

# 04. 신경망

## • 신경망

- 1940년대에 개발(디지털 컴퓨터와 탄생 시기 비슷)
- 인간 지능에 필적하는 컴퓨터 개발이 목표

## • 4. 1 절

- 일반적 관점에서 간단히 소개

## • 4. 2 - 4. 3 절

- 패턴 인식의 분류 알고리즘으로서 구체적으로 설명
- 4.2절 : 선형 분류기로서 퍼셉트론
- 4.3절 : 비선형 분류기로서 다층 퍼셉트론

## 4.1 소개

### 4.1.1 뇌상과 전개

컴퓨터 과학 : 계산 능력의 획기적 발전으로 지능 처리에 대한 욕구

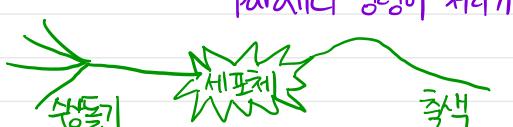
의학 : 두뇌의 정보처리 방식 연구 → 비밀 벗겨지기 시작

뇌의 정보 처리를 모방하는 컴퓨터의 개발

sequential 명령어 처리기



(a) 폰 노이만 컴퓨터의 구조



(b) 사람 뇌의 정보 처리 단위인 뉴런

세포체: 간단한 연산 수행

수송줄기: 다른 뉴런으로부터 신호를 받음

축삭: 다른 뉴런에 신호를 전달

뇌의 정보처리 모방하여 인간에 펼쳐하는 지능 컴퓨터에 도전  
→ 인공 신경망(ANN: Artificial Neural Network)이 대표적

- 간략한 역사

- 1943. McCulloch 과 Pitts 최초 신경망 제안
- 1949. Hebb의 학습 알고리즘
- 1958. Rosenblatt 퍼셉트론
- Widrow 와 Hoff, Adaline 과 Madeline
- 1960년대, 신경망의 고대포장
- 1969, Minsky 와 Papert, Perceptrons라는 저서에서 퍼셉트론 한계 지적

- 퍼셉트론은 선형 분류기에 불가하고 XOR도 해결 못함

- 이후 신경망 연구 되조

- 1986, Rumelhart, Hinton, 그리고 Williams, 다중 퍼셉트론과

- 유동 역전파 학습 알고리즘

- 필기 숫자 인식 같은 복잡하고 실용적인 문제에 높은 성능
- 신경망 연구 다시 활기 찾음
- 현재 가장 널리 활용되는 문제 해결 도구

#### 4.1.2 수학적 모델로서의 신경망

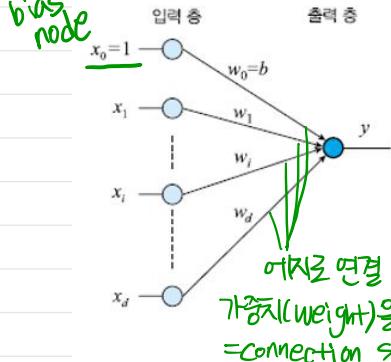
- 신경망 특성
- 학습 가능 - 현실적 문제에서 우수한 성능
- 뛰어난 일반화 능력 - 다양한 문제 해결 도구
- 병렬처리 가능 (분류, 예측, 허브 구사학, 합성, 평가, ...)

## 4.2. 퍼셉트론

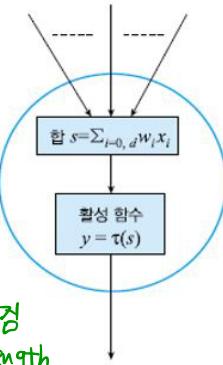
분명한 한계를 가지나, 시너의 초석이 됨

### 4.2.1 구조와 원리

bias node



(a) 전체 구조



(b) 출력 노드의 연산

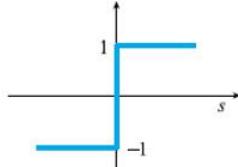
(c) 활성 함수  
 $\tau(\cdot)$ 

그림 4.2 퍼셉트론의 구조

입력층:  $d+1$  개의 노드 (특징 벡터  $x=(x_1, \dots, x_d)^T$ )

출력층: 한 개의 노드(두 개의 부류  $W_1$ 과  $W_2$ 로 분류하는 이진 분류기)

- 노드의 연산

- 입력 노드: 받은 신호를 단순히 전달

- 출력 노드: 합계산과 활성 함수 계산

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

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

(4.2)

특징 벡터  $x = (x_1, x_2, \dots, x_d)^T$

가중치 벡터  $w = (w_1, w_2, \dots, w_d)^T$

(4.2)로 정의되는 퍼셉트론은 선형 분류기여 해당

$$d(x) = w^T x + b > 0 \text{ 이면 } x \in w_1 \quad \}$$

$$\underline{d(x) = w^T x + b < 0 \text{ 이면 } x \in w_2} \quad \}$$

선형 분류기가 사용하는 결정 직선

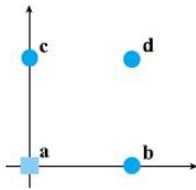
예제 4.1

$$a = (0,0)^T, t_a = -1$$

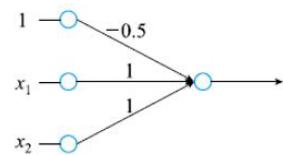
$$b = (1,0)^T, t_b = 1$$

$$c = (0,1)^T, t_c = 1$$

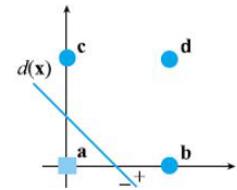
$$d = (1,1)^T, t_d = 1$$



(a) OR 분류 문제



(b) OR 분류기로서 퍼셉트론



(c) 퍼셉트론은 선형 분류기

그림 4.3 퍼셉트론의 예

이 퍼셉트론은  $w = (1,1)^T, b = -0.5$

네 개 샘플 중에  $c = (0,1)^T$ 를 입력해 보자.

