

Machine Learning 이해와 활용

- Advanced modeling -

가톨릭대학교 미디어기술콘텐츠학과
강호철

강의 내용

- **비지도 학습의 특징공간**

- 공간변환
- 선형 인자 모델
- 매니폴드 학습

- **서포트 벡터 머신**

- 기본 개념
- 선형 SVM

- **인공 신경망**

- 기본개념
- 단일 퍼셉트론
- 다중 퍼셉트론

- **양상블**

- 랜덤 포레스트
- 그레이디언트 부스팅 회귀 트리
- Voting

- **딥러닝 소개**

- 기본 개념
- 케라스
- 합성곱 신경망



비지도 학습의 특징공간



공간 변환의 이해

■ 개념

- 2개의 구집을 가진 2차원 특징 공간을 극좌표 공간으로 변환하면
1차원 만으로 2개 군집 표현 가능

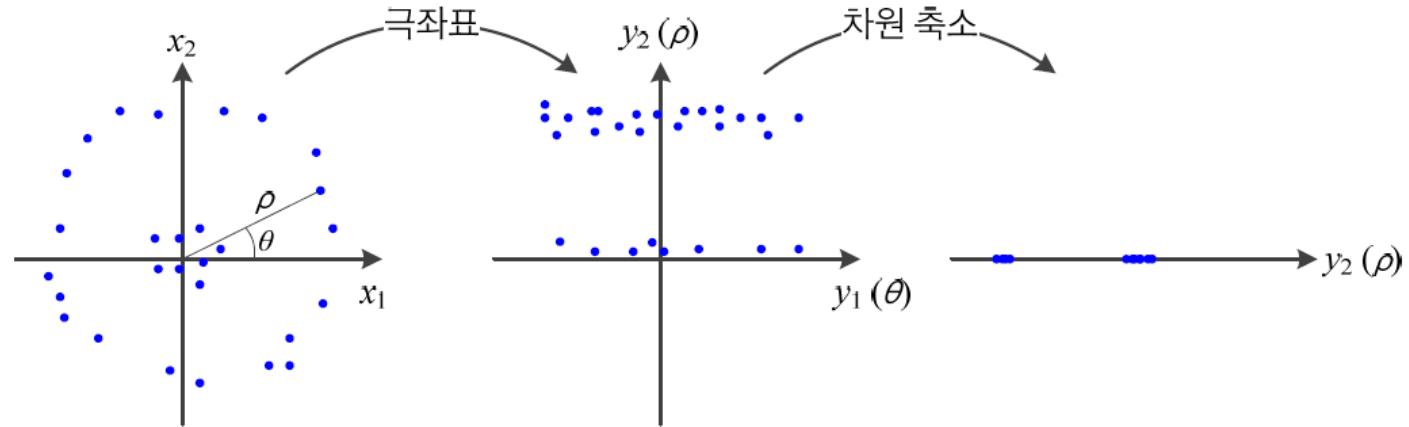


그림 6-16 공간 변환의 예

- 실제 문제에서는 비지도 학습을 이용하여 최적의 공간 변환을 자동으로 알아내야 함 (표현학습)

공간 변환의 이해

- 인코딩과 디코딩

- 원래 공간을 다른 공간으로 변환 (인코딩)
- 변환된 공간을 원래 공간으로 역변환 (디코딩)

$$\hat{\mathbf{x}} = g(f(\mathbf{x})) \quad (6.16)$$

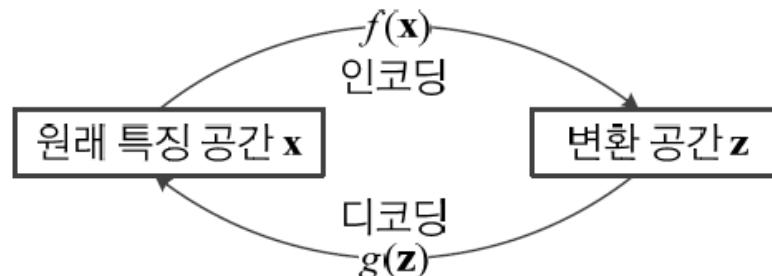


그림 6-17 공간 변환과 역변환

- 예) 데이터 압축의 경우, 역변환으로 얻은 $\hat{\mathbf{x}}$ 은 원래 신호 \mathbf{x} 와 가급적 같아야 함
- 예) 데이터 가시화에서는 2차원 또는 3차원의 \mathbf{z} 공간으로 변환. 디코딩은 불필요

선형 인자 모델 (Linear factor model)

