다층 퍼셉트론
-Multi-Layer Perceptron (MLP)-

20201484 김성훈

AI·빅데이터학과

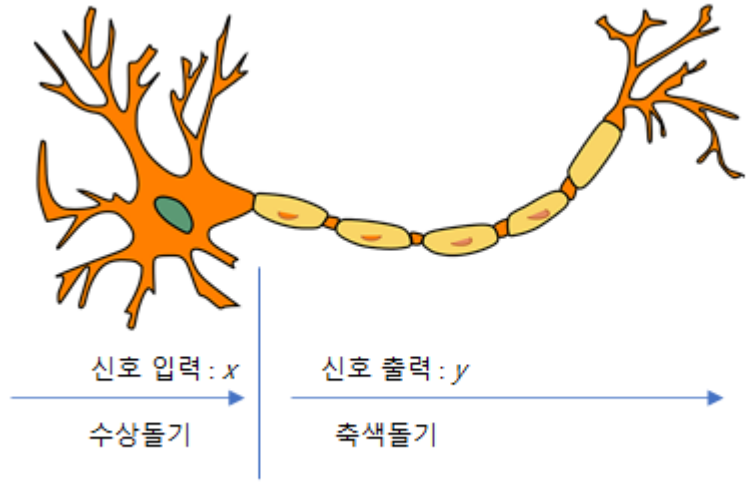
rlatjdgns0816@gmail.com

목차

Chapter3.1 신경망 기초

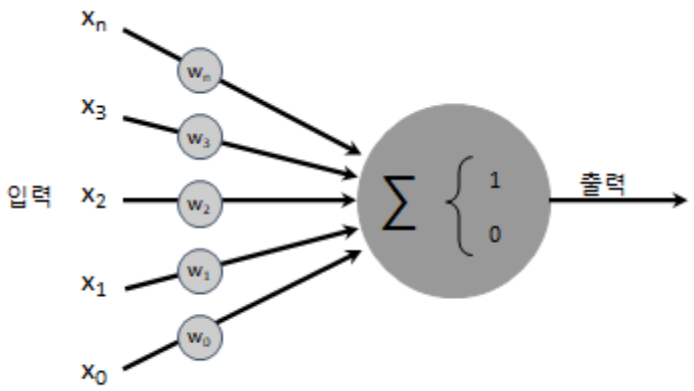
Chapter3.1 - 신경망 기초

- **인공신경망과 생물신경망**
  - 인공신경망: 컴퓨터가 실수를 통해 배우고 지속적으로 개선하는데 사용하는 적응형 시스템을 생성



신경세포(Neuron)

	생물신경망	인공신경망
입력(x)	수상돌기	입력층
출력(y)	축색돌기	출력층



인공뉴런(Artificial Neuron)

https://deepcell.kr/bbs/board.php?bo_table=dl&wr_id=11

Chapter3.1 - 신경망 기초

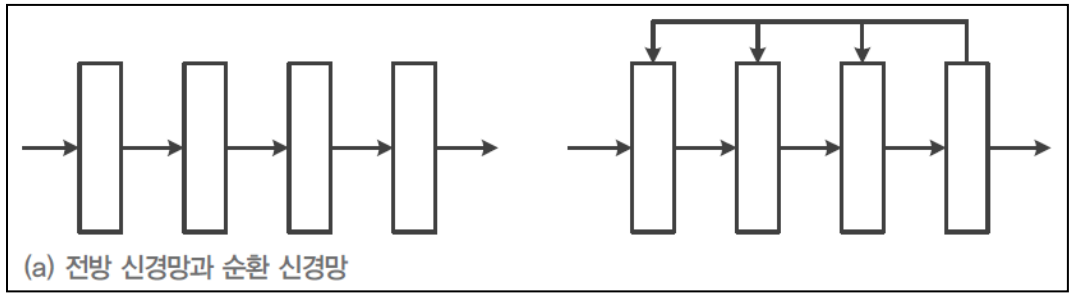
- **신경망의 역사**
  - 1943년 매컬럭(McCulloch)과 피츠(Pitts)의 최초의 신경망
  - 1949년 – 헤브(Hebb)가 최초로 학습 알고리즘 발표
  - 1958년 – 로젠블랫(Rosenblatt)이 퍼셉트론 제안
  - 위드로(Widrow)와 호프(Hoff)는 Adaline과 Madaline을 발표
  - 1960년 – 과대평가로 학계와 매스컴의 주목
  - 1969년 – 민스키(Minsky)와 페퍼트(Papert)의 퍼셉트론 한계를 수학적으로 입증 →『Perceptrons』
  - 신경망 연구 퇴조
  - 1986년 – 루멜하트(Rumelhart)는 다층 퍼셉트론과 역전파 알고리즘 제시 →『Parallel Distributed Processing』
  - 신경망 연구 부활
  - 1990년 SVM이 신경망보다 더 좋은 성능을 가짐
  - 현재 – 딥러닝이 실현되어 기계 학습의 주류로 자리 매김

Chapter3.1 - 신경망 기초

- **신경망의 종류**
  - 전방(feedforward) 신경망과 순환(recurrent) 신경망
    - 전방 신경망: 모든 계산이 왼쪽에서 오른쪽으로 진행
    - 순환 신경망: 오른쪽에서 왼쪽으로 진행하는 피드백 계산도 포함

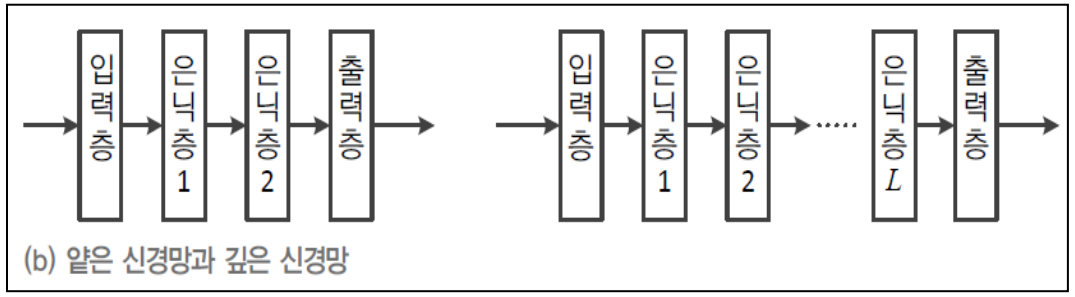
ex. DMLP(깊은 다층 퍼셉트론), CNN(컨볼루션 신경망))

ex. RNN, LSTM



(a) 전방 신경망과 순환 신경망

  - 얕은 신경망과 깊은 신경망

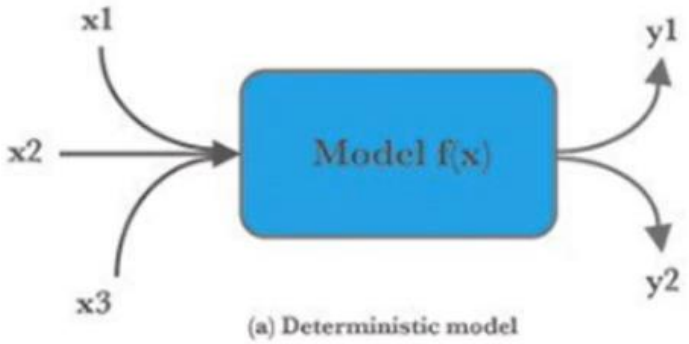


(b) 얕은 신경망과 깊은 신경망

※ 몇 개의 은닉층을 기준으로 구분하는지 명확 X

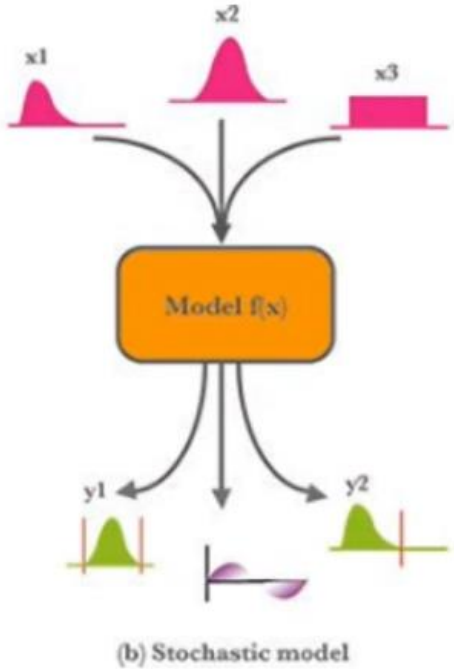
Chapter3.1 - 신경망 기초

- **신경망의 종류**
  - 결정론(deterministic) 신경망
    - 매개변수(파라미터, 계수)가 고정으로 입력에 의한 출력 값이 고정
      - Ex) 회귀, 분류