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

$$\begin{aligned} \text{결정직선 } d(x) &= w_1 x_1 + w_2 x_2 + b \\ &= x_1 + x_2 - 0.5 \end{aligned}$$

#### 4.2.2 학습과 인식

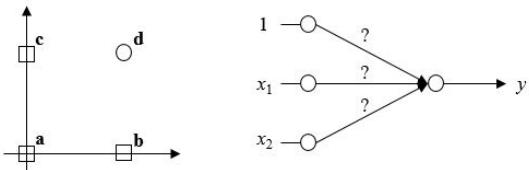
퍼셉트론을 어떻게 학습시킬 것인가?

Training set이 주어졌을 때 이들을 모두 올바르게 분류하는 퍼셉트론을 찾는 문제를 다루어야 한다.

퍼셉트론 학습이란? 훈련 집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n)\}$  이 주어졌을 때 이들을 모두 옳게 분류하는 퍼셉트론(즉  $w$ 와  $b$ )을 찾아라. 샘플  $(x_i, t_i)$ 에서  $x_i$ 는 특징벡터이고  $t_i$ 는 뿌리 표지로서  $x_i \in W_1$  이면  $t_i = 1$ 이고  $x_i \in W_2$  이면  $t_i = -1$ 이다.  $X$ 는 선형 분리 가능하다고 가정한다.

예) AND 분류 문제

$$\begin{aligned} \mathbf{a} &= (0,0)^T & \mathbf{b} &= (1,0)^T & \mathbf{c} &= (0,1)^T & \mathbf{d} &= (1,1)^T \\ t_a &= -1 & t_b &= -1 & t_c &= -1 & t_d &= 1 \end{aligned}$$



- 패턴 인식에서 일반적인 학습 알고리즘 설계 고정
- 단계 1: 분류기 구조 정의와 분류 과정의 수학적 정의. 이 식은 매개변수 집합  $\Theta$ 로 정의.

이미 4.2.1 절에서 수행, 분류 고정은 (4.2)로 표현, 퍼셉트론의 매개변수 집합은  $\Theta = \{W, b\}$

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

퍼셉트론이 범하는 오류율을 비용 함수로 추할 수 있다. 어떤  $\Theta$ 가  $X$ 의 샘플을 모두 옳게 분류한다면  $J(\Theta)$ 는 0이 되어야 한다.  $\Theta_1$ 이  $\Theta_2$ 보다 품질이 좋다면  $J(\Theta_1) < J(\Theta_2)$  이어야 한다.

$$J(\theta) = \sum_{x_k \in Y} (-t_k) (w^T x_k + b) \rightarrow \text{가장 널리 사용되는 비용함수}$$

Y: 오분류된 샘플 집합

case 1)  $t_i = 1$

$x_i$  가 오분류 되었으므로  $w^T x_i + b < 0$  ( $t_i$  이 -1로 봄을 되면)

$$\therefore t_k (w^T x_k + b) < 0 \Rightarrow -t_k (w^T x_k + b) > 0$$

case 2)  $t_i = -1$

$x_i$  가 오분류 되었으니  $w^T x_i + b > 0$

$$\therefore t_k (w^T x_k + b) < 0 \Rightarrow -t_k (w^T x_k + b) > 0$$

즉 모든 경우에서  $-t_k (w^T x_k + b) > 0$  이므로,

Y가 글수록 (즉, 틀리는 샘플의 수가 많을수록) 값이 크고, Y가 공집합일 때는 0이 됨을 알 수 있다.

- 단계 3:  $J(\theta)$ 를 최적화하는  $\theta$ 를 찾는 알고리즘 설계

비용함수의 최소값을 찾는 문제 = 잘못 분류된 샘플들의 총 개수를 최소화  
(0으로)

- Gradient descent method (내선적 경사법) 사용

매 업데이트마다 순간이율기의 일정 비율만큼 움직인다.

학습률 (learning rate) : 한발씩을 기울기에

$J(\theta) = 0$  이 될 때까지 과정 반복

비례해서 얼마나 내린지

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

$w_1, b_1$        $w_0, b_0$

$\theta$ 는  $\{w, b\}$  이므로  $\partial J(\theta)/\partial \theta$ 를 얻기 위해  $w$ 와  $b$ 를 미분해 보자.

$$\frac{\partial J(\theta)}{\partial w} = \sum_{x_k \in Y} (-t_k) x_k$$

$$\frac{\partial J(\theta)}{\partial b} = \sum_{x_k \in Y} (-t_k) \quad (4.6)$$

(4.6)<sup>o</sup> (4.5)에 대입하면 퍼셉트론 학습 규칙을 얻는다.  
= delta rule

$$w(h+1) = w(h) + p(h) \sum_{x_k \in Y} t_k x_k$$

$$b(h+1) = b(h) + p(h) \sum_{x_k \in Y} t_k$$

$b$ 를  $w_0$ 로 표기하고,  $w$ 와  $x$ 에 차원 하나를 추가하여  
 $\hat{w} = (w_0, w_1, \dots, w_d)^T$ ,  $\hat{x} = (x_0, x_1, \dots, x_d)^T = (1, x_1, \dots, x_d)^T$

$$\hat{w}(h+1) = \hat{w}(h) + p(h) \sum_{x_k \in Y} t_k \hat{x}_k \quad (4.7)$$

최적화 문제로 공식화



해 찾기

단계 1 과 2

단계 3

## 알고리즘 [4.1]

## 퍼셉트론 학습 (배치 모드 batch mode)

입력: 훈련 집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ , 학습률  $\rho$

출력: 퍼셉트론 가중치  $w$ ,  $b$

알고리즘:

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

2~10의 repeat 루프를 한번 돌면 모든 샘플을 한 번씩  
살펴보는 셈인데 이를 한 세대(epoch) 이라 부른다.

