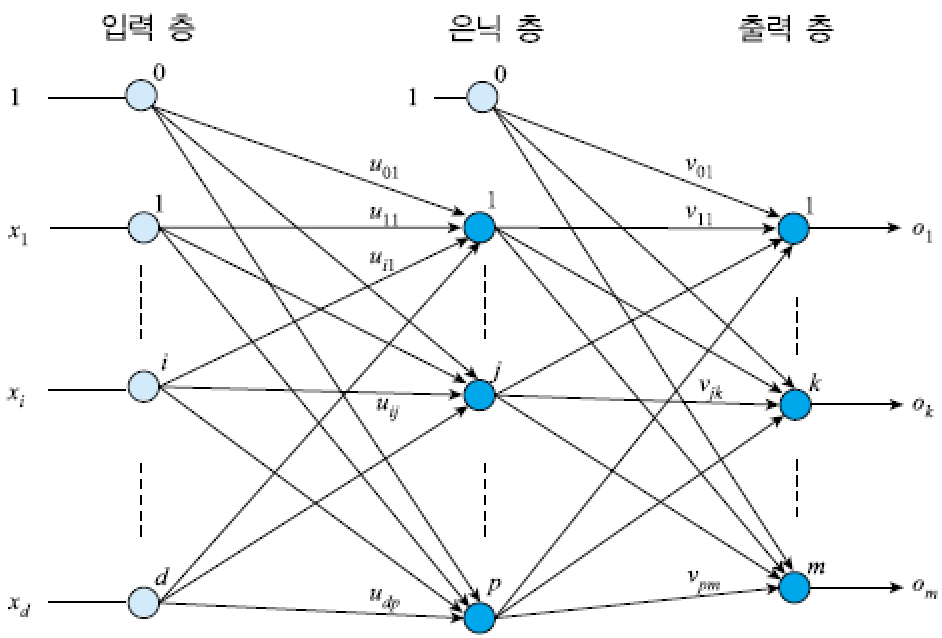


그림 4.9 다층 퍼셉트론의 구조와 표기

4.3.1 구조와 원리

- 신경망은 일종의 함수

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

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

4.3.1 구조와 원리

- 전방 계산 (forward computation)

은닉 층의 j번째 노드, $1 \le j \le p$:

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

출력 층의 k번째 노드, $1 \le k \le m$:

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

4.3.1 구조와 원리

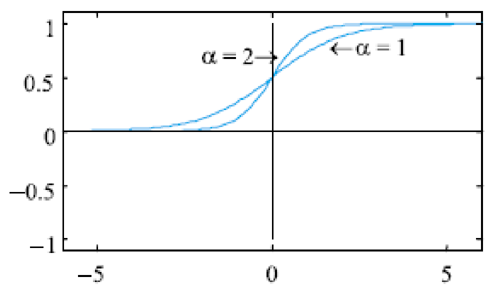
- 활성 함수 (activation function)
  - 시그모이드라는 비선형 함수 사용

이진 시그모이드 함수:

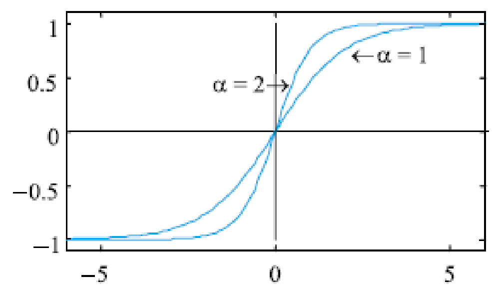
$$\left.\begin{aligned} \tau_1(x) &= \frac{1}{1+e^{-\alpha x}} \\ \tau_1'(x) &= \alpha\tau_1(x)(1-\tau_1(x)) \end{aligned}\right\} \quad (4.14)$$

양극 시그모이드 함수:

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



(a) 이진 시그모이드



(b) 양극 시그모이드

그림 4.10 활성 함수로 널리 사용되는 두 가지 시그모이드 함수

4.3.1 구조와 원리

- 예제 4.3 다층 퍼셉트론의 공간 분할 능력
  - 활성 함수에 따른 공간 분할

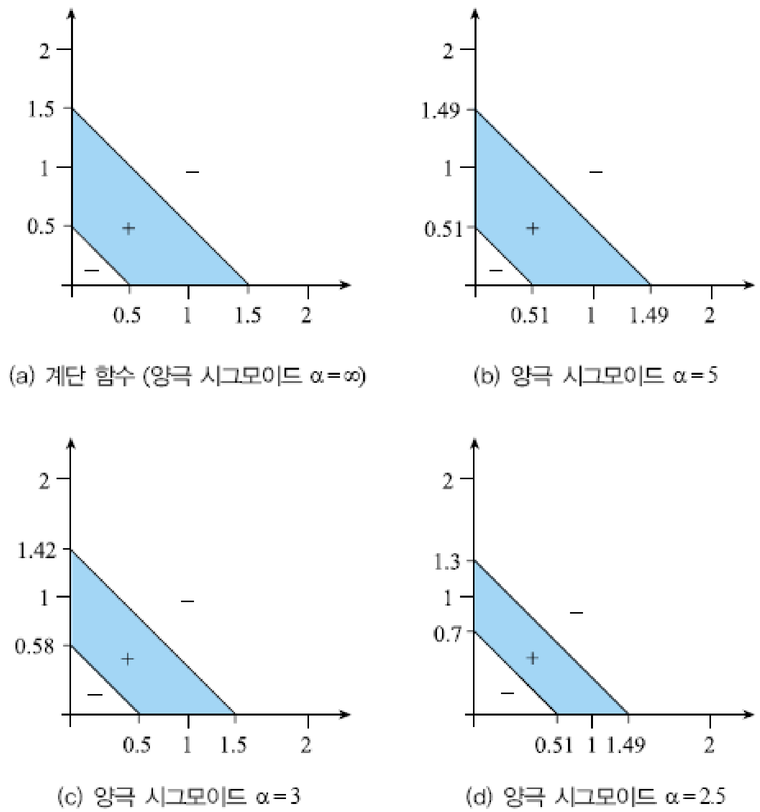


그림 4.11 활성 함수에 따른 다층 퍼셉트론의 공간 분할 능력

4.3.1 구조와 원리

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

4.3.2 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 $\mathbf{t}_i = (0, \cdots, 1, \cdots, 0)^T$이다. 즉 j 번째 요소만 1이고 나머지 요소는 모두 0을 갖는다. 이것은 이진 모드를 사용할 때의 값이고 만일 양극 모드를 사용한다면 $\mathbf{t}_i = (-1, \cdots, 1, \cdots, -1)^T$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

4.3.2 학습

- 단계 1
  - (4.12)와 (4.13)의 전방 계산이 분류기의 식
  - 매개변수 집합 Θ={u, v}

- 단계 2 (비용 함수 정의)

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

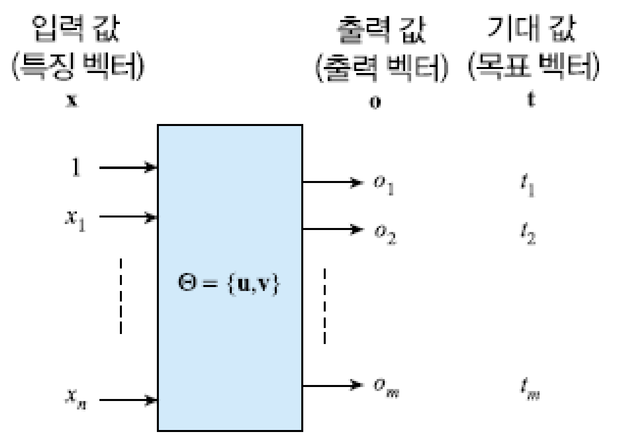


그림 4.12 다층 퍼셉트론의 입력, 출력, 그리고 기대값

4.3.2 학습

- 단계 3 (최적 해 찾음)
  - (4.16)의 오류를 줄이는 방향으로 Θ를 수정해 나감

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

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 **u**와 **v**

알고리즘:

1. **u**와 **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 **u**와 **v**를 계산한다.
7.   }
8. } **until** (stop-condition);

라인 5를 어떻게?

4.3.2 학습

- v_{jk}를 위한 갱신값 Δv_{jk}의 유도

$$
\begin{aligned}
\frac{\partial E}{\partial v_{jk}} &= \frac{\partial(0.5\sum_{r=1}^{m}(t_r - o_r)^2)}{\partial v_{jk}} \\
&= \frac{\partial(0.5(t_k - o_k)^2)}{\partial v_{jk}} \\
&= -(t_k - o_k)\frac{\partial o_k}{\partial v_{jk}} \\
&= -(t_k - o_k)\frac{\partial \tau(o\_sum_k)}{\partial v_{jk}} \\
&= -(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial v_{jk}} \\
&= -(t_k - o_k)\tau'(o\_sum_k)z_j
\end{aligned}
$$

v_{jk}가 미치는 영향

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

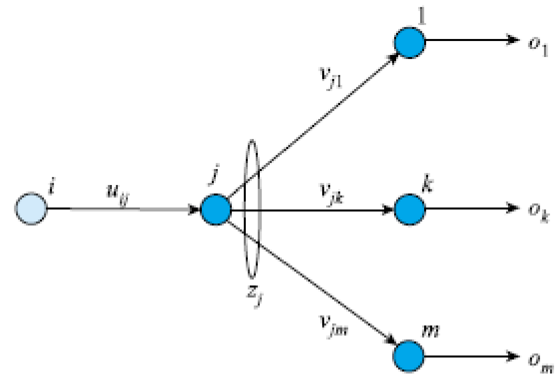
$$\Delta v_{jk} = -\rho\frac{\partial E}{\partial v_{jk}} = \rho\delta_k z_j, 0 \le j \le p, 1 \le k \le m \qquad (4.19)$$