(a) Deterministic model

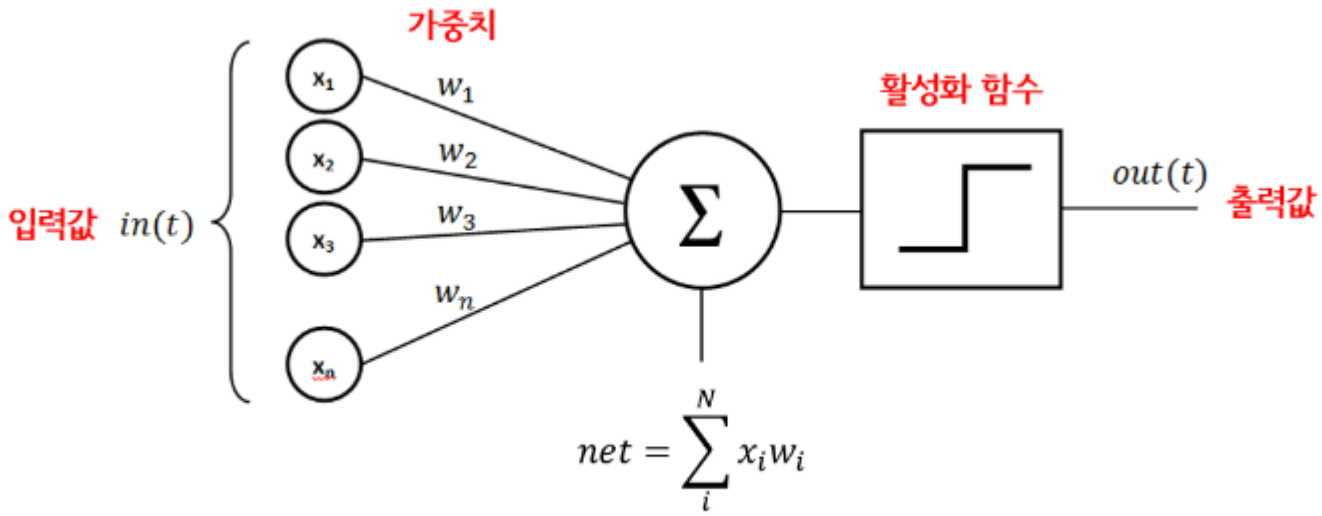
- 확률론적(stochastic) 신경망
  - 확률에 따른 난수를 사용하므로 입력이 같아도 다른 출력 값
    - Ex) RBM, DBM



(b) Stochastic model

Chapter3.2 퍼셉트론

Chapter3.2 - 퍼셉트론



퍼셉트론의 동작 과정 (출처: 구글 무료 이미지 검색 + 가공)

초기 형태의 인공 신경망으로

다수의 입력으로부터 하나의 결과를 내보내는 알고리즘입니다.

Chapter3.2 - 퍼셉트론

- **구조**
  - 입력층
    - 입력 노드 하나가 특징 벡터의 특징 하나에 해당
    - 입력층의 첫번째 노드는 바이어스(bias) → x_0
    - 연산을 하지 않음 → 층 개수에서 제외
  - 출력층
    - Y라 표기된 하나의 노드

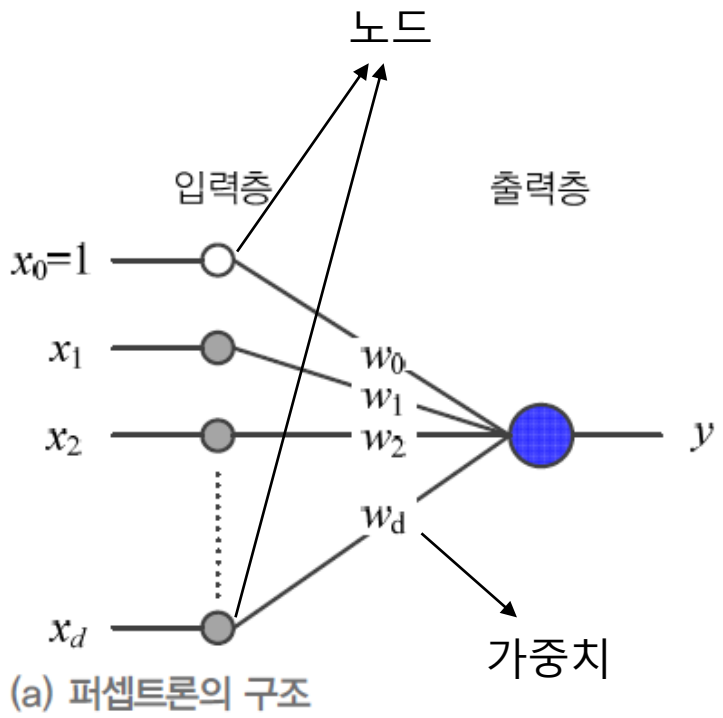


그림 3-3 퍼셉트론의 구조와 동작

* 에지 : 입력 노드와 출력 노드를 연결하는 선 → 가중치의 값을 가짐

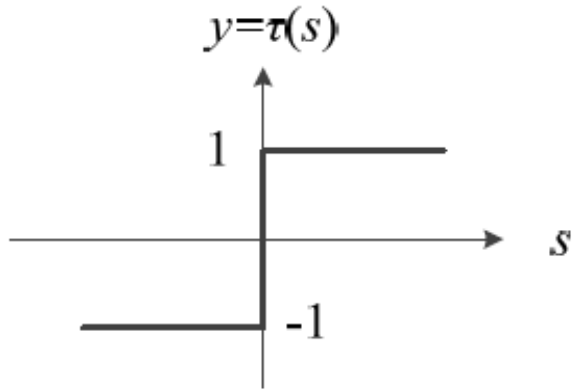
Chapter3.2 - 퍼셉트론

- **동작**

  1) 초기화
     - 가중치를 무작위의 작은 값으로 할당
  2) 입력과 가중치의 선형 조합
  3) 활성화 함수 적용
  4) 예측 및 오차 계산
  5) 가중치 업데이트
  6) 2단계 ~ 5단계 반복
  7) 종료 조건

$$y = \tau(s)$$

$$\text{이때 } s = w_0 + \sum_{i=1}^{d} w_i x_i, \qquad \tau(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases} \qquad (3.1)$$



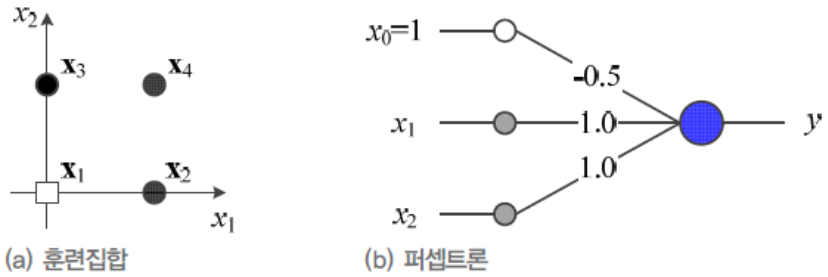
(b) 계단함수를 활성함수 $\tau(s)$로 이용함

Chapter3.2 - 퍼셉트론

예제 3-1 **퍼셉트론의 동작**

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.

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



(a) 훈련집합 (b) 퍼셉트론

그림 3-4 OR 논리 게이트를 이용한 퍼셉트론의 동작 예시

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

$$s = x_0 w_0 + x_1 w_1 + x_2 w_2 \qquad \tau(s) = 1 \text{ or } -1$$

샘플 4개를 하나씩 입력하여 제대로 분류하는지 확인해 보자.

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

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

Chapter3.2 - 퍼셉트론

- **동작 (행렬표기)**

$$s = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0, \qquad \text{여기서 } \mathbf{x} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}},\ \mathbf{w} = (w_1, w_2, \cdots, w_d)^{\mathrm{T}} \tag{3.2}$$

$$s = \mathbf{w}^{\mathrm{T}}\mathbf{x}, \qquad \text{여기서 } \mathbf{x} = (1, x_1, x_2, \cdots, x_d)^{\mathrm{T}},\ \mathbf{w} = (w_0, w_1, w_2, \cdots, w_d)^{\mathrm{T}} \tag{3.3}$$

$$y = \tau(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \tag{3.4}$$

Chapter3.2 - 퍼셉트론

- **분류기로 해석**

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

  - $w_1 \sim w_d$: 직선, 초평면의 방향
  - w_0은 절편을 의미

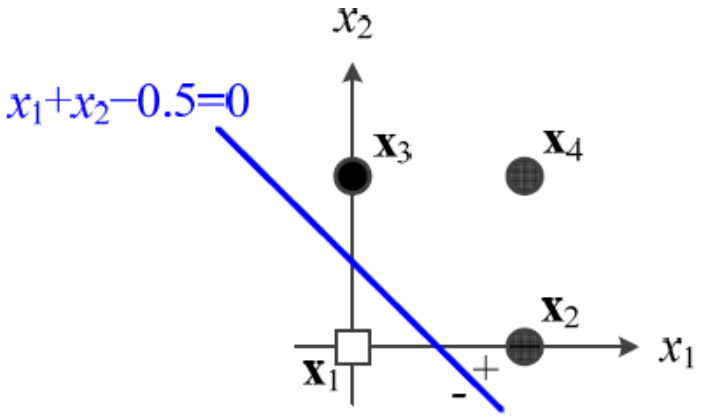
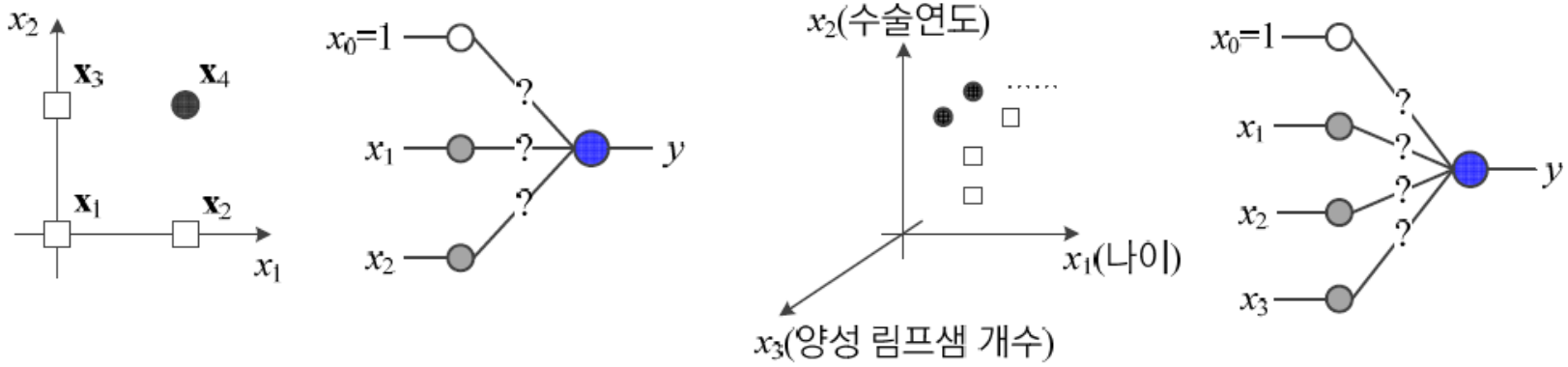


그림 3-5 [그림 3-4(b)]의 퍼셉트론에 해당하는 결정 직선

- **선형 분리 가능**
  - 데이터를 직선, 평면 또는 초평면 등으로 나누어 각 클래스에 대한 구분이 가능한 경우

Chapter3.2 - 퍼셉트론

- 학습



(a) AND 분류 문제 (b) Haberman survival 분류 문제

그림 3-6 어떻게 학습시킬 것인가?