$$\text{ex 4.2)} \quad w(0) = (-0.5, 0.75)^T, b(0) = 0.375, \rho = 0.4$$

$$a = (0, 0)^T, t_a = -1$$

$$b = (1, 0)^T, t_b = 1$$

$$c = (0, 1)^T, t_c = 1$$

$$d = (1, 1)^T, t_d = 1$$

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

$$T(d(a)) = T(0.375) = 1 \quad \text{오류샘플!}$$

$$T(d(b)) = T(-0.5 + 0.375) = -1$$

$$T(d(c)) = T(0.75 + 0.375) = 1$$

$$T(d(d)) = T(-0.5 + 0.75 + 0.375) = 1$$

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

$$w(1) = w(0) + 0.4(t_a a + t_b b)$$

$$= \begin{pmatrix} -0.5 \\ 0.75 \end{pmatrix} + 0.4 \left[ -\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} -0.1 \\ 0.75 \end{pmatrix}$$

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

$$d(x) = -0.1x_1 + 0.75x_2 + 0.375$$

$$T(d(a)) = T(0.375) = 1$$

$$T(d(b)) = T(-0.1 + 0.375) = 1$$

$$T(d(c)) = T(0.75 + 0.375) = 1$$

$$T(d(d)) = (-0.1 + 0.75 + 0.375) = 1$$

$$Y = \{b\}$$

$$w(2) = w(1) + 0.4(t_a a)$$

$$= \begin{pmatrix} -0.1 \\ 0.15 \end{pmatrix} + 0.4 \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.1 \\ 0.15 \end{pmatrix}$$

$$b(2) = b(1) + 0.4(t_b b) = 0.315 - 0.4 = -0.025$$

$$d(x) = -0.1x_1 + 0.15x_2 - 0.025$$

$$T(d(a)) = T(-0.025) = -1$$

$$T(d(b)) = T(-0.1 - 0.025) = -1$$

$$T(d(c)) = T(0.15 - 0.025) = 1$$

$$T(d(d)) = T(-0.1 + 0.15 - 0.025) = 1$$

$$w(3) = w(2) + 0.4(t_b b) = \begin{pmatrix} -0.1 \\ 0.15 \end{pmatrix} + 0.4 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.3 \\ 0.15 \end{pmatrix}$$

$$b(3) = b(2) + 0.4(t_b b) = -0.025 + 0.4 = 0.315$$

$$d(x) = 0.3x_1 + 0.15x_2 + 0.315$$

$$Y = \{a\}$$

$$d(x) = 0.3x_1 + 0.15x_2 - 0.025$$

$$w = (0.3, 0.15)$$

$$b = -0.025$$

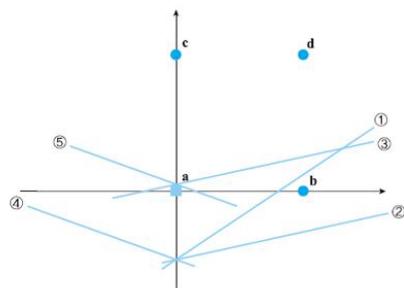


그림 4.4 예제 4.2의 패션트론 학습 과정의 시각화

학습이 끝났으므로 마지막 샘플  $x$ 가 입력되면 학습된 퍼시트론으로  $x$ 를 인식할 수 있다.

- 인식 알고리즘

$$\left. \begin{array}{l} w^T x + b > 0 \text{ 이면, } x \in W_1 \\ w^T x + b < 0 \text{ 이면, } x \in W_2 \end{array} \right\} \quad (4.8)$$

### 4.2.3 구현

4.1의 알고리즘에 대한 부연 설명이다.

1.  $w$ 와  $b$ 의 초기화: 보통  $-0.5 \sim 0.5$  사이의 작은 난수를 생성하여 설정

2. 학습률: 4.1에서는 고정된 학습률을 사용하였으나, 세대(epoch) 수에 따라 적응적으로 설정할 수 있다. 예를 들어, 세대 수  $h$ 에 따라 선형적으로 줄이는 방법이 있다.

$$P(h) = P_s - \frac{(P_s - P_e)}{H} h \quad \begin{matrix} \text{최대 세대 수} \\ | \\ \text{시작 학습률} \end{matrix}$$

$\begin{matrix} \text{현재 세대 수} \\ | \\ \text{끝날 때 학습률} \end{matrix}$

### 3. 패턴모드와 배치모드

패턴모드: 오분류된 모든 샘플을 모은 다음, 이들을 가지고 한꺼번에 가중치를 갱신

배치모드: 샘플을 하나 입력하고 틀리게 인식하면 곧바로 가중치 갱신

입력: 훈련 집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ , 학습률  $\rho$

출력: 퍼셉트론 가중치  $w$ ,  $b$

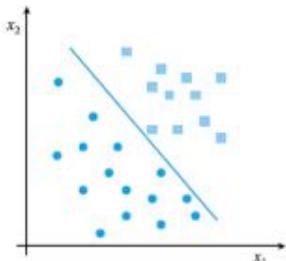
알고리즘:

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

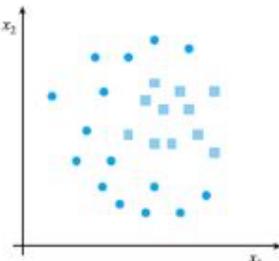


앞에서는 선형 분리가 가능하다고 가정하였으나, 그렇지 않은 경우  
오분류하는 샘플 수를 최소화하기 위한 새로운 목표를 세워야 한다.

이를 포켓 알고리즘(Pocket algorithm)이라고 한다.



(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  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ , 학습률  $\rho$

출력: 퍼셉트론 가중치  $w$ ,  $b$

알고리즘:

1.  $w$ 와  $b$ 를 초기화하고, 이들을  $w_{best}$ 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.          $y = \tau(w^T x_i + b)$ ; // 식 (4.2)
    7.         if ( $y \neq t_i$ ) {  $w = w + \rho t_i x_i$ ;  $b = b + \rho t_i$ } //  $w$ 와  $b$  갱신
    8.     }
  9.      $w$ 와  $b$ 로  $N$  개의 샘플을 인식하여 정인식률  $q$ 를 구한다.
  10.    if ( $q > q_{best}$ ) {  $w_{best} = w$ ;  $b_{best} = b$ ;  $q_{best} = q$ ; } // 더 좋은 가중치 발견함
  11.     $h = h + 1$ ;
12. } until (stop-condition);
13.  $w = w_{best}$ ;  $b = b_{best}$ ;
14.  $w$ 와  $b$ 를 저장한다.