- 인자 (factor)
 - 관찰되지 않는 변수 (앞의 예에서 Z)
 - 잠복 변수(latent variable) 혹은 은닉 변수(hidden variable) 이라고도 함
 - 예) 학생들의 시험 점수 데이터
 - 국어, 영어, 수학, 물리, 화학, 생물, 철학, 문학 등 8개 과목의 점수
 - N명의 학생 데이터 수집하고 분석하는 방법
 - 전통적인 통계 기법
 - 선형 인자 모델 사용
 - $X_n = (x_1, x_2, \dots, x_8) : 8\text{차원}$
 - $Z_n = (z_1, z_2) : 2\text{차원}$



선형 인자 모델 (Linear factor model)

- 선형 인자 모델 개념
 - 선형 연산을 이용하여 관찰한 데이터를 인자로 변환하는 방법
 - 공간 변환
 - 행렬 곱으로 인코딩과 디코딩 과정 표현

$$f: \mathbf{z} = \mathbf{W}_{enc}\mathbf{x} + \boldsymbol{\alpha}_{enc} \quad (6.17)$$

$$g: \mathbf{x} = \mathbf{W}_{dec}\mathbf{z} + \boldsymbol{\alpha}_{dec} \quad (6.18)$$

- $\boldsymbol{\alpha}$ 는 데이터를 원점으로 이동하거나 잡음을 추가하는 등의 역할
- 인자 \mathbf{z} 와 추가 항 $\boldsymbol{\alpha}$ 에 따라 여러 가지 모델이 존재
 - \mathbf{z} 에 확률 개념이 없고 $\boldsymbol{\alpha}$ 를 생략하면 PCA
 - 관찰 벡터 \mathbf{x} 와 인자 \mathbf{z} 는 결정론적인 1:1 매핑 관계
 - \mathbf{z} 와 $\boldsymbol{\alpha}$ 가 가우시안 분포를 따른다고 가정하면 확률 PCA (probabilistic PCA)
 - \mathbf{z} 가 비가우시안 분포를 따른다고 가정하는 ICA



패턴 인식에서의 특징

■ 일반적인 특징 추출

■ 특징 추출의 예

- 필기 숫자 인식

6

- 크기 정규화
- 이진화

00001100
00010000
00100000
01100000
11000110
11000011
11000001
11111110

특
징
추
출

방법 1: 화소 각각을 특징으로 삼음
64 차원 특징 벡터
 $x=(0,0,0,0,1,1,0,0,0,0,0,1,\dots,1,1,0)^T$

방법 2: 가로
이등분과 세
로 이등분하
여 검은 화소
의 개수 비율
을 특징으로
삼음

$$\begin{array}{r} x_2=14/10 \\ \hline 14 & 10 \\ 00001100 \\ 00010000 \\ 00100000 \\ 01100000 \\ \hline 11000110 \\ 11000011 \\ 11000001 \\ 11111110 \end{array} \left| \begin{array}{l} 6 \\ 18 \end{array} \right. \quad x_1=6/18$$

$$x=(6/18, 14/10)^T$$

또 다른 방법으로는?

■ 특징의 우수성 기준

- 분별력
- 차원

패턴 인식에서의 특징

- 일반적인 특징 추출

- 특징 생성

- 외부의 물리적 패턴을 특징 벡터라는 수학적 표현으로 변환

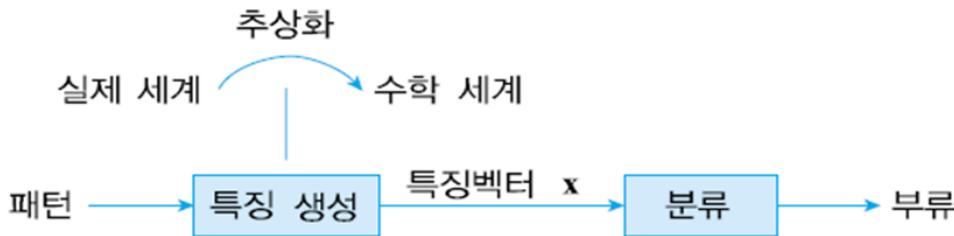


그림 8.1 패턴 인식의 일반적인 흐름

- 특징 생성 과정은 매우 다양

- 특징 추출은 외부 환경에 맞게 설계해야 하기 때문
 - 숫자와 한글은 다른 특징 필요할 수 있음
 - 한글도 통째로 인식하는 방법과 자소로 분할한 후 인식하는 방법이 다른 특징 필요할 수 있음
 - 정면 얼굴로 국한하는 하는 경우와 제약이 없는 얼굴 인식의 특징은 다를 수 있음