차원, 샘플⇧ → 사람이 매개변숫값(가중치)을 알아내는 일이 매우 어렵다

Chapter3.2 - 퍼셉트론

- **학습**
  - 목적함수
    - 모델의 출력과 실제 값 간의 오차를 측정하는 함수
    - 조건 1) $J(w) \geq 0$
    - 조건 2) w가 최적이면, 즉 모든 샘플을 맞추면 $J(w) = 0$
    - 조건 3) 틀리는 샘플이 많은 w일수록 $J(w)$는 큰 값

$$J(\mathbf{w}) = \sum_{\mathbf{x}_k \in Y} -y_k \left(\mathbf{w}^{\mathrm{T}} \mathbf{x}_k \right) \qquad (3.7)$$

$w^T x_k$: 퍼셉트론이 계산한 값

Y : w가 틀린 샘플의 집합

y_k: 잘못 예측한 값 → (-) 부호 붙이는 이유

Chapter3.2 - 퍼셉트론

- **학습**
  - 가중치 갱신 규칙 : $\Theta = \Theta - \rho g$
  - $-\frac{dJ(w)}{d\theta}$ 방향에 목적함수의 최저점이 존재

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

⇨ w_i에 대한 g (Gradient)

$$\frac{\partial J(\mathbf{w})}{\partial w_i} = \sum_{\mathbf{x}_k \in Y} -y_k x_{ki}, \qquad i = 0,1,\cdots,d \tag{3.8}$$

델타 규칙: $$w_i = w_i + \rho \sum_{\mathbf{x}_k \in Y} y_k x_{ki}, \qquad i = 0,1,\cdots,d \tag{3.9}$$

Chapter3.2 - 퍼셉트론

- **퍼셉트론 학습(배치 버전)**
  - 모든 샘플의 그레이디언트를 구한 후 델타 규칙을 통해 한꺼번에 갱신

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 w를 설정한다.
2  repeat
3     Y = ∅    // 틀린 샘플 집합
4     for j=1 to n
5        y = τ(w^T x_j)                  // 식 (3.4)
6        if(y ≠ y_j) Y = Y ∪ x_j         // 틀린 샘플을 집합에 추가한다.
7     if(Y ≠ ∅)
8        for i=0 to d                    // 식 (3.9)
9           w_i = w_i + ρ Σ_{x_k∈Y} y_k x_ki
10 until (Y = ∅)
11 ŵ = w
```

→ y는 퍼셉트론이 예측한 값
y_j는 j번째 샘플의 정답

Chapter3.2 - 퍼셉트론

- **퍼셉트론 학습(스토캐스틱 버전)**
  - 하나의 샘플에 대해서 그레이디언트를 계산 후 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 w을 설정한다.
2  repeat
3     𝕏의 샘플 순서를 섞는다.
4     quit=true
5     for j=1 to n
6        y = τ(w^T x_j)     // 식 (3.4)
7        if(y ≠ y_j)
8           quit=false
9           for i=0 to d
10             w_i = w_i + ρ y_j x_ji
11 until(quit)     // 틀린 샘플이 없을 때까지
12 ŵ = w
```

Chapter3.2 - 퍼셉트론

- **학습**
  - 행렬 표기
    - 델타 규칙: $w = w + \rho \sum_{X_k \in Y} y_k x_k$
    - [알고리즘 3-1] 행렬 표기로 수정

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

    - [알고리즘 3-2] 행렬 표기로 수정

$$\left.\begin{array}{l} 9.\ \text{for } i = 0 \text{ to } d \\ 10.\quad w_i = w_i + \rho y_j x_{ji} \end{array}\right\} \rightarrow\ 9.\ \mathbf{w} = \mathbf{w} + \rho y_j \mathbf{x}_j$$

Chapter3.3 다층 퍼셉트론

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론**

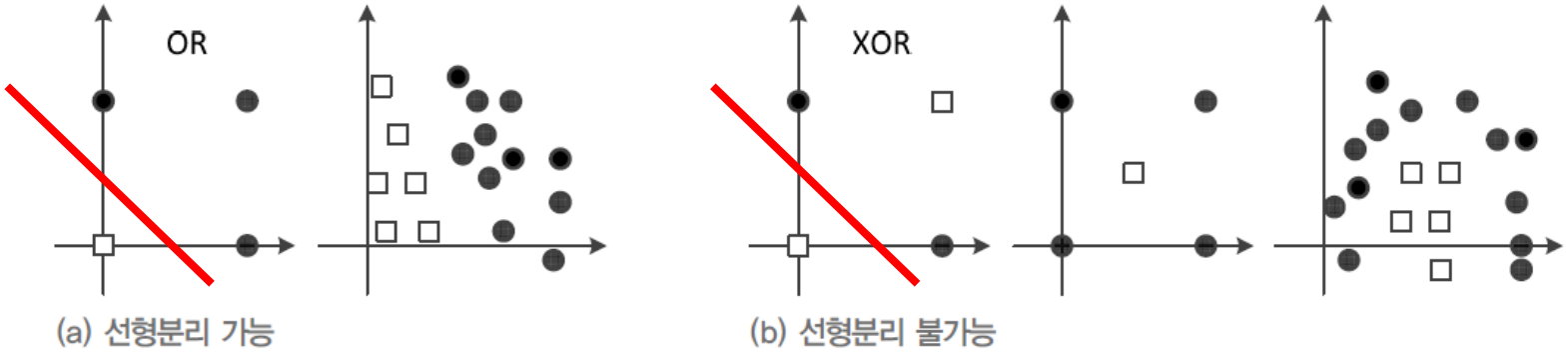


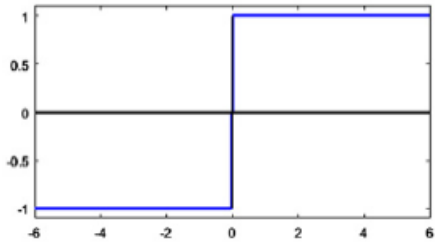
그림 3–7 선형분리가 가능한 상황과 불가능한 상황

- (a)는 선형분리 가능 → 100% 정확률 가능

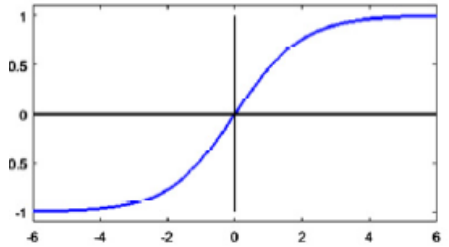
- (b)는 선형분리 불가능 → 75% 정확률이라는 한계

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론**
  - 새로 도입한 기법
    - 은닉층 추가 → 분류하기 유리한 새로운 특징 공간으로 변환
    - 새로운 활성함수 도입 → 퍼셉트론: 계단함수(결성 의사결정)
      다층 퍼셉트론: 시그모이드 함수(연성 의사결정)
      → 더 융통성 있는 의사결정 가능
      - Ex) 고양이 이미지가 주어질 시
        - 퍼셉트론 출력 : 고양이
        - 다층 퍼셉트론 출력 : 0.8 (고양이일 가능성이 80%라는 것을 의미)
    - 오류 역전파 알고리즘을 사용



(a) 계단 함수

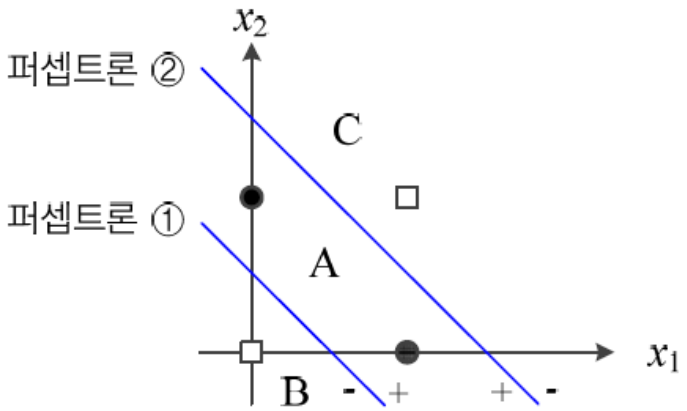


(c) 하이퍼볼릭 탄젠트 시그모이드

Chapter3.3 – 다층 퍼셉트론

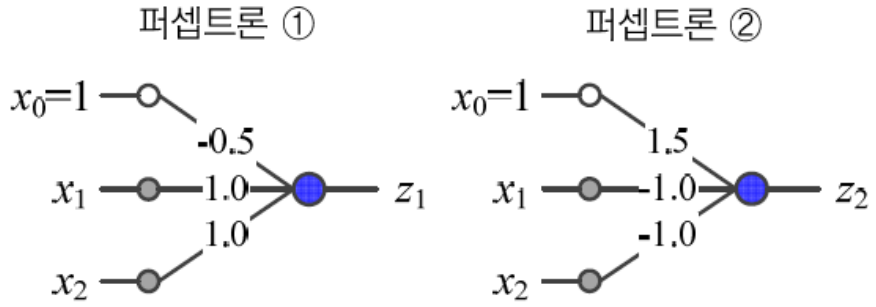
- **특징 공간 변환**
  - 퍼셉트론①과 퍼셉트론②가 모두 +1이면 이면 ● 부류이고 -1이면 □ 부류