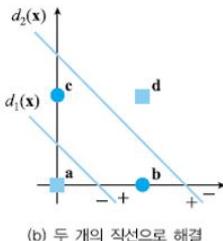
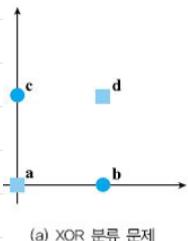
①  $h$ 가 최대 세대 수  $H$ 에 도달하면 끝내는 것  
 ② 여러 세대에 걸쳐 품질(예를 들어 정인식률)이  
 더 이상 좋아지지 않으면 종료

## 4.3 다층 퍼셉트론 (MLP: Multi-layer Perceptron)

- 퍼셉트론의 한계
- 퍼셉트론을 여러 층으로 이어 붙이는 다층 퍼셉트론

### 4.3.1 구조와 원리

- 퍼셉트론은 15% 정인식률이 한계
- 이 한계 극복을 위해 두 개의 퍼셉트론(결정 직선) 사용



$$x_1 + x_2 - 0.5 \\ -x_1 - x_2 + 1.5$$

$w_1^T x + b_1 > 0 \text{ AND } w_2^T x + b_2 > 0, x \in W_1$

 $w_1^T x + b_1 < 0 \text{ OR } w_2^T x + b_2 < 0, x \in W_2$

분류 알고리즘

- 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	
a	0	0	-1	+1	-1
b	1	0	+1	+1	+1
c	0	1	+1	+1	+1
d	1	1	+1	-1	-1

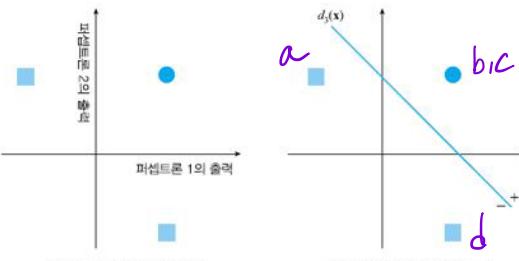
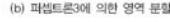


그림 4.7 새로운 공간에서의 샘플 분포와 영역 분할



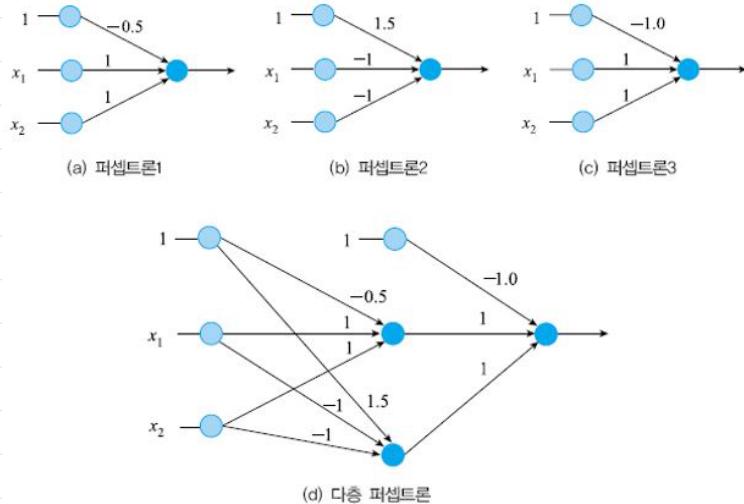
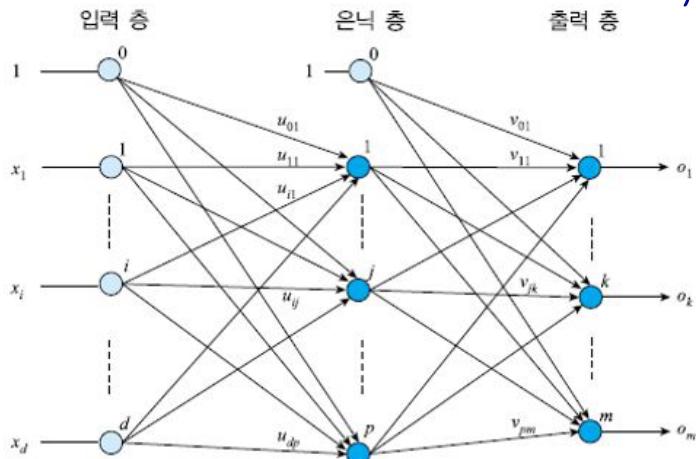


그림 4.8 세 개의 퍼셉트론과 이들을 연결하여 만든 다중 퍼셉트론

받은 값을 은닉층으로  
전달



$$O = (O_1, O_2, \dots, O_m)^T$$

\* 신경망은 일종의  
함수

$$O = f(\alpha)$$

$$\begin{aligned} Z &= P(\alpha) \\ \theta &= q(Z) \end{aligned}$$

or

$$O = q(P(\alpha))$$

연한파장: 받은 값을 그대로 전송

잔한파장: 가중치 합과 활성 함수 계산이라는 두 가지 연산 수행

이 연산을 수식적으로 표현하면,

$$\left( \begin{array}{l} \text{은닉층의 } j\text{번째 노드, } 1 \leq j \leq P : \\ z\_sum_j = \sum_{i=1}^d x_i u_{ij} + u_{oj} \\ z_j = T(z\_sum_j) \end{array} \right) \quad (4.12)$$

$$\left( \begin{array}{l} \text{출력층의 } k\text{번째 노드, } 1 \leq k \leq M : \\ o\_sum_k = \sum_{j=1}^P z_j v_{jk} + v_{ok} \\ o_k = T(o\_sum_k) \end{array} \right) \quad (4.13)$$

왼쪽에서 출발하여 앞으로만 전진하므로 전방계산  
(forward computation)  
이라 부름

(Input  $\rightarrow$  Output 으로)

- 다중 퍼셉트론이 사용하는 활성 함수  $T(\cdot)$

$\rightarrow$  Sigmoid Nonlinear 함수 사용

학습 과정에 많이 사용

이진 시그모이드 함수: (4.14)