패턴 인식에서의 특징

- 일반적인 특징 추출
 - 센싱으로 얻은 신호의 다양성
 - 영상
 - 시간성 신호
 - 측정 벡터
 - 특징 추출과 특징 선택

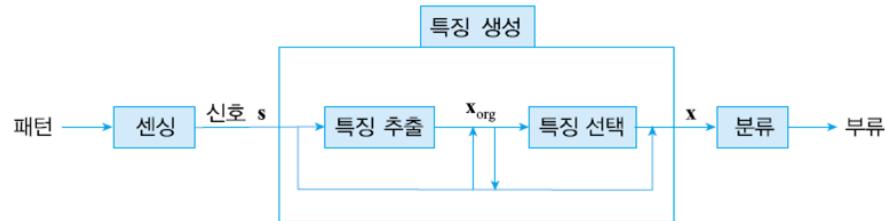


그림 8.2 특징 생성의 절차

$$\left. \begin{array}{l} \text{특징추출 } \mathbf{x}_{\text{org}} = e(\mathbf{s}) \\ \text{특징선택 } \mathbf{x} = s(\mathbf{x}_{\text{org}}) \end{array} \right\}$$

- 다양한 상황

$$\left. \begin{array}{l} \mathbf{x} = \mathbf{s} \\ \mathbf{x} = e(\mathbf{s}) \\ \mathbf{x} = s(\mathbf{s}) \\ \mathbf{x} = s(e(\mathbf{s})) \end{array} \right\}$$



주성분 분석

- PCA, Principal Component Analysis
 - 훈련 집합을 이용하여 매개 변수 추정
 - 특징 추출하는데 이용
 - 정보 손실을 최소화하는 조건에서 차원 축소
- 동기
 - 정보 손실을 최소화하며 신호 s 를 보다 낮은 차원의 특징 벡터 x 로 변환
 - $d < D$, (D : 신호의 차원, d : 특징 벡터의 차원)
 - 변환 행렬 U 필요

$$x = e(s; U) = Us$$

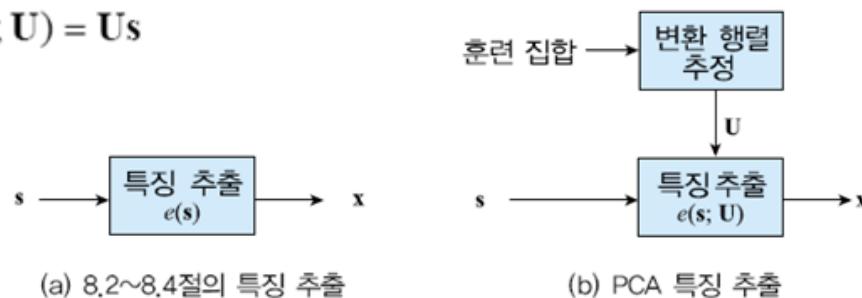


그림 8.12 특징 추출 방법의 비교

주성분 분석

- 데이터 처리
 - 데이터를 원점 중심으로 옮기는 전처리 수행

$$\left. \begin{array}{l} \mathbf{x}_i = \mathbf{x}_i - \boldsymbol{\mu}, \quad i = 1, 2, \dots, n \\ \text{○] 때 } \boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \end{array} \right\} \quad (6.19)$$

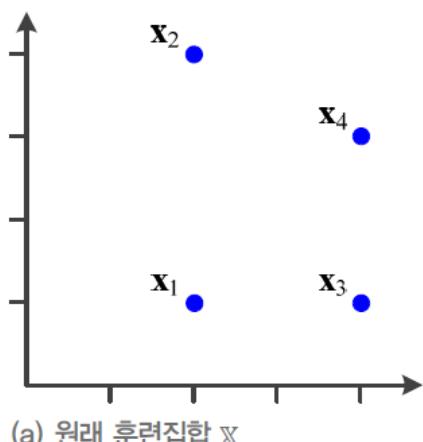
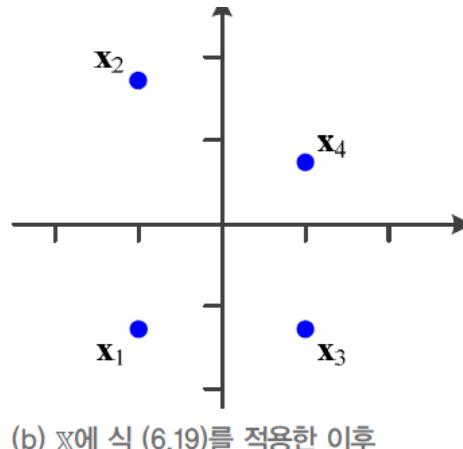