분류기



(a) 퍼셉트론 2개를 이용한 공간분할

구조

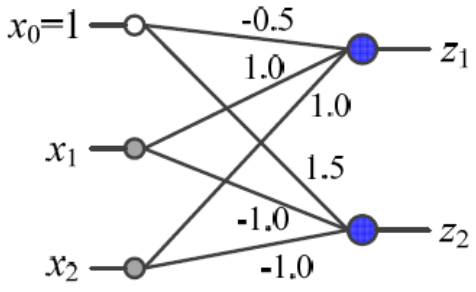


(b) 퍼셉트론 2개

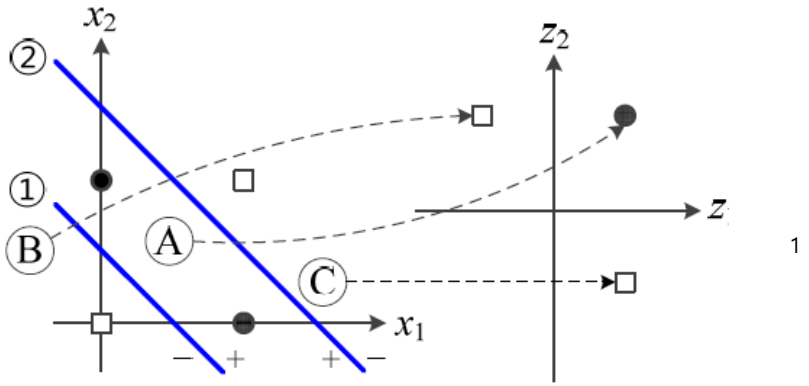
그림 3-8 XOR 문제의 해결

Chapter3.3 – 다층 퍼셉트론

- **특징 공간 변환**
  - 원래 공간 $\mathrm{z} = (x_1, x_2)^T$를 새로운 특징 공간 $(z_1, z_2)^T$로 변환 → 저급 특징에서 고급 특징을 추출
  - 새로운 특징 공간 z에서는 선형 분리 가능



(a) 두 퍼셉트론을 병렬로 결합

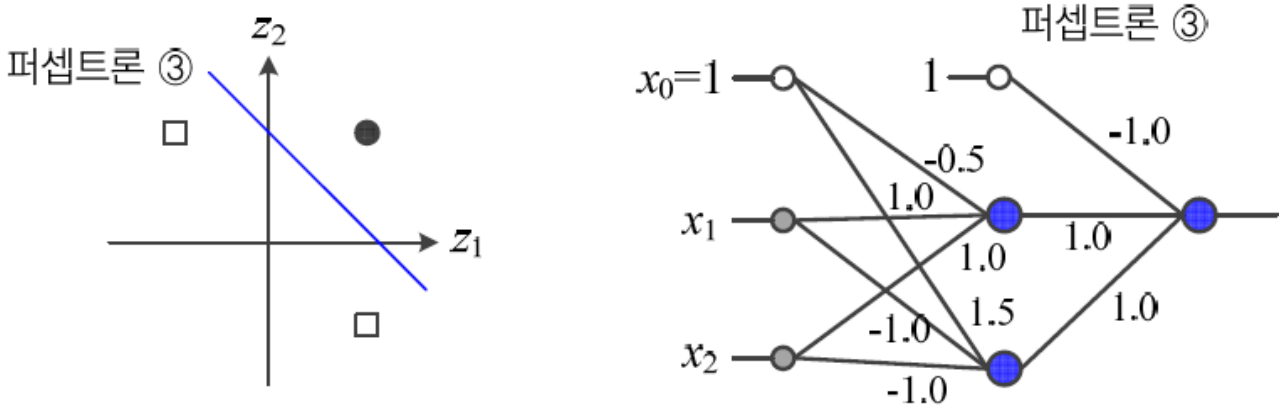


(b) 원래 특징 공간 x를 새로운 특징 공간 z로 변환

그림 3-9 특징 공간의 변환

Chapter3.3 – 다층 퍼셉트론

- **특징 공간 변환**
  - 새로운 특징 공간 $(z_1, z_2)^T$에 대해 2개의 영역으로 나누는 세번째 퍼셉트론③

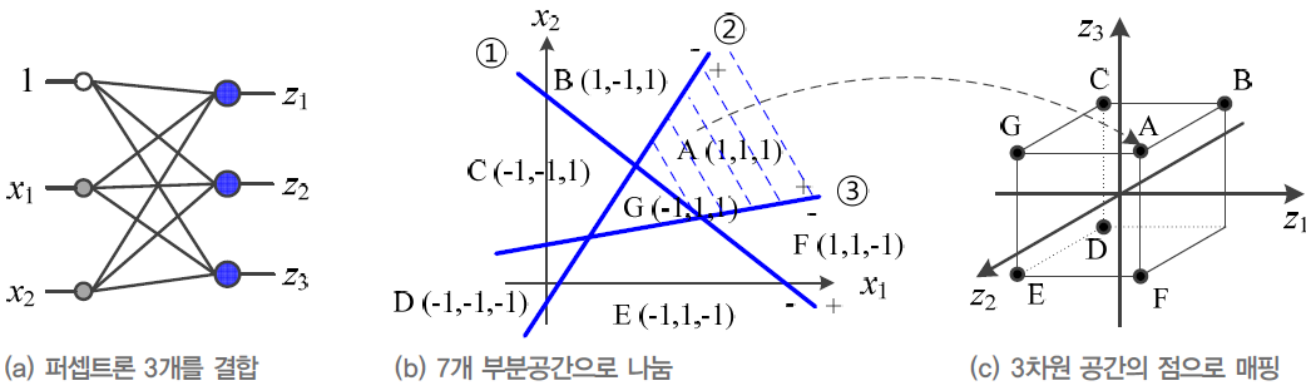


(a) 새로운 특징 공간에서 분할 (b) 퍼셉트론 3개를 결합한 다층 퍼셉트론

그림 3-10 다층 퍼셉트론

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론 용량**
  - 3개의 퍼셉트론을 결합하여 7개의 영역으로 나누고 한 영역을 3차원 점으로 변환
  - 활성함수인 계단함수를 사용하여 영역을 점으로 매핑



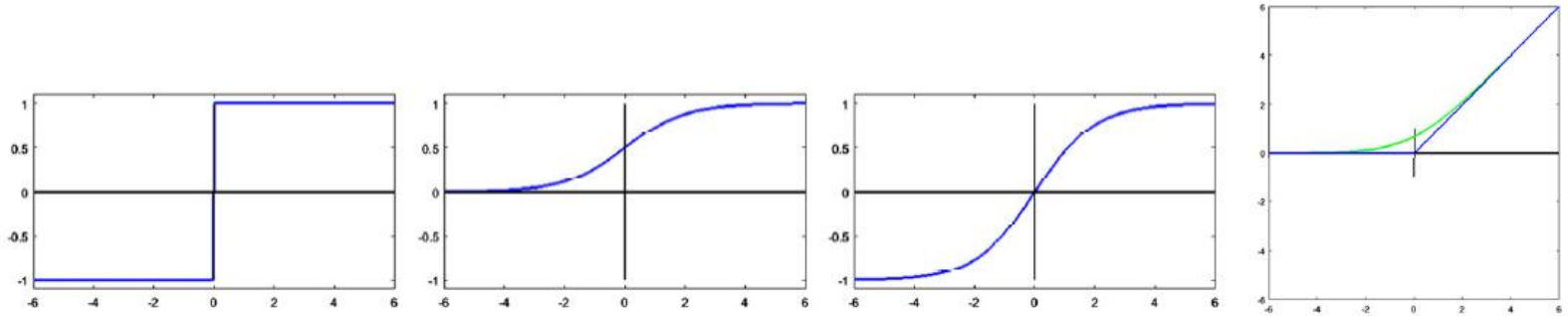
(a) 퍼셉트론 3개를 결합 (b) 7개 부분공간으로 나눔 (c) 3차원 공간의 점으로 매핑

그림 3-11 퍼셉트론을 3개 결합했을 때 공간 변환

  - p개의 퍼셉트론을 결합하면 p차원 공간으로 변환
    - $1 + \sum_{i=1}^{p} i$ → 영역의 개수

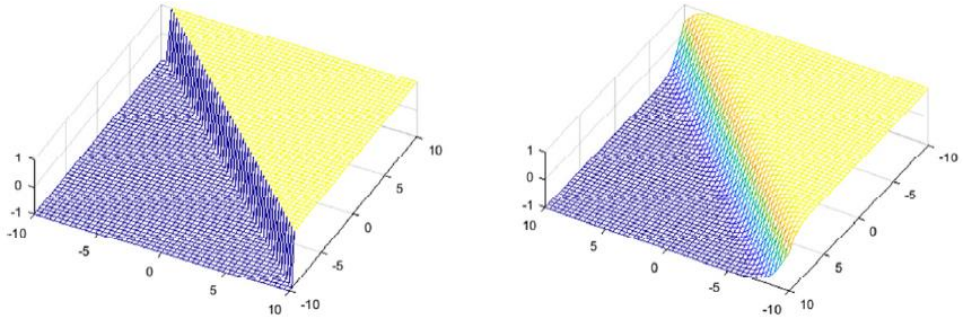
Chapter3.3 – 다층 퍼셉트론

- **활성함수**
  - 경성 의사결정 – 계단형 활성함수
  - 연성 의사결정 – 로지스틱 시그모이드, 하이퍼볼릭 탄젠트 시그모이드, softplus와 rectifier