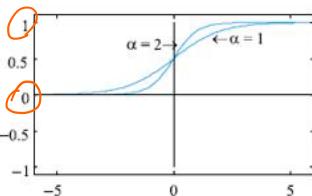
$$T_1(x) = \frac{1}{1 + e^{-\alpha x}}$$

양극 시그모이드 함수

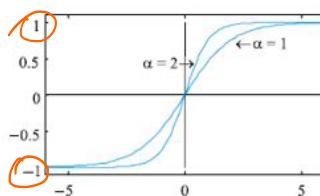
$$T_2(x) = \frac{2}{1 + e^{-\alpha x}} - 1 \quad (4.15)$$

$$T_1'(x) = \alpha T_1(x)(1 - T_1(x))$$

$$T_2'(x) = \frac{\alpha}{2}(1 + T_2(x))(1 - T_2(x))$$



(a) 이진 시그모이드



(b) 양극 시그모이드

그림 4.10 활성 함수로 널리 사용되는 두 가지 시그모이드 함수

- 전 구간에서 미분 가능

- 계산 간단, 미분값도  
간단

- 계단함수와 모양  
비슷

-  $\alpha$ 가 클수록 경사가  
큽니다

Proof,,

$$\frac{d}{dx} \text{sigmoid}(x) = \frac{d}{dx} (1 + e^{-x})^{-1}$$

$$= (-1) \frac{1}{(1 + e^{-x})^2} \cdot \frac{d}{dx} (1 + e^{-x})$$

$$= (-1) \frac{1}{(1 + e^{-x})^2} (0 + e^{-x}) \cdot \frac{d}{dx} (-x)$$

$$= (-1) \frac{1}{(1 + e^{-x})^2} (e^{-x})(-1)$$

$$= \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$= \frac{1+e^{-x} - 1}{(1+e^{-x})^2}$$

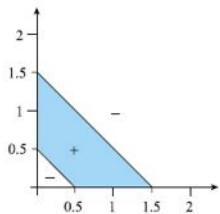
$$= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}} \left( 1 - \frac{1}{1+e^{-x}} \right)$$

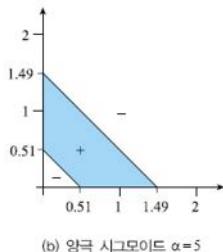
$$= \text{Sigmoid}(x) (1 - \text{Sigmoid}(x))$$

\* Cost function (비용함수)의 값이 최소가 되는 지점을 찾는 과정에서 활성화 함수는 미분되어야 한다.  $x=0$ 에서 연속이 아니므로 불가.

### ex 4.3) 다층 퍼셉트론의 공간분할



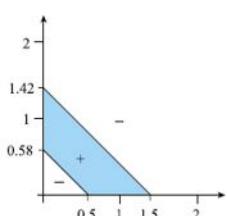
(a) 계단 함수 (양극 시그모이드  $\alpha=\infty$ )



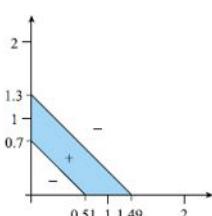
(b) 양극 시그모이드  $\alpha=5$

$\alpha$  값이 작을수록  $w_i$ 에 해당하는 영역이 줄어든다.

→ 두 퍼셉트론 모두 오차 클 확률이 줄어들어서



(c) 양극 시그모이드  $\alpha=3$



(d) 양극 시그모이드  $\alpha=2.5$

그림 4.11 활성 함수에 따른 디층 퍼셉트론의 공간 분할 능력

## FFMLP (feed-forward MLP)의 아키텍처

- 은닉층은 몇개로?
- 층간의 연결은 어떻게? (완전연결, 부분연결, 반대방향)
- 각 층의 노드는 몇개로?
- 어떤 활성함수 사용?

### 4.3.2 학습

MLP 학습이란? 훈련집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$  이 주어졌을 때 이들을 분류하는 다중 퍼셉트론(즉  $U$ 와  $V$ )을 찾아라.  $(x_i, t_i)$ 에서  $x_i$ 는 특징벡터이고  $t_i$ 는 부류표지 벡터(class label vector)로서  $x_i \in W_i$ 이면  $t_i = (0, \dots, 1, \dots, 0)^T$ 이다. 즉,  $j$ 번째 요소만 1이고 나머지 요소는 모두 0을 갖는다. 이는 이진모드를 사용할 때의 값이고, 만일 양극 모드를 사용한다면  $t_i = (-1, \dots, 1, \dots, -1)^T$ 로 하면 된다.

- 단계 1: 뿐만 아니라 구조 정의와 분류 과정의 수학식 정의. 이 식은 매개변수  
집합  $\Theta$ 로 정의.

다중 퍼셉트론의 매개 변수 집합은  $\Theta = \{U, V\}$   
 $U$ 는 입력층과 은닉층 사이에 있는 가중치이며,  
 $V$ 는 은닉층과 출력층 사이의 가중치이다.

## - 단계 2: 분류기 품질 측정용 비용함수 $J(\theta)$ 정의

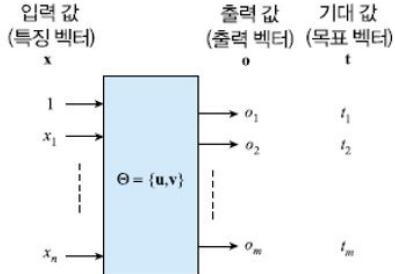


그림 4.12 다층 퍼셉트론의 입력, 출력, 그리고 기대값

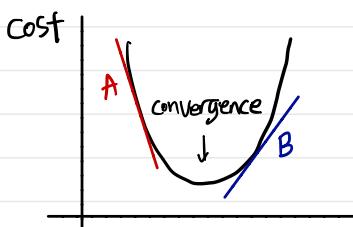
훈련 샘플  $X = (x_1, x_2, \dots, x_d)^T$  가 입력되었을 때 우리가 원하는 출력은  $t = (t_1, t_2, \dots, t_m)^T$  이다.  $\theta$ 로 정의되는 다층 퍼셉트론이  $o = (o_1, o_2, \dots, o_m)^T$  을 출력했다면

$$E = \frac{1}{2} \sum_{k=1}^m (t_k - o_k)^2 \quad (4.1b)$$

## - 단계 3: $J(\theta)$ 를 최적화하는 $\theta$ 를 찾는 알고리즘 설계

$$\left( \begin{array}{l} V(h+1) = V(h) + \Delta V = V(h) \ominus P \frac{\partial E}{\partial V} \\ u(h+1) = u(h) + \Delta u = u(h) \ominus Q \frac{\partial E}{\partial V} \end{array} \right) \quad (4.1n)$$

샘플 하나에 대한 학습 수행 : 패턴 모드 쓰는 이유



A의 위치의 기울기에서의 경사의 기울기는 음수이기 때문에 w의 값을 증가시키게 되고, B의 위치에서의 경사의 기울기는 양수이기 때문에 w의 값을 감소시키게 된다.