그림 6-18 \mathbf{x} 의 평균을 0으로 변환



(b) \mathbf{x} 에 식 (6.19)을 적용한 이후

주성분 분석

■ 변환식

- 6.17에서 z 에 확률 개념이 없고 α 를 생략하면 주성분 분석이 됨
- 변환행렬 W 는 d^*q
- 주성분 분석은 d 차원의 x 를 q 차원의 z 로 변환 ($q < d$)
 - W 의 j 번째 열 벡터와의 내적 (x 를 u_j 로 투영)

$$z = W^T x \\ \textcircled{O} \text{ 때 } W = (u_1 \ u_2 \ \cdots \ u_q) \textcircled{O} \text{ 고, } u_j = (u_{1j}, u_{2j}, \dots, u_{dj})^T \quad \left. \right\} \quad (6.20)$$

예) 2차원을 1차원으로 변환하는 상황($d = 2, q = 1$)

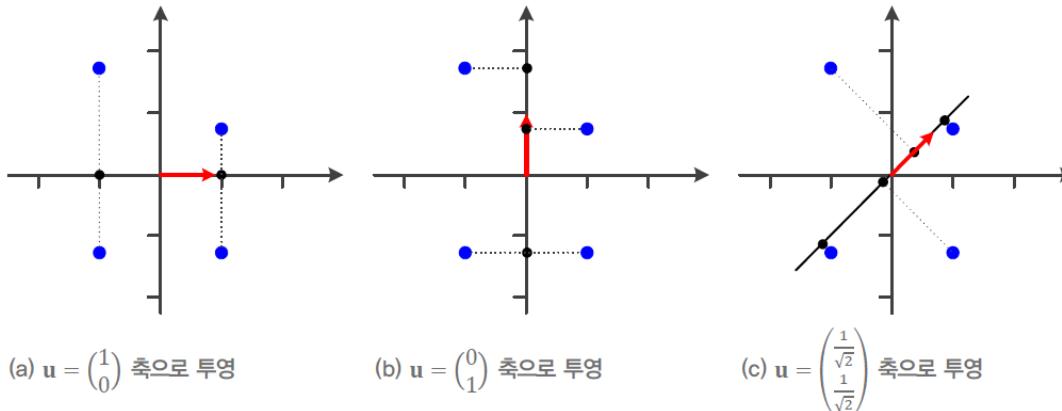


그림 6-19 투영에 의해 2차원을 1차원으로 변환

주성분 분석

- 정보 손실

- 훈련집합이 가지고 있는 정보

- 샘플들 간 거리, 상대적 위치 등

- 주성분 분석은 손실을 최소화 하고 저차원으로 변환해야 함

- 앞의 그림에서 정보 손실의 예

- [그림 6-19(a)]는 x_1 과 x_2 쌍, x_3 과 x_4 쌍이 같은 점으로 변환되는 정보 손실

- [그림 6-19(b)]는 x_1 과 x_3 쌍이 같은 점으로 변환되는 정보 손실

- [그림 6-19(c)]는 4개 점이 모두 다른 점으로 변환되어 정보 손실이 가장 적음

- 주성분의 목적

- 주성분 분석은 변환된 훈련집합 $\mathbb{Z} = \{z_1, z_2, \dots, z_n\}$ 의 **분산이 클수록 정보 손실이 적다고 판단**
 - **분산이 가장 크게 되도록 변환하는 w 를 찾아라!**

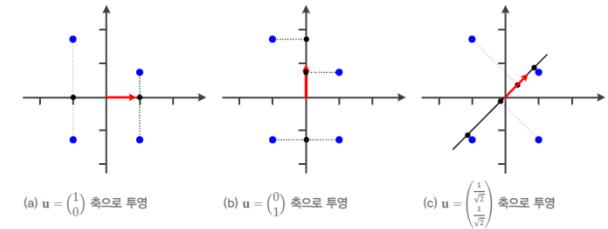


그림 6-19 투영에 의해 2차원을 1차원으로 변환

주성분 분석

예제 6-2 [그림 6-19]의 세 가지 경우의 분산

[그림 6-18(a)]의 훈련집합에 식 (6.19)를 적용하기 전과 후는 다음과 같다.

$$\mathbf{x}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \Rightarrow \mathbf{x}_1 = \begin{pmatrix} -1 \\ -1.25 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} -1 \\ 1.75 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 1 \\ -1.25 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 1 \\ 0.75 \end{pmatrix}$$

[그림 6-19(a)]의 $\mathbf{u} = (1 \ 0)^T$ 축으로 투영된 점은 다음과 같다. $z_1 \sim z_4$ 의 분산은 1.00이다.