(a) 계단 함수 (b) 로지스틱 시그모이드 (c) 하이퍼볼릭 탄젠트 시그모이드 (d) softplus와 rectifier

그림 3-12 신경망이 사용하는 활성함수



(a) 계단함수의 딱딱한 공간 분할 (b) 로지스틱 시그모이드의 부드러운 공간 분할

그림 3-13 퍼셉트론의 공간 분할 유형

Chapter3.3 – 다층 퍼셉트론

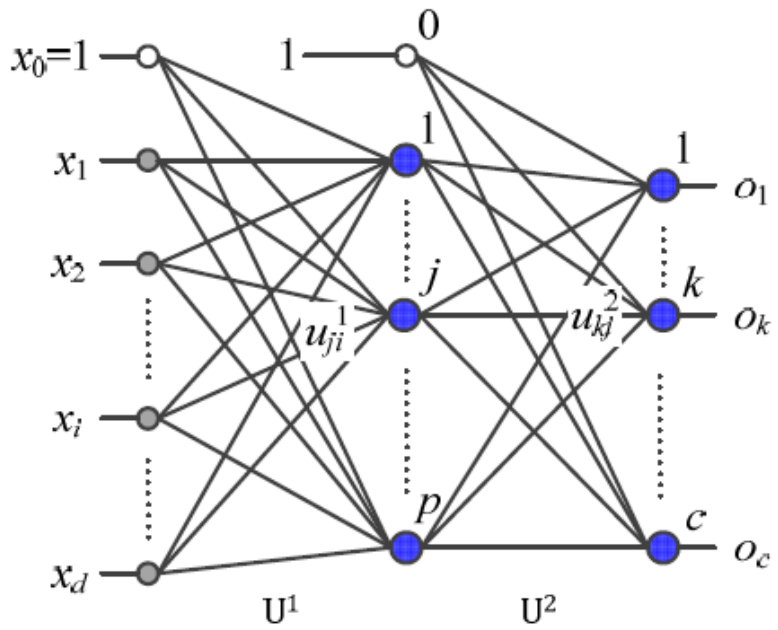
- **활성함수**
  - a는 기울기를 의미 (커질수록 계단함수가 됨)
  - 함수 값과 1차 도함수 값을 계산하는 속도 → 전체 학습 시간을 좌우

표 3-1 활성함수로 사용되는 여러 함수

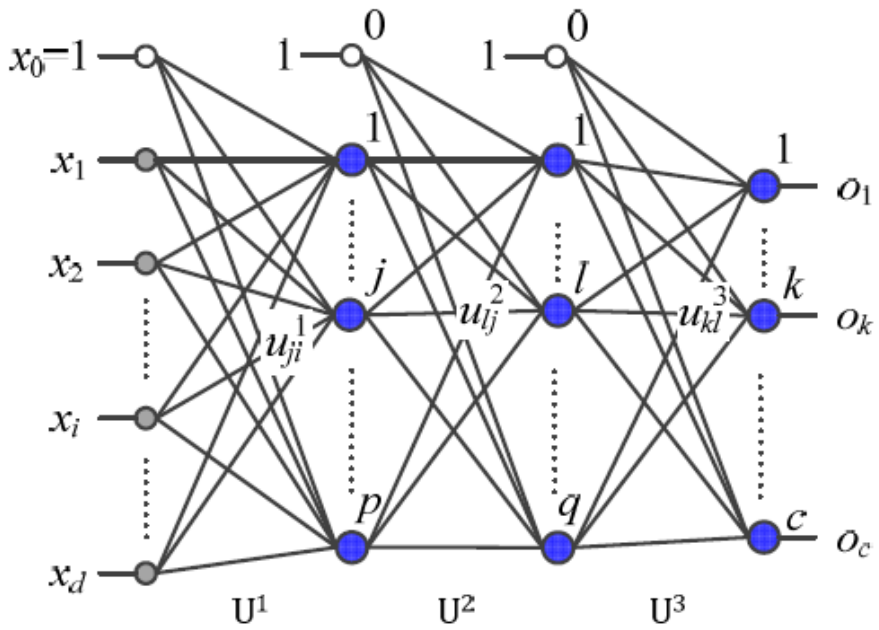
함수 이름	함수	1차 도함수	범위
계단	$\tau(s)=\begin{cases}1 & s\geq 0\\ -1 & s<0\end{cases}$	$\tau'(s)=\begin{cases}0 & s\neq 0\\ \text{불가} & s=0\end{cases}$	-1과 1
로지스틱 시그모이드	$\tau(s)=\frac{1}{1+e^{-as}}$	$\tau'(s)=a\tau(s)(1-\tau(s))$	(0,1)
하이퍼볼릭 탄젠트	$\tau(s)=\frac{2}{1+e^{-as}}-1$	$\tau'(s)=\frac{a}{2}(1-\tau(s)^2)$	(-1,1)
소프트플러스	$\tau(s)=\log_e(1+e^s)$	$\tau'(s)=\frac{1}{1+e^{-s}}$	$(0,\infty)$
렉티파이어(ReLU)	$\tau(s)=\max(0,s)$	$\tau'(s)=\begin{cases}0 & s<0\\ 1 & s>0\\ \text{불가} & s=0\end{cases}$	$[0,\infty)$

Chapter3.3 – 다층 퍼셉트론

- 다층 퍼셉트론의 구조



(a) 2층 퍼셉트론



(b) 3층 퍼셉트론

그림 3-14 다층 퍼셉트론의 구조

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론의 구조**
  - $u^{l}_{ji} \rightarrow (l-1)$층의 i번째와 (l) 층의 j번째를 연결하는 가중치를 의미
  - ex) 3층 페셉트론 기준
    - $u^{1}_{21} \rightarrow$ 입력층의 첫번째 노드와 첫번째 은닉층의 두번째 노드를 연결하는 가중치를 의미
    - $u^{2}_{cp} \rightarrow$ 첫번째 은닉층의 p번째 노드와 두번째 은닉층의 c번째 노드를 연결하는 가중치를 의미

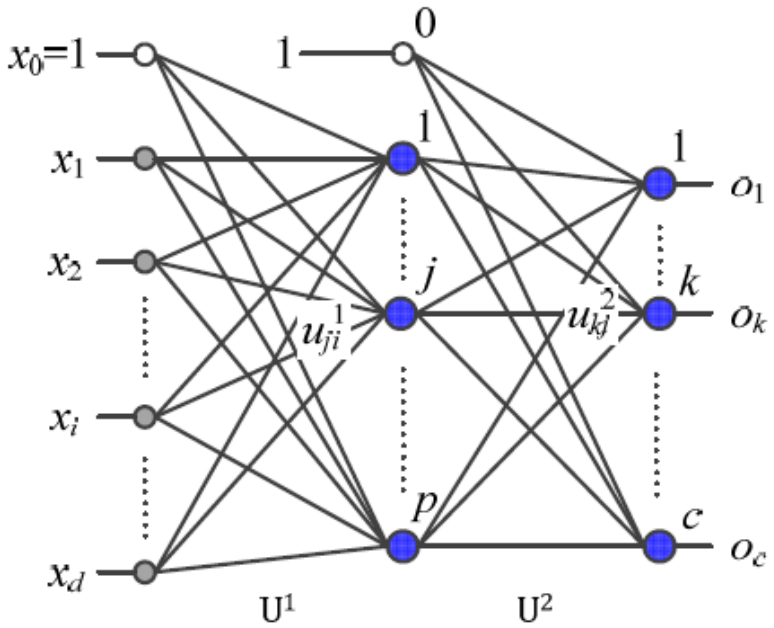
2층 퍼셉트론의 가중치 행렬:

$$\mathbf{U}^1 = \begin{pmatrix} u^1_{10} & u^1_{11} & \cdots & u^1_{1d} \\ u^1_{20} & u^1_{21} & \cdots & u^1_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ u^1_{p0} & u^1_{p1} & \cdots & u^1_{pd} \end{pmatrix}, \quad \mathbf{U}^2 = \begin{pmatrix} u^2_{10} & u^2_{11} & \cdots & u^2_{1p} \\ u^2_{20} & u^2_{21} & \cdots & u^2_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ u^2_{c0} & u^2_{c1} & \cdots & u^2_{cp} \end{pmatrix} \tag{3.11}$$

Chapter3.3 – 다층 퍼셉트론

- **병렬분산 구조**
  - 분산처리:
    - 하나의 노드가 다음 층의 모든 노드한테 영향을 미치는 것
    - Ex) 입력층의 i번째 노드가 은닉층의 노드 p개 모두에 영향을 미침
  - 병렬처리:
    - 네트워크를 분할하여 다수의 컴퓨터에서 처리 → 처리 시간 감소
    - Ex)
      - 하나의 노드 처리 시간 : t
      - 총 노드의 개수 : p개

→ 총 처리 시간 : t



(a) 2층 퍼셉트론

그림 3-14 다층 퍼셉트론의 구조

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론의 동작**
  - 2층 퍼셉트론
    - $f_1(x)$은 첫 번째 은닉층의 연산
    - $f_2(x)$은 출력층의 연산
  - 3층 퍼셉트론
    - $f_1(x)$은 첫 번째 은닉층의 연산
    - $f_2(x)$은 두 번째 은닉층의 연산
    - $f_3(x)$은 출력층의 연산