4.3.2 학습

- u_{ij}를 위한 갱신값 Δu_{ij}의 유도

$$
\begin{aligned}
\frac{\partial E}{\partial u_{ij}} &= \frac{\partial(0.5\sum_{k=1}^{m}(t_k - o_k)^2)}{\partial u_{ij}} \\
&= -\sum_{k=1}^{m}(t_k - o_k)\frac{\partial o_k}{\partial u_{ij}} \\
&= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial u_{ij}} \\
&= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)\frac{\partial o\_sum_k}{\partial z_j}\frac{\partial z_j}{\partial u_{ij}} \\
&= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)v_{jk}\frac{\partial z_j}{\partial u_{ij}} \\
&= -\sum_{k=1}^{m}(t_k - o_k)\tau'(o\_sum_k)v_{jk}\tau'(z\_sum_j)x_i \\
&= -\sum_{k=1}^{m}\delta_k v_{jk}\tau'(z\_sum_j)x_i
\end{aligned}
$$

$$\downarrow$$



u_{ij}가 미치는 영향

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

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

4.3.2 학습

오류 역전파 알고리즘

알고리즘 [4.5] 다층 퍼셉트론 (MLP) 학습을 위한 오류 역전파 알고리즘 (패턴 모드)

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 **u**와 **v**

알고리즘:

// 초기화

u와 **v**를 초기화한다.

$x_0 = z_0 = 1$; // 바이어스

repeat {

  for (X의 샘플 각각에 대해) {

    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

    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)

    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

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

    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)

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

    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

    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)

  }

} **until** (stop-condition);

u와 **v**를 저장한다.

4.3.2 학습

■ 예제 4.4 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.

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

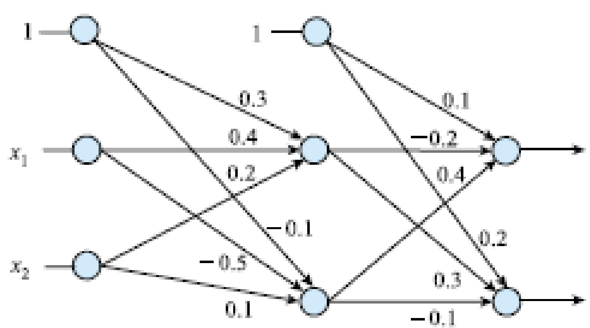


그림 4.13 다층 퍼셉트론 학습 과정의 예시

4.3.2 학습

- 예제 4.4

전방 계산을 해 보자.

라인 6:

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

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

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

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

라인 7:

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

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

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

$o_2 = \tau_2(0.31131) = 0.15441$

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

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

4.3.2 학습

- 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:

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

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

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

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

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

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

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:

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

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

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

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

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

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

예제 4.4

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:

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

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

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

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

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

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

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:

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

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

$$z_1 = 0.31440$$

$$z_2 = -0.22821$$

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

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

$$o_1 = -0.08272$$

$$o_2 = 0.20288$$

$\mathbf{o} = (-0.08272, 0.20288)^T$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. ■■■

4.3.2 학습

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

4.3.3 인식

- 인식 알고리즘

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

알고리즘 [4.6] 다층 퍼셉트론 (MLP)에 의한 인식

입력: MLP ($\mathbf{u}$와 $\mathbf{v}$), 미지 패턴 $\mathbf{x}$

출력: 부류 ω_q

알고리즘:

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

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

4.3.4 구현과 몇 가지 부연 설명

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

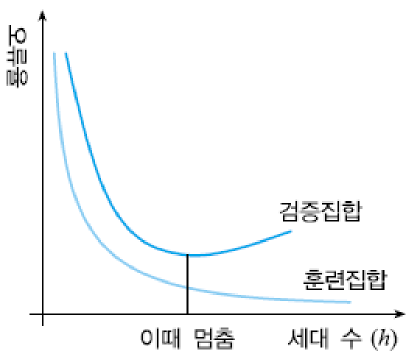


그림 4.15 일반화 기준에 따른 멈춤 조건

  - 목적 벡터의 표현과 활성 함수 (이진 모드와 양극 모드)
  - 샘플 처리 순서
  - 학습률
  - 국소 최적 점 탈출

4.3.4 구현과 몇 가지 부연 설명

- 매개변수 설정
  - 일반적인 경우에 적용되는 보편 규칙은 없다.
  - 경험과 실험을 통해 설정해야 한다.
  - 신경망 성능이 매개변수에 아주 민감하지는 않기 때문에 어느 정도의 실험과 경험을 통해 설정 가능