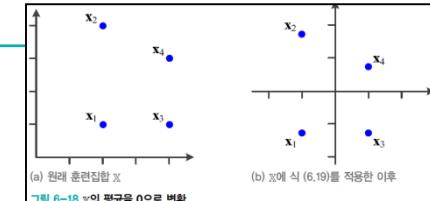
$$z_1 = (1 \ 0) \begin{pmatrix} -1 \\ -1.25 \end{pmatrix} = -1, \ z_2 = (1 \ 0) \begin{pmatrix} -1 \\ 1.75 \end{pmatrix} = -1, \ z_3 = (1 \ 0) \begin{pmatrix} 1 \\ -1.25 \end{pmatrix} = 1, \ z_4 = (1 \ 0) \begin{pmatrix} 1 \\ 0.75 \end{pmatrix} = 1$$

이제 [그림 6-19(c)]의 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right)^T$ 축으로 투영된 점을 구해 보자. $z_1 \sim z_4$ 의 분산은 1.093이다.

$$z_1 = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right) \begin{pmatrix} -1 \\ -1.25 \end{pmatrix} = -1.591, \ z_2 = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right) \begin{pmatrix} -1 \\ 1.75 \end{pmatrix} = 0.530,$$

$$z_3 = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right) \begin{pmatrix} 1 \\ -1.25 \end{pmatrix} = -0.177, \ z_4 = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right) \begin{pmatrix} 1 \\ 0.75 \end{pmatrix} = 1.237$$

따라서 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right)^T$ 축이 $\mathbf{u} = (1 \ 0)^T$ 보다 우수하다고 할 수 있다. 그렇다면 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \right)^T$ 보다 더 좋은 축이 있을까? 이제부터 최적해를 찾는 방법을 살펴보자.



(a) 원래 훈련집합 X

(b) X 에 식 (6.19)를 적용한 이후

그림 6-18 X의 청교을 0으로 변환

주성분 분석

■ PCA 최적화 문제

문제 6.1 $\mathbb{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ 의 분산을 최대화하는 q 개의 축, 즉 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_q$ 를 찾아라. 이 단위 벡터는 식 (6.20)에 따라 변환 행렬 \mathbf{W} 를 구성한다.

$$\left. \begin{array}{l} \mathbf{z} = \mathbf{W}^T \mathbf{x} \\ \text{이때 } \mathbf{W} = (\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_q) \text{이고, } \mathbf{u}_j = (u_{1j}, u_{2j}, \dots, u_{dj})^T \end{array} \right\} \quad (6.20)$$

- $q = 1$ 로 단순화 하여 분산 계산, \mathbf{z} 의 평균은 0으로 전처리
- 다음 식을 최대로 하는 \mathbf{U} 를 구해야 함

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (z_i - \bar{z})^2 = \frac{1}{n} \sum_{i=1}^n z_i^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}_i)^2 = \mathbf{u}^T \Sigma \mathbf{u} \quad (6.21)$$

문제 6.2 식 (6.21)의 분산 σ^2 을 최대로 하는 \mathbf{u} 를 찾아라.



(참고) 평균과 분산

- 평균과 분산

$$\left. \begin{array}{l} \text{평균 } \mu = \frac{1}{n} \sum_{i=1}^n x_i \\ \text{분산 } \sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \end{array} \right\} \quad (2.36)$$

- 평균 벡터와 공분산 행렬

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad (2.37)$$

$$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \quad (2.39)$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \ddots & \sigma_{2d} \\ \vdots & & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \dots & \sigma_{dd} \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2d} \\ \vdots & \ddots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \dots & \sigma_d^2 \end{pmatrix}$$



주성분 분석

■ PCA 최적화 문제

■ 문제 풀이

- \mathbf{u} 가 단위 벡터라는 사실을 적용하여 문제를 다시 쓰면,

문제 6.3 $L(\mathbf{u}) = \mathbf{u}^T \Sigma \mathbf{u} + \lambda(1 - \mathbf{u}^T \mathbf{u})$ 를 최대로 하는 \mathbf{u} 를 찾아라.

- $L(\mathbf{u})$ 를 \mathbf{u} 로 미분하면, $\frac{\partial L(\mathbf{u})}{\partial \mathbf{u}} = 2\Sigma\mathbf{u} - 2\lambda\mathbf{u}$
- $\frac{\partial L}{\partial \mathbf{u}} = 0$ 을 풀면,

$$\Sigma\mathbf{u} = \lambda\mathbf{u} \quad (6.22)$$

■ 주성분 분석의 학습 알고리즘

1. 훈련집합으로 공분산 행렬 Σ 를 계산한다.
2. 식 (6.22)를 풀어 d 개의 고윳값과 고유 벡터를 구한다.
3. 고윳값이 큰 순서대로 $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \dots, \mathbf{u}_d$ 를 나열한다. (이들을 주성분이라 부름)
4. q 개의 주성분 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_q$ 를 선택하여 식 (6.20)에 있는 행렬 \mathbf{W} 에 채운다.