2층 퍼셉트론 : $\mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}))$

3층 퍼셉트론 : $\mathbf{o} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$

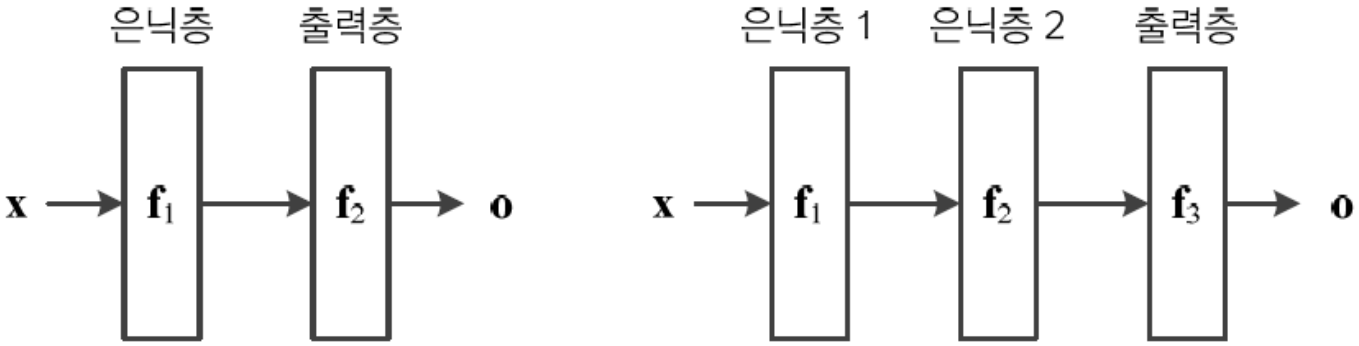


그림 3-15 다층 퍼셉트론을 간략화한 구조

Chapter3.3 – 다층 퍼셉트론

- **다층 퍼셉트론 동작**

j번째 은닉 노드의 연산:

$$z_j = \tau(zsum_j),\ j = 1,2,\cdots,p \tag{3.13}$$

이때 $zsum_j = \mathbf{u}_j^1\mathbf{x}$이고 $\mathbf{u}_j^1 = (u_{j0}^1, u_{j1}^1, \cdots, u_{jd}^1)$, $\mathbf{x} = (1, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$

k번째 출력 노드의 연산:

$$o_k = \tau(osum_k),\ k = 1,2,\cdots,c \tag{3.14}$$

이때 $osum_k = \mathbf{u}_k^2\mathbf{z}$이고 $\mathbf{u}_k^2 = (u_{k0}^2, u_{k1}^2, \cdots, u_{kp}^2)$, $\mathbf{z} = (1, z_1, z_2, \cdots, z_p)^{\mathrm{T}}$

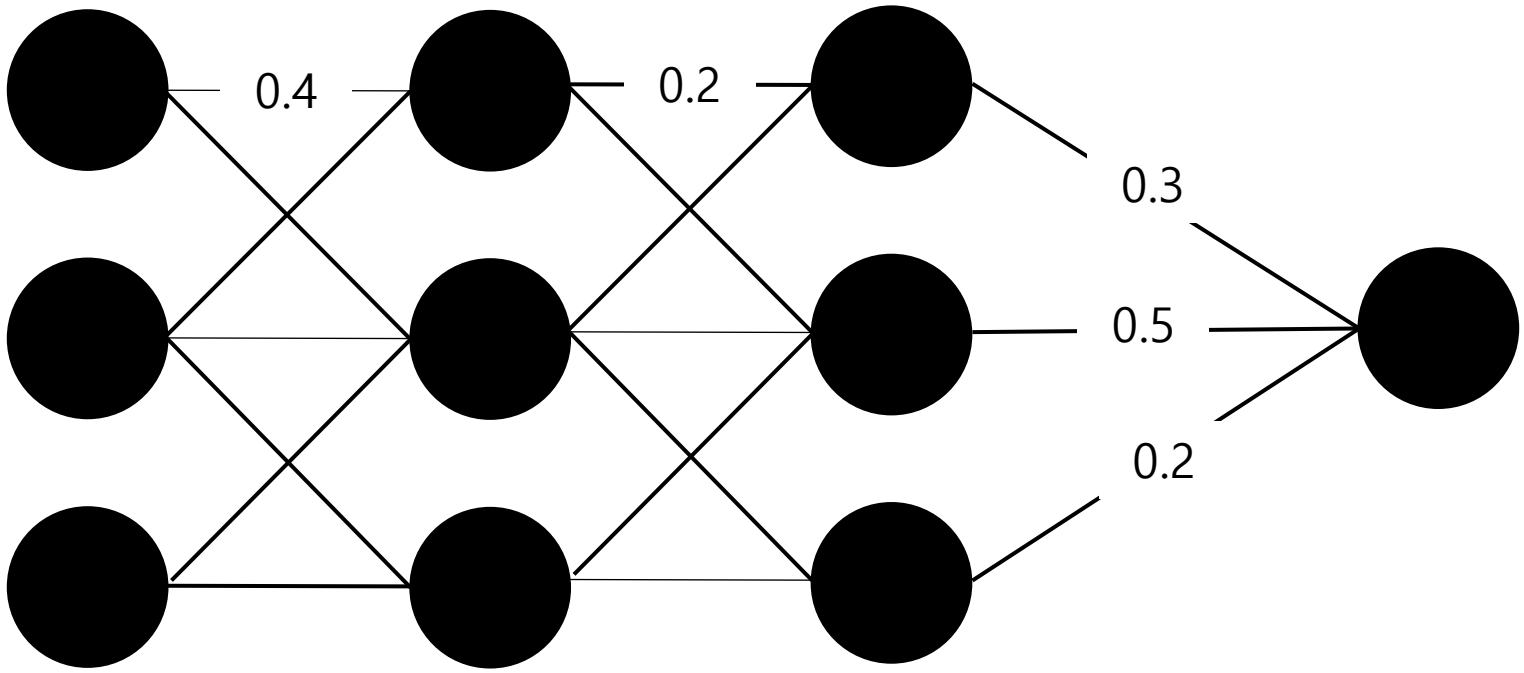
[redacted] $s = w_0 + \sum_{i=1}^{d} w_i x_i$,

- **간단한 행렬 표기** → $\mathbf{o} = \boldsymbol{\tau}(\mathbf{U}^2\boldsymbol{\tau}_h(\mathbf{U}^1\mathbf{x}))$ (3.15)

Chapter3.4 오류 역전파 알고리즘

Chapter3.4 – 오류 역전파 알고리즘

- 오류 역전파



Chapter3.4 – 오류 역전파 알고리즘

- **기계학습의 궁극적인 목표**
  - X를 완벽하게 Y로 매핑하는 최적의 함수 f를 알아내는 것

$$\left.\begin{aligned} &\mathbf{Y} = \mathbf{f}(\mathbf{X}) \\ &\text{풀어 쓰면} \quad \mathbf{y}_i = \mathbf{f}(\mathbf{x}_i), i = 1,2,\cdots,n \end{aligned}\right\} \tag{3.17}$$

기계학습이 해야 할 일

$$\Theta = \{U^1, U^2\}$$

$$\hat{\Theta} = \underset{\Theta}{\boldsymbol{argmin}} \|\boldsymbol{f}(\boldsymbol{X};\Theta) - \boldsymbol{Y}\|_2^2 \longrightarrow \text{유클리드 거리}$$

$\|\boldsymbol{f}(\boldsymbol{X};\Theta) - \boldsymbol{Y}\|^2$을 유클리드 거리로 계산했을 때 최소값으로 하는 Θ값을 구해라.

Chapter3.4 – 오류 역전파 알고리즘

- **목적함수의 정의**
  - 평균 제곱 오차$MSE(Mean\ Squre\ Error)$로 정의

$$\left.\begin{aligned} &\text{온라인 모드:} && e = \frac{1}{2}\|\mathbf{y}-\mathbf{o}\|_2^2 \\ &\text{배치 모드:} && e = \frac{1}{2n}\sum_{i=1}^{n}\|\mathbf{y}_i-\mathbf{o}_i\|_2^2 \end{aligned}\right\} \tag{3.19}$$

    - 온라인 모드: 하나의 샘플에 대해서 그레이디언트를 계산 후 바로 가중치 갱신
      - 스토캐스틱 경사 하강법에 사용
    - 배치 모드: 모든 샘플의 그레이디언트를 계산 후 한번에 가중치 갱신
      - 배치 경사 하강법에 사용

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파 알고리즘 설계**
  - 목적함수 (온라인 모드)

$$J(\Theta) = \frac{1}{2}\|\mathbf{y} - \mathbf{o}(\Theta)\|_2^2 \tag{3.20}$$

  - 가중치 갱신 규칙 (델타 규칙)

$$\left.\begin{aligned}\mathbf{U}^1 &= \mathbf{U}^1 - \rho\frac{\partial J}{\partial \mathbf{U}^1}\\ \mathbf{U}^2 &= \mathbf{U}^2 - \rho\frac{\partial J}{\partial \mathbf{U}^2}\end{aligned}\right\} \tag{3.21}$$

※ ρ는 학습률을 의미

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파 알고리즘 설계**
  - 하나의 샘플에 대해서 그레이디언트를 계산 후 즉시 갱신

알고리즘 3-3 다층 퍼셉트론을 위한 스토케스틱 경사 하강법

입력: 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$, 학습률 ρ
출력: 가중치 행렬 $\mathbf{U}^1$과 $\mathbf{U}^2$

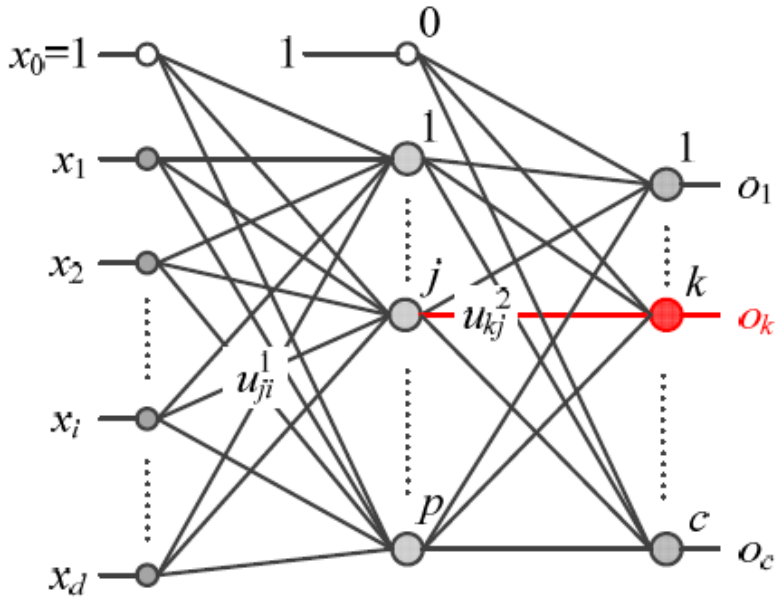
```
1  U¹과 U²를 초기화한다.
2  repeat
3     𝕏의 순서를 섞는다.
4     for (𝕏의 샘플 각각에 대해)
5           식 (3.15)로 전방 계산을 하여 o를 구한다.
6           ∂J/∂U¹와 ∂J/∂U²를 계산한다.
7           식 (3.21)로 U¹과 U²를 갱신한다.
8  until (멈춤 조건)
```