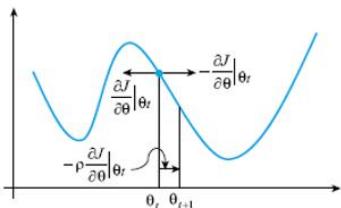


그림 11.9 내리막 경사법

입력: 훈련 집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ , 학습률  $\rho$

출력: 가중치  $u$ 와  $v$

알고리즘:

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

$j$ 번째 은닉 노드와  $k$ 번째 출력 노드 간의 가중치  
 $V_{jk}$ 를 위한 갱신값  $\Delta V_{jk}$ 의 유도

$$\left( \begin{array}{l} \text{총 출력 노드, } 1 \leq k \leq M : \\ o_{-sum_k} = \sum_{j=1}^p z_j V_{jk} + v_{ok} \\ o_k = T(o_{-sum_k}) \end{array} \right) \quad (4.13)$$

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

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

$$= -(t_k - o_k) \frac{\partial T(o_{-sum_k})}{\partial V_{jk}}$$

$$= -(t_k - o_k) T'(o_{-sum_k}) \frac{\partial o_{-sum_k}}{\partial V_{jk}}$$

$$= \underline{-(t_k - o_k) T'(o_{-sum_k}) z_j} s_k$$

$$\begin{cases} \delta_k = (t_k - o_k) T'(O\_sum_k), \quad 1 \leq k \leq m & (4.18) \\ \Delta V_{jk} = -P \frac{\partial E}{\partial V_{jk}} = P \delta_k z_j, \quad 0 \leq j \leq p, \quad 1 \leq k \leq m & (4.19) \end{cases}$$

i번째 입력 노드와 j번째 은닉 노드 간의  
가중치 생산값  $\Delta u_{ij}$ 의 유도

$$\left( \begin{array}{l} \text{은닉층의 } j\text{번째 노드, } 1 \leq j \leq p : \\ z\_sum_j = \sum_{i=1}^d z_i u_{ij} + u_{0j} \\ z_j = T(z\_sum_j) \end{array} \right) \quad (4.12)$$

$$\begin{aligned} \frac{\partial E}{\partial u_{ij}} &= \frac{\partial (0.5 \sum_{k=1}^m (t_k - o_k)^2)}{\partial u_{ij}} = - \sum_{k=1}^m (t_k - o_k) \frac{\partial o_k}{\partial u_{ij}} \\ &= - \sum_{k=1}^m (t_k - o_k) T'(O\_sum_k) \frac{\partial O\_sum_k}{\partial u_{ij}} \\ &= - \sum_{k=1}^m (t_k - o_k) T'(O\_sum_k) \frac{\partial O\_sum_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial u_{ij}} \\ &= - \sum_{k=1}^m (t_k - o_k) T'(O\_sum_k) v_{jk} \frac{\partial z_j}{\partial u_{ij}} \\ &= - \sum_{k=1}^m (t_k - o_k) T'(O\_sum_k) v_{jk} T'(z\_sum_j) x_i \\ &= - \sum_{k=1}^m \delta_k v_{jk} T'(z\_sum_j) x_i \end{aligned}$$

$$\hookrightarrow \eta_j = T'(z\_sum_j) \sum_{k=1}^m \delta_k v_{jk}, \quad 1 \leq j \leq p \quad (4.20)$$

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

$i=0$ 은 바이어스,  $x_0=1$

## (4.18) ~ (4.21) : Generalized delta rule 일반 델타 규칙

주어진 샘플에 대한 전방 계산으로 오류를 측정한 후 출력 중에서 시작하여 반대 방향으로 전진하며 이 오류를 전파한다. 여기서 오류 전파란 오류를 줄이는 쪽으로의 가중치 갱신을 뜻한다. 따라서 이 알고리즘을 오류 역전파 알고리즘(error back-propagation algorithm)이라 부른다.

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

입력: 훈련 집합  $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ , 학습률  $\rho$

출력: 가중치  $u$ 와  $v$

알고리즘:

// 초기화

1.  $u$ 와  $v$ 를 초기화한다.

2.  $x_0 = z_0 = 1$ ; // 바이어스

3. repeat {

4. for ( $X$ 의 샘플 각각에 대해) {

5. 현재 샘플을  $x = (x_1, x_2, \dots, x_d)^T$   $t = (t_1, t_2, \dots, t_m)^T$  으로 표기한다.

// 전방 계산

6. for ( $j = 1$  to  $p$ )  $\{z\_sum_j = \sum_{i=0}^d x_i u_j; z_j = \tau(z\_sum_j);\}$  // (4.12)

7. for ( $k = 1$  to  $m$ )  $\{o\_sum_k = \sum_{j=0}^p z_j v_{jk}; o_k = \tau(o\_sum_k);\}$  // (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  $v_{jk}, 0 \leq j \leq p, 1 \leq k \leq m$ 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  $u_y, 0 \leq i \leq d, 1 \leq j \leq p$ 에 대해)  $\Delta u_y = \rho \eta_j x_i;$  // (4.21)

// 가중치 갱신

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

13. for (모든  $u_y, 0 \leq i \leq d, 1 \leq j \leq p$ 에 대해)  $u_y = u_y + \Delta u_y;$  // (4.17)

14. }

15. } until (stop-condition);

16.  $u$ 와  $v$ 를 저장한다.