$$\begin{aligned}\partial L(\mathbf{u}) / \partial \mathbf{u} &= \partial \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{u}^T \mathbf{s}_i - \mathbf{u}^T \bar{\mathbf{s}})^2 + \lambda(1 - \mathbf{u}^T \mathbf{u}) \right) / \partial \mathbf{u} \\ &= \frac{2}{N} \sum_{i=1}^N (\mathbf{u}^T \mathbf{s}_i - \mathbf{u}^T \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}}) - 2\lambda\mathbf{u} \\ &= 2\mathbf{u}^T \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}}) \right) - 2\lambda\mathbf{u} \\ &= 2\mathbf{u}^T \Sigma - 2\lambda\mathbf{u} \\ &= 2\Sigma\mathbf{u} - 2\lambda\mathbf{u}\end{aligned}$$

주성분 분석

예제 6-3

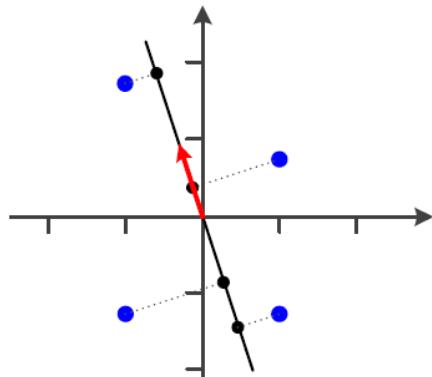
PCA 수행

식 (6.22)를 풀어 [그림 6-18]에 있는 데이터의 최적해를 구해 보자. 먼저 공분산 행렬 Σ 와 Σ 의 고윳값과 고유 벡터를 구하면 다음과 같다. 공분산을 구하는 방법은 2장의 식 (2.39)를 참조하라.

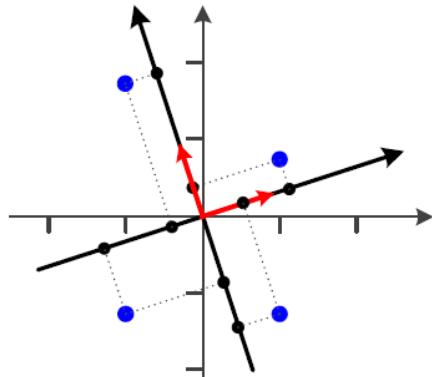
$$\Sigma = \begin{pmatrix} 1.000 & -0.250 \\ -0.250 & 1.688 \end{pmatrix}$$

$$\lambda_1 = 1.7688, \mathbf{u}_1 = \begin{pmatrix} -0.3092 \\ 0.9510 \end{pmatrix}, \lambda_2 = 0.9187, \mathbf{u}_2 = \begin{pmatrix} -0.9510 \\ -0.3092 \end{pmatrix}$$

고유 벡터 2개 중 고윳값이 큰 \mathbf{u}_1 을 선택하고, \mathbf{u}_1 에 샘플 4개를 투영하면 [그림 6-20(a)]가 된다. 변환된 점의 분산은 1.7688로 [그림 6-19]에 있는 축보다 훨씬 크다는 사실을 확인할 수 있다. \mathbf{u}_1 은 PCA 알고리즘으로 찾은 최적으로서 더 좋은 축은 없다.



(a) 축 1개 사용



(b) 축 2개 사용

그림 6-20 PCA가 찾은 최적 변환

주성분 분석

■ 디코딩 과정

- 역변환은 $\mathbf{x} = (\mathbf{W}^T)^{-1} \mathbf{z}$ 인데, \mathbf{W} 가 정규직교 행렬이므로 식 (6.23)이 됨

$$\tilde{\mathbf{x}} = \mathbf{W}\mathbf{z} \quad (6.23)$$

- $q = d$ 로 설정하면 \mathbf{W} 가 $d * d$ 이고 $\tilde{\mathbf{x}}$ 는 원래 샘플 \mathbf{x} 와 같게 됨([그림 6-20(b)]의 예시)
 - 원래 공간을 단지 일정한 양만큼 회전하는 것에 불과

■ 실제로는 $q < d$ 로 설정하여 차원 축소를 꾀함

- 많은 응용이 있음
 - 데이터 압축
 - $q = 2$ 또는 $q = 3$ 으로 설정하여 2차원 또는 3차원으로 축소하여 데이터 가시화



주성분 분석

알고리즘 [8.3] PCA에 의한 변환 행렬 구함

입력: 훈련 집합 $X = \{s_1, s_2, \dots, s_N\}$, 원하는 차원 d

출력: 변환 행렬 U , 평균 벡터 \bar{s}

알고리즘:

1. $\bar{s} = \frac{1}{N} \sum_{i=1}^N s_i$ // X 의 평균 벡터
2. **for** ($i = 1$ **to** N) $s'_i = s_i - \bar{s}$; // 평균 벡터를 빼줌
3. $s'_i, 1 \leq i \leq N$ 의 공분산 행렬 Σ 를 구한다.
4. Σ 의 고유 벡터와 고유 값을 구한다.
5. 고유 값 기준으로 가장 큰 d 개의 고유 벡터를 선택한다.
이들을 u_1, u_2, \dots, u_d 라 하자.
6. (8.34)로 변환 행렬 U 를 만든다.
7. **return** U, \bar{s} ;

$$U = \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_d^T \end{pmatrix}$$

알고리즘 [8.4] PCA에 의한 특징 추출

입력: 변환 행렬 U , 평균 벡터 \bar{s} , 샘플 s

출력: 특징 벡터 x

알고리즘:

1. $s = s - \bar{s}$; // 샘플에서 평균 벡터를 뺀다.
2. $x = Us$; // (8.35)
3. **return** x



주성분 분석

■ 사례 연구

■ 고유 얼굴

- ▣ 1990년대 초 Turk와 Pentland가 제안
- ▣ 얼굴 인식에서 가장 널리 쓰이는 방법 중의 하나

■ 분류기 학습과 인식 단계

1. $D = n^2$ 차원을 몇 차원으로 줄일지 결정하고 그것을 d 라 한다.
2. X 와 d 를 알고리즘 [8.3]의 입력으로 주어 변환 행렬 \mathbf{U} 와 평균 벡터 $\bar{\mathbf{s}}$ 를 구한다.
3. (분류기 학습) X 의 샘플 각각을 알고리즘 [8.4]에 넣어 특징 벡터를 추출한다.
이렇게 구한 특징 벡터 집합으로 새로운 훈련 집합 X' 를 구성한다.
4. (분류기 학습) X' 로 분류기를 훈련한다. (이때 분류기는 신경망, SVM, k -NN, 트리 분류기 등 어느 것이라도 좋다.)
5. (인식) 인식을 해야 하는 새로운 얼굴 영상이 들어오면 (8.36)으로 신호 \mathbf{s} 표현으로 바꾸고 이것을 알고리즘 [8.4]에 넣어 특징 벡터 \mathbf{x} 를 추출한다. \mathbf{x} 를 분류기에 넣어 인식한다.

- 변환 행렬 \mathbf{U} 를 구성하는 고유 벡터 \mathbf{u}_i 를 고유 얼굴이라 부름
- 고유얼굴 기법: 256*256 얼굴 영상($d = 65536$)을 $q = 7$ 차원으로 변환하여 얼굴 인식(정면 얼굴에 대해 96% 정확률) → 상위 몇 개의 고유 벡터가 대부분 정보를 가짐

주성분 분석

- 요약
 - 비지도 학습을 사용해 데이터 변환하는 이유
 - 시각화
 - 데이터 압축
 - 추가적인 (전)처리
- 실습



선형 분별

- (참고) LDA, Linear Discriminant Analysis
 - Fisher's Linear Discriminant
 - 데이터 분포를 학습하여 결정경계를 만들고 데이터를 분류하는 모델
 - 특징 추출 보다는 분류기 설계에 해당
 - PCA와 원리가 비슷하나 목표가 다름
 - PCA는 정보 손실 최소화 (샘플의 부류 정보 사용 안함)
 - LDA는 분별력을 최대화 (샘플의 부류 정보 사용함)
- 원리

- 축으로의 투영 $y = \mathbf{w}^T \mathbf{x}$
- 분별력 관점에서 유리한 것은?

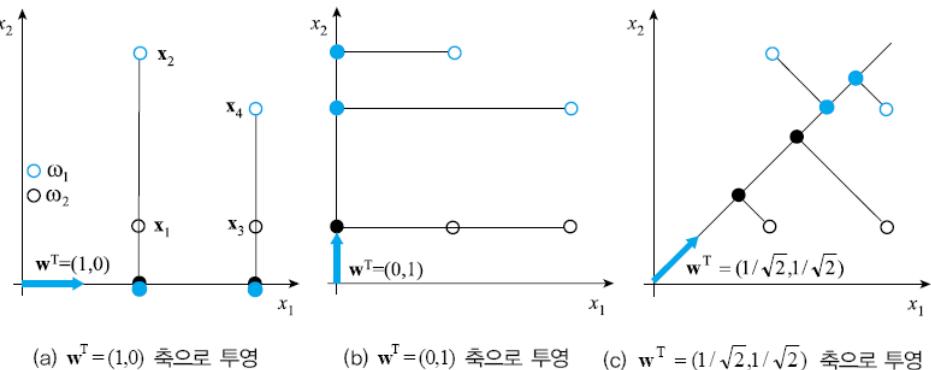


그림 8.15 2 차원 공간을 1 차원 공간으로 투영 (Fisher의 LD)