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파의 유도**

$$\frac{\partial J}{\partial u_{kj}^2} = \frac{\partial(0.5\|\mathbf{y}-\mathbf{o}(\mathbf{U}^1,\mathbf{U}^2)\|_2^2)}{\partial u_{kj}^2}$$
$$= \frac{\partial\left(0.5\sum_{q=1}^{c}(y_q-o_q)^2\right)}{\partial u_{kj}^2}$$
$$= \frac{\partial(0.5(y_k-o_k)^2)}{\partial u_{kj}^2}$$
$$= -(y_k-o_k)\frac{\partial o_k}{\partial u_{kj}^2}$$
$$= -(y_k-o_k)\frac{\partial \tau(osum_k)}{\partial u_{kj}^2}$$
$$= -(y_k-o_k)\tau'(osum_k)\frac{\partial osum_k}{\partial u_{kj}^2}$$
$$= -(y_k-o_k)\tau'(osum_k)z_j$$

$$osum_k = \cdots + z_j \cdot u_{kj}^2 + \cdots$$



(a) u_{kj}^2가 미치는 영향

그림 3-18 매개변수가 미치는 영향

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파의 유도**

$$\delta_k = (y_k - o_k)\tau'(osum_k), \qquad 1 \le k \le c \tag{3.22}$$

$$\frac{\partial J}{\partial u_{kj}^2} = \Delta u_{kj}^2 = -\delta_k z_j, \qquad 0 \le j \le p, 1 \le k \le c \tag{3.23}$$

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파의 유도**

$$\frac{\partial J}{\partial u_{ji}^1} = \frac{\partial(0.5\|\mathbf{y}-\mathbf{o}(\mathbf{U}^1,\mathbf{U}^2)\|_2^2)}{\partial u_{ji}^1}$$

$$= \frac{\partial\left(0.5\sum_{q=1}^{c}(y_q-o_q)^2\right)}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q-o_q)\frac{\partial o_q}{\partial u_{ji}^1} \quad \rightarrow \quad o_q = \tau(osum_q)$$

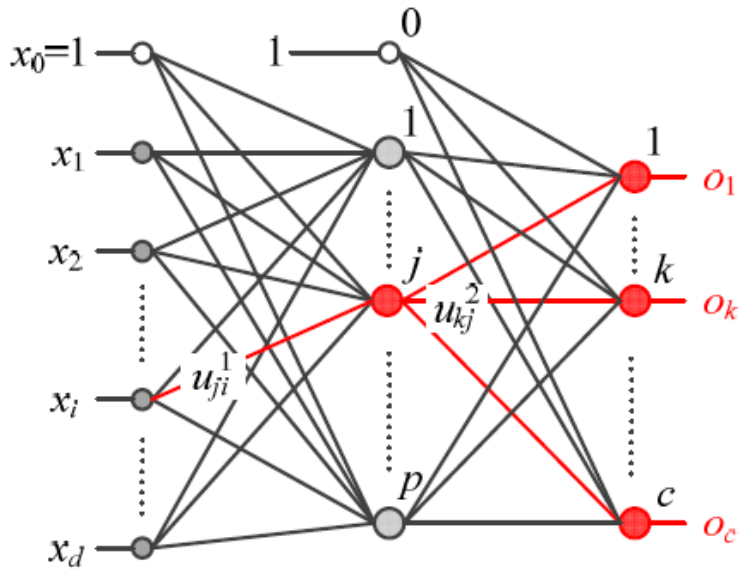
$$= -\sum_{q=1}^{c}(y_q-o_q)\tau'(osum_q)\frac{\partial osum_q}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q-o_q)\tau'(osum_q)\frac{\partial osum_q}{\partial z_j}\frac{\partial z_j}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q-o_q)\tau'(osum_q)u_{qj}^2\frac{\partial z_j}{\partial u_{ji}^1}$$

$$= -\sum_{q=1}^{c}(y_q-o_q)\tau'(osum_q)u_{qj}^2\tau'(zsum_j)x_i$$

$$= -\tau'(zsum_j)x_i\sum_{q=1}^{c}(y_q-o_q)\tau'(osum_q)u_{qj}^2$$



(b) u_{ji}^1이 미치는 영향

Chapter3.4 – 오류 역전파 알고리즘

- 오류 역전파의 유도

$$\eta_j = \tau'(zsum_j) \sum_{q=1}^{c} \delta_q u_{qj}^2 , \qquad 1 \le j \le p \tag{3.24}$$

$$\frac{\partial J}{\partial u_{ji}^1} = \Delta u_{ji}^1 = -\eta_j x_i, \qquad 0 \le i \le d, 1 \le j \le p \tag{3.25}$$

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파의 유도**

알고리즘 3-4 다층 퍼셉트론 학습을 위한 스토캐스틱 경사 하강법

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력: 가중치 행렬 $\mathbf{U}^1$과 $\mathbf{U}^2$

1 $\mathbf{U}^1$과 $\mathbf{U}^2$를 초기화한다.
2 repeat
3 $\mathbb{X}$의 순서를 섞는다.
4 for ($\mathbb{X}$의 샘플 각각에 대해)
5 현재 처리하는 샘플을 $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$, $\mathbf{y} = (y_1, y_2, \cdots, y_c)^{\mathrm{T}}$라 표기한다.
6 x_0과 z_0을 1로 설정한다. // 바이어스
// 전방 계산
7 for (j=1 to p) $zsum_j = \mathbf{u}_j^1\mathbf{x}$, $z_j = \tau(zsum_j)$ // 식 (3.13)
8 for (k=1 to c) $osum_k = \mathbf{u}_k^2\mathbf{z}$, $o_k = \tau(osum_k)$ // 식 (3.14)
// 오류 역전파
9 for (k=1 to c) $\delta_k = (y_k - o_k)\tau'(osum_k)$ // 식 (3.22)
10 for (k=1 to c) for (j=0 to p) $\Delta u_{kj}^2 = -\delta_k z_j$ // 식 (3.23)
11 for (j=1 to p) $\eta_j = \tau'(zsum_j)\sum_{q=1}^{c}\delta_q u_{qj}^2$ // 식 (3.24)
12 for (j=1 to p) for (i=0 to d) $\Delta u_{ji}^1 = -\eta_j x_i$ // 식 (3.25)
// 가중치 갱신
13 for (k=1 to c) for (j=0 to p) $u_{kj}^2 = u_{kj}^2 - \rho\Delta u_{kj}^2$ // 식 (3.21)
14 for (j=1 to p) for (i=0 to d) $u_{ji}^1 = u_{ji}^1 - \rho\Delta u_{ji}^1$ // 식 (3.21)
15 until (멈춤 조건)

Chapter3.4 – 오류 역전파 알고리즘

- **오류 역전파의 유도(행렬 표기)**
  - GPU를 사용한 고속 행렬 연산에 적합

알고리즘 3-5 다층 퍼셉트론 학습을 위한 스토캐스틱 경사 하강법(행렬 표기)

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력: 가중치 행렬 $\mathbf{U}^1$과 $\mathbf{U}^2$

1 $\mathbf{U}^1$과 $\mathbf{U}^2$를 초기화한다.
2 repeat
3 $\mathbb{X}$의 순서를 섞는다.
4 for ($\mathbb{X}$의 샘플 각각에 대해)
5 현재 처리하는 샘플을 $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$, $\mathbf{y} = (y_1, y_2, \cdots, y_c)^{\mathrm{T}}$라 표기한다.
6 x_0과 z_0을 1로 설정한다. // 바이어스
  // 전방 계산
7 $\mathbf{zsum} = \mathbf{U}^1\mathbf{x}$, $\tilde{\mathbf{z}} = \boldsymbol{\tau}(\mathbf{zsum})$ // 식 (3.13), $\mathbf{zsum}_{p*1}$, ${\mathbf{U}^1}_{p*(d+1)}$, $\mathbf{x}_{(d+1)*1}$, $\tilde{\mathbf{z}}_{p*1}$
8 $\mathbf{osum} = \mathbf{U}^2\mathbf{z}$, $\mathbf{o} = \boldsymbol{\tau}(\mathbf{osum})$ // 식 (3.14), $\mathbf{osum}_{c*1}$, ${\mathbf{U}^2}_{c*(p+1)}$, $\mathbf{z}_{(p+1)*1}$, $\mathbf{o}_{c*1}$
  // 오류 역전파
9 $\boldsymbol{\delta} = (\mathbf{y} - \mathbf{o}) \odot \boldsymbol{\tau}'(\mathbf{osum})$ // 식 (3.22), $\boldsymbol{\delta}_{c*1}$
10 $\Delta\mathbf{U}^2 = -\boldsymbol{\delta}\mathbf{z}^{\mathrm{T}}$ // 식 (3.23), ${\Delta\mathbf{U}^2}_{c*(p+1)}$
11 $\boldsymbol{\eta} = \left(\boldsymbol{\delta}^{\mathrm{T}}\tilde{\mathbf{U}}^2\right)^{\mathrm{T}} \odot \boldsymbol{\tau}'(\mathbf{zsum})$ // 식 (3.24), ${\tilde{\mathbf{U}}^2}_{c*p}$, $\boldsymbol{\eta}_{p*1}$
12 $\Delta\mathbf{U}^1 = -\boldsymbol{\eta}\mathbf{x}^{\mathrm{T}}$ // 식 (3.25), ${\Delta\mathbf{U}^1}_{p*(d+1)}$
  // 가중치 갱신
13 $\mathbf{U}^2 = \mathbf{U}^2 - \rho\Delta\mathbf{U}^2$ // 식 (3.21)
14 $\mathbf{U}^1 = \mathbf{U}^1 - \rho\Delta\mathbf{U}^1$ // 식 (3.21)
15 until (멈춤 조건)