## 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.

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

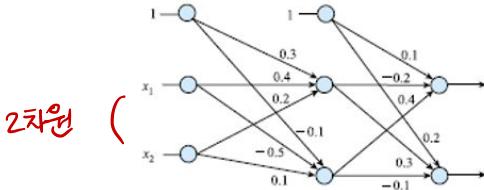


그림 4.13 다층 퍼셉트론 학습 과정의 예시

전방계산을 해보자.

2번 6.

$$Z_{\text{Sum}_1} = 1 \times 0.3 + 0.7 \times 0.4 + 0.2 \times 0.2 = 0.62000$$

$$Z_{\text{Sum}_2} = 1 \times (-0.1) + 0.7 \times (-0.5) + 0.2 \times (0.1) = -0.43000$$

$$Z_1 = T(0.62000) = 2 / (1 + e^{-0.62000}) = 0.30044$$

$$Z_2 = T(-0.43000) = 2 / (1 + e^{-0.43000}) = -0.21175$$

2번 7.

$$D_{\text{Sum}_1} = 1 \times 0.1 + 0.30044 \times (-0.2) + (-0.21175) \times 0.4 = -0.04479$$

$$D_{\text{Sum}_2} = 1 \times 0.2 + 0.30044 \times (0.3) + (-0.21175) \times (-0.1) = 0.31131$$

$$O_1 = T(-0.04479) = -0.02239$$

$$O_2 = T(0.31131) = 0.15441$$

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

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

이제 오류 역전파 단계를 계산해보자.

21일 8.

$$\delta_1 = (-1.0 + 0.02239) T'(-0.04419) \\ = -0.911161 \times 0.5 \times (1 + T(-0.04419))(1 - T(-0.04419)) \\ = -0.48856$$

$$\delta_2 = (1.0 - 0.15441) T'(0.31131) \\ = 0.84559 \times 0.5 \times (1 + T(0.31131))(1 - T(0.31131)) \\ = 0.41211$$

21일 9.

원년층의 8번째 노드 가중치

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

$$\Delta v_{02} = 0.2 \times (0.41211) \times 1.0 = 0.08254$$

$$\Delta_{11} = 0.2 \times (-0.48856) \times 0.30044 = -0.02936$$

$$\Delta_{12} = 0.2 \times (0.41211) \times 0.30044 = 0.02480$$

$$\Delta_{21} = 0.2 \times (-0.48856) \times (-0.21115) = 0.02069$$

$$\Delta_{22} = 0.2 \times (0.41211) \times (-0.21115) = -0.01148$$

2인 10.

$$\eta_1 = T'(0.6200) \times ((-0.48856) \times (-0.2) + (0.41271) \times (0.3)) \\ = 0.10016$$

$$\eta_2 = T'(-0.4300) \times ((-0.48856) \times (0.4) + (0.41271) \times (-0.1)) \\ = -0.11304$$

2인 11.

$$\Delta U_{01} = (0.2) \times (0.10016) \times 1.0 = 0.02015$$

$$\Delta U_{02} = (0.2) \times (-0.11304) \times 1.0 = -0.02261$$

$$\Delta U_{11} = (0.2) \times (0.10016) \times 0.7 = 0.01411$$

$$\Delta U_{12} = (0.2) \times (-0.11304) \times 0.7 = -0.01583$$

$$\Delta U_{21} = (0.2) \times (0.10016) \times 0.2 = 0.00403$$

$$\Delta U_{22} = (0.2) \times (-0.11304) \times 0.2 = -0.00452$$

이제) 가중치 갱신 단계를 수행해 보자.

2인 12.

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

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

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

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

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

$$V_{22} = -0.1 - 0.01148 = -0.11148$$

2인 13.  $U_{01} = 0.3 + 0.02015 = 0.32015$

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

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

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

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

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

이제 각인 4로 돌아가  $X$ 에 있는 다른 샘플을 가지고 같은 과정을 반복하면 된다.

이 예제를 마치기 전 학습한 효과를 확인해 보자.

각인 6과 7.

$$Z_{\text{sum}_1} = 1.0 \times 0.32015 + 0.7 \times 0.41411 + 0.2 \times 0.20403 \\ = 0.65083$$

$$Z_{\text{sum}_2} = 1.0 \times (-0.12261) + 0.7 \times (-0.51583) + 0.2 \times 0.09548 \\ = -0.46460$$

$$Z_1 = T(0.65083) = 0.31440$$

$$Z_2 = T(-0.46460) = -0.22821$$

$$O_{\text{sum}_1} = 1.0 \times 0.00229 + 0.31440 \times (-0.022936) + \\ (-0.22821) \times 0.42069 = -0.16582$$

$$O_{\text{sum}_2} = 1.0 \times 0.28254 + 0.31440 \times (0.32480) + \\ (-0.22821) \times (-0.11148) = 0.41147$$

$$O_1 = T(-0.16582) = -0.08272$$

$$O_2 = T(0.41147) = 0.20288$$

$O = (-0.08272, 0.20288)^T$  를 얻어, 우리가 원하는  $t = (-1, 1)^T$ 에 가까워졌음을 알 수 있다. 오류도  $E = 0.13840$  이 되어 이전보다 줄었음을 알 수 있다.

- 오류 역전파 알고리즘의 계산 복잡도

$$\Theta((d+m)PHN)$$

비깥쪽 repeat 루프의 반복 횟수:  $H$  (epoch 수에 해당)

안쪽 for 루프: 샘플의 수  $N$ 번 만큼

그 안쪽의 이중 루프: 가중치의 개수에 비례

$$(d+1)p + (p+1)m$$

⇒  $H$ 는 보통 수십~수천,  $N$ 은 수천~수백이 필요하므로  
다층 퍼셉트론의 학습은 많은 시간을 필요로 한다.