Chapter3.5

미니배치 스토캐스틱 경사 하강법

Chapter3.5 – 미니배치 스토캐스틱 경사 하강법

- **미니배치 방식**
  - 샘플의 개수인 t를 정한다. (보통 수십 ~ 수백 정도 크기로 설정)
  - 훈련집합에서 t개의 샘플을 무작위로 뽑아 미니배치를 구성한다.
  - 미니배치에 속한 샘플들의 그레이디언트 평균으로 가중치를 갱신

알고리즘 3-6 다층 퍼셉트론 학습을 위한 '미니배치' 스토캐스틱 경사 하강법

입력: 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ, 미니배치 크기 t
출력: 가중치 행렬 $\mathbf{U}^1$과 $\mathbf{U}^2$

1 $\mathbf{U}^1$과 $\mathbf{U}^2$를 초기화한다.
2 repeat
3 $\mathbb{X}$와 $\mathbb{Y}$에서 t개의 샘플을 무작위로 뽑아 미니배치 $\mathbb{X}'$와 $\mathbb{Y}'$를 만든다.
4 $\Delta\mathbf{U}^2 = \mathbf{0}$, $\Delta\mathbf{U}^1 = \mathbf{0}$
5 for ($\mathbb{X}'$의 샘플 각각에 대해)
6 현재 처리하는 샘플을 $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^T$, $\mathbf{y} = (y_1, y_2, \cdots, y_c)^T$라 표기한다.
7 x_0와 z_0를 1로 설정한다. // 바이어스
  // 전방 계산
8 $\mathbf{zsum} = \mathbf{U}^1\mathbf{x}$, $\tilde{\mathbf{z}} = \tau(\mathbf{zsum})$ // 식 (3.13), $\mathbf{zsum}_{p*1}$, ${\mathbf{U}^1}_{p*(d+1)}$, $\mathbf{x}_{(d+1)*1}$, $\tilde{\mathbf{z}}_{p*1}$
9 $\mathbf{osum} = \mathbf{U}^2\mathbf{z}$, $\mathbf{o} = \tau(\mathbf{osum})$ // 식 (3.14), $\mathbf{osum}_{c*1}$, ${\mathbf{U}^2}_{c*(p+1)}$, $\mathbf{z}_{(p+1)*1}$, $\mathbf{o}_{c*1}$
  // 오류 역전파
10 $\boldsymbol{\delta} = (\mathbf{y} - \mathbf{o}) \odot \tau'(\mathbf{osum})$ // 식 (3.22), $\boldsymbol{\delta}_{c*1}$
11 $\Delta\mathbf{U}^2 = \Delta\mathbf{U}^2 + (-\boldsymbol{\delta}\mathbf{z}^T)$ // 식 (3.23)을 누적, ${\Delta\mathbf{U}^2}_{c*(p+1)}$
12 $\boldsymbol{\eta} = (\boldsymbol{\delta}^T\tilde{\mathbf{U}}^2)^T \odot \tau'(\mathbf{zsum})$ // 식 (3.24), ${\tilde{\mathbf{U}}^2}_{c*p}$, $\boldsymbol{\eta}_{p*1}$
13 $\Delta\mathbf{U}^1 = \Delta\mathbf{U}^1 + (-\boldsymbol{\eta}\mathbf{x}^T)$ // 식 (3.25)를 누적, ${\Delta\mathbf{U}^1}_{p*(d+1)}$
  // 가중치 갱신
14 $\mathbf{U}^2 = \mathbf{U}^2 - \rho\left(\frac{1}{t}\right)\Delta\mathbf{U}^2$ // 식 (3.21) - 평균 그레이디언트로 갱신
15 $\mathbf{U}^1 = \mathbf{U}^1 - \rho\left(\frac{1}{t}\right)\Delta\mathbf{U}^1$ // 식 (3.21) - 평균 그레이디언트로 갱신
16 until (멈춤 조건)

- 수렴 속도가 증가
- GPU를 이용한 병렬처리에도 유리
- 현대 기계학습은 미니매치를 표준처럼 여김

Chapter3.6

다층 퍼셉트론에 대한 인식

Chapter3.6 – 다층 퍼셉트론에 의한 인식(예측)

- **예측 (테스트) 단계**
  - 전방 계산 한번만 실행 → 선형 가능해짐으로 결과값 나오는게 빠름

알고리즘 3-7 다층 퍼셉트론을 이용한 인식

입력: 테스트 샘플 $\mathbf{x}$ // 신경망의 가중치 $\mathbf{U}^1$과 $\mathbf{U}^2$는 이미 설정되었다고 가정함.
출력: 부류 y

1. $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)^{\mathrm{T}}$로 확장하고, x_0과 z_0을 1로 설정한다.
2. $\mathbf{zsum} = \mathbf{U}^1\mathbf{x}$
3. $\tilde{\mathbf{z}} = \boldsymbol{\tau}(\mathbf{zsum})$ // 식 (3.13)
4. $\mathbf{osum} = \mathbf{U}^2\mathbf{z}$
5. $\mathbf{o} = \boldsymbol{\tau}(\mathbf{osum})$ // 식 (3.14)
6. $\mathbf{o}$에서 가장 큰 값을 가지는 노드에 해당하는 부류 번호를 y에 대입한다.

$$y = \underset{k}{\operatorname{argmax}}\, o_k$$

Chapter3.7
다층 퍼셉트론의 특성

Chapter3.7 – 다층 퍼셉트론의 특성

- **오류 역전파 알고리즘의 빠른 속도**

d = 입력 노드 개수
p = 은닉 노드 개수
c = 출력 노드 개수

n = 샘플의 개수
q = 세대 수 (epoch)

표 3–2 전방 계산과 오류 역전파 과정이 사용하는 연산 횟수

과정	수식	덧셈	곱셈
전방 계산	식 (3.13)	dp	dp
	식 (3.14)	pc	pc
오류 역전파	식 (3.22)	c	c
	식 (3.23)		cp
	식 (3.24)	cp	cp
	식 (3.25)		dp
	식 (3.21)	$cp+dp$	$cp+dp$

전방 계산 연산 횟수: 2dp + 2pc
오류 역전파 연산 횟수: 3dp + 5pc
→ 한 샘플에 대한 연산 횟수: 5dp + 7pc

학습 알고리즘 계산 시간: θ((dp + pc)nq)

Chapter3.7 – 다층 퍼셉트론의 특징

- **모든 함수를 정확하게 근사할 수 있는 능력**

> "… standard multilayer feedforward network architectures using arbitrary squashing functions can approximate virtually any function of interest to any desired degree of accuracy, provided sufficiently many hidden units are available. … 은닉 노드가 충분히 많다면, 포화함수(활성함수)로 무엇을 사용하든 표준 다층 퍼셉트론은 어떤 함수라도 원하는 정확도만큼 근사화할 수 있다."

  - 접근방법1) 은닉층이 하나인 신경망을 개선하려는 노력 (1980~1990)
    - 은닉층 하나만 가진 퍼셉트론은 범용근사자

  - 접근방법2) 은닉층을 여럿으로 늘려 깊은 신경망으로 확장하여 개선하려는 노력

Chapter3.6 – 다층 퍼셉트론에 의한 인식(예측)

- **성능 향상을 위한 휴리스틱의 중요성**
  - 휴리스틱: 복잡한 문제를 해결하기 위해 사용되는 규칙 또는 방법 (시간 절약, 단순화된 결정을 내리기 위함)
    - 아키텍처: 은닉층과 은닉 노드의 개수를 정해야 한다. 은닉층과 은닉 노드를 늘리면 신경망의 용량은 커지는 대신, 추정할 매개변수가 많아지고 학습 과정에서 과잉적합할 가능성이 커진다. 1.6절에서 소개한 바와 같이 현대 기계 학습은 복잡한 모델을 사용하되, 적절한 규제 기법을 적용하는 경향이 있다.
    - 초깃값: [알고리즘 3–4]의 라인 1에서 가중치를 초기화한다. 보통 난수를 생성하여 설정하는데, 값의 범위와 분포가 중요하다. 이 주제는 5.2.2절에서 다룬다.
    - 학습률: 처음부터 끝까지 같은 학습률을 사용하는 방식과 처음에는 큰 값으로 시작하고 점점 줄이는 적응적 방식이 있다. 5.2.4절에서 여러 가지 적응적 학습률 기법을 소개한다.
    - 활성함수: 초창기 다층 퍼셉트론은 주로 로지스틱 시그모이드나 tanh 함수를 사용했는데, 은닉층의 개수를 늘림에 따라 그레이디언트 소멸과 같은 몇 가지 문제가 발생한다. 따라서 깊은 신경망은 주로 ReLU 함수를 사용한다. 5.2.5절에서 여러 가지 ReLU 함수를 설명한다.

Chapter3.7 – 다층 퍼셉트론의 특징

- **실용적인 성능**
  - 1980년대와 1990년대
    - 인쇄체 문자 또는 필기체 문자를 인식
    - 우편물 자동분류기, 은행 또는 관공서의 전표 인식기, 자동차 번호판 인식기 등이 등장
  - 한계
    - 음성 인식은 잡음이 심한 환경에서 성능이 급격히 떨어짐
    - 필기체로 쓴 주소를 인식하는데 힘듦을 겪음
    - 우편물 자동분류기 옆에는 사람이 대기하다가 인식에 실패한 우편물을 수동으로 인식
    - 수가 한정된 체스, 장기는 좋은 성능을 보이나 수가 많은 바둑에서는 한계를 보임