### 4.3.3 인식

다층 퍼셉트론을 미지의 패턴을 인식하는데 사용해보자. 인식은 전방계산  
한번으로 끝난다. 전방계산을 하여 얻은 출력 벡터  $\mathbf{o} = (o_1, o_2, \dots, o_m)^T$   
을 조사하여 가장 큰 값을 갖는 요소의 인덱스를 인식 결과로  
결정한다.

#### 알고리즘 [4.6] 다층 퍼셉트론 (MLP)에 의한 인식

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

출력: 부류  $\omega_q$

알고리즘:

1.  $\mathbf{u}$ 와  $\mathbf{v}$ 를 읽어 MLP를 설정한다.

2.  $x_0 = z_0 = 1$ ; // 바이어스

3. for ( $j = 1$  to  $p$ ) {  $z\_sum_j = \sum_{i=0}^d x_i u_{ij}$ ;  $z_j = \tau(z\_sum_j)$ ; } // 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.  $\mathbf{x}$ 를  $q = \arg \max_j o_j$  인  $\omega_q$ 로 분류한다. // 가장 큰 값을 갖는 부류

시간복잡도:  $\Theta((d+m)P)$

→  $N$ 에 무관,

매우 빠름

수식으로 쓰면,  
 $x$ 를  $w_q$ 로 분류,  
이때,  $q = \arg \max_j o_j, 1 \leq j \leq m$

 $) \quad (4.22)$ 

#### 4.3.4 구현과 몇 가지 부연 설명

##### 1. 아키텍처

① 같은 성능을 갖는다면 가급적 단순한 것을 선택하라.

→ 가장 보편적으로 사용하고 있는 아키텍처는 은닉 층이 하나인 전방 다층 퍼셉트론 (FFMLP)이고, 모든 노드가 같은 활성 함수를 사용한다. 하나의 은닉 층만 있어도 주어진 훈련 집합의 입력과 출력 관계를 충분한 정확도로 근사할 수 있다는 이론적인 근거가 있다.

② 은닉 노드의 개수는 매우 중요하다.

신경망의 크기 = 가중치의 개수  $(d+1)p + (p+1)m$

$X \geq 1000$ 개의 샘플을 가졌다면 1000개의 샘플로 3340개의 매개 변수를 추정하는 문제가 된다. 매개 변수에 비해 턱 없이 정보가 부족하다. 반면에,  $p$ 를 너무 적게 하면 신경망의 용량이 작아 필요한 정보를 충분히 담을 수 없게 된다. 따라서 적절한 값으로 설정하는 것이 매우 중요하나, 일반적인 경우에 대해 초적의  $p$ 값을 추정하는 방법은 없다. 보통 여러 값에 대해 신경망을 훈련하고 그들 중에 가장 좋은 성능을 보이는 것을 택한다.

## 2. 가중치 초기화 (라인 1)

학습이 어떤 초기값을 가지고 출발?

- ① 전역 최저점으로 수렴할지 or
- 지역 최저점으로 수렴할지와 관련

→  $\theta_3, \theta_5$ 에서 출발하면 최적 점으로 수렴하나,  
 $\theta_2$ 에서 출발하면 지역 최적점에 빠짐

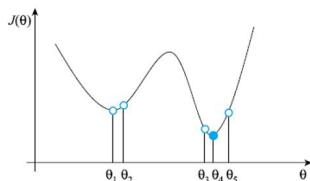


그림 11.7 최소화 문제에서 최적 해 (속이 찬 점)와 부 최적 해 (속이 빈 점)

### ② 수렴 속도에 영향

→  $\theta_5$ 보다  $\theta_3$ 에서 출발하는 것이 더 빠른 수렴 보장

보편적으로 작은 반수로 설정한다. (-0.5 ~ 0.5)

일반적인 경우에 대해 전역 최저 점을 보장하며 가장 빠르게 수렴하는 초기값을 찾는 방법은 알려져 있지 않음

## 3. 언제 종료할 것인가?

수렴 (Convergence) 여부는 어떻게 판단?

- ① 세대 수를 1부터 시작하여 N에 도달하면 멈춘다.

→ N을 얼마나 설정할 것인가? 보통은 여러번 실험을 통해 적절한 값을 알아냄

- ② 한 세대가 끝난 후 (4.23)으로 정의되는 평균 제곱 오차 (Mean squared error, MSE)를 구하여 MSE가 어떤 임계값보다 작으면 멈춤

$$MSE = (\sum_{i=1}^N E_i^2) / N \quad (4.23)$$

→ 이 방법에서도 임계값 어떻게 설정할지의 문제에 봉착.  
보통은 여러 번 실험을 통해 적절한 임계값을 알아냄

- ③ MSE 값의 변화 추이를 관찰해 여러 세대에 걸쳐  
변화가 아주 작을 때 멈춘다
- 이 때에도 임계값 설정 필요

- ④ 훈련 집합  $X$  와 별도로 검증 집합  $X'$  가 주어졌다면,  
세대가 끝날 때마다  $X'$  를 알고리즘 [4.6] 으로 인식.

→ 보통 초기에는 오류율이 급격하게 감소하다 어느 정도 지나면  
속도가 떨어진다. 그리고 어느 순간에  
오류율이 증가하는데 이 때 멈춘다.  
그리고 바로 증가하기 전의  $N$  와  $V$  를  
답으로 취한다. (검증 집합이 따로  
없는 상황: 고차 검증 방법)

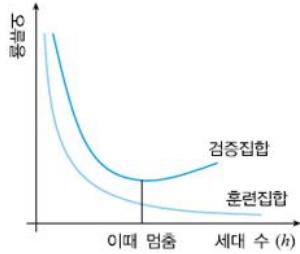


그림 4.15 일반화 기준에 따른 멈춤 조건

#### 4. 목적 벡터의 표현과 활성 함수

이전모드  $t = (0, \dots, 1, \dots, 0)^T$   
양극모드  $t = (-1, \dots, 1, \dots, -1)^T$

- 매개변수 설정
- 일반적인 경우에 적용되는 보편적인 규칙은 X
- 경험과 실험을 통해 설정
- 신경망 성능이 매개변수에 아주 민감하지는 않기에 어느 정도의 실험과 경험을 통해 설정 가능