선형 분별

- 문제의 공식화
 - 분별력에 유리한 정도를 수식화 하는 방법?
 - 최적의 축을 찾는 방법?
- 기본 아이디어
 - 같은 부류의 샘플은 모여있고 다른 부류의 샘플은 떨어져 있을수록 유리한다.
 - 부류 간 퍼짐(between-class scatter),
부류 내 퍼짐 (within-class scatter)

$$\mathbf{m}_i = \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{x} \quad y = \mathbf{w}^T \mathbf{x}$$

$$\left. \begin{aligned} \bar{m}_i &= \frac{1}{N_i} \sum_{y \in \omega_i} y \\ &= \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{w}^T \mathbf{x} \\ &= \mathbf{w}^T \mathbf{m}_i \end{aligned} \right\}$$

$$\bar{s}_i^2 = \sum_{y \in \omega_i} (y - \bar{m}_i)^2$$

$$\text{부류내 퍼짐} = \bar{s}_1^2 + \bar{s}_2^2$$

$$\text{부류간 퍼짐} = |\bar{m}_1 - \bar{m}_2| = |\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2| = |\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)|$$



선형 분별

■ 목적 함수 $J(\mathbf{w})$

$$J(\mathbf{w}) = \frac{\text{부류간 페짐}}{\text{부류내 페짐}} = \frac{|\bar{m}_1 - \bar{m}_2|^2}{\bar{s}_1^2 + \bar{s}_2^2} \quad (8.43)$$

▣ $J(\mathbf{w})$ 를 최대화하는 \mathbf{w} 를 찾아라.

■ 문자와 분모를 다시 쓰면,

| | |
|---|---|
| $\begin{aligned}\bar{s}_1^2 + \bar{s}_2^2 &= \sum_{y \in \omega_1} (y - \bar{m}_1)^2 + \sum_{y \in \omega_2} (y - \bar{m}_2)^2 \\ &= \sum_{x \in \omega_1} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mathbf{m}_1)^2 + \sum_{x \in \omega_2} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mathbf{m}_2)^2 \\ &= \sum_{x \in \omega_1} \mathbf{w}^T (\mathbf{x} - \mathbf{m}_1) (\mathbf{x} - \mathbf{m}_1)^T \mathbf{w} + \sum_{x \in \omega_2} \mathbf{w}^T (\mathbf{x} - \mathbf{m}_2) (\mathbf{x} - \mathbf{m}_2)^T \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_W \mathbf{w}\end{aligned}$ | $\begin{aligned} \bar{m}_1 - \bar{m}_2 ^2 &= (\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{m}_2)^2 \\ &= \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2) (\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w} \\ &= \mathbf{w}^T \mathbf{S}_B \mathbf{w}\end{aligned}$ |
|---|---|

▷ $\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2) (\mathbf{m}_1 - \mathbf{m}_2)^T$

▷ $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$, $\mathbf{S}_i = \sum_{x \in \omega_i} (\mathbf{x} - \mathbf{m}_i) (\mathbf{x} - \mathbf{m}_i)^T$



선형 분별

- 목적 함수를 다시 쓰면,

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad (8.48)$$

- $\partial J(\mathbf{w}) / \partial \mathbf{w} = 0$ 으로 두고 풀면,

$$(\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w} = (\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} \quad (8.49)$$

□ (8.49)를 정리하면, $(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \alpha_1 (\mathbf{m}_1 - \mathbf{m}_2)$$

$$\mathbf{S}_W \mathbf{w} = \frac{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}{\mathbf{w}^T \mathbf{S}_B \mathbf{w}} \alpha_1 (\mathbf{m}_1 - \mathbf{m}_2)$$

$$\mathbf{S}_W \mathbf{w} = \alpha_2 \alpha_1 (\mathbf{m}_1 - \mathbf{m}_2)$$

$$\mathbf{w} = \alpha_2 \alpha_1 \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

□ 결국 답은 (즉 구하고자 한 최적의 축은),

$\mathbf{w} = \alpha \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$

←Fisher's linear discriminant (8.50)



선형 분별

■ 예제 8.8 Fisher의 선형 분별

ω_1 샘플 (파랑): $\mathbf{x}_2 = (2,4)^T$, $\mathbf{x}_4 = (4,3)^T$

ω_2 샘플 (검정): $\mathbf{x}_1 = (2,1)^T$, $\mathbf{x}_3 = (4,1)^T$

$$\mathbf{m}_1 = (3, 3.5)^T$$

$$\mathbf{m}_2 = (3, 1)^T$$

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2 = \left[\begin{pmatrix} -1 \\ 0.5 \end{pmatrix} (-1 \quad 0.5) + \begin{pmatrix} 1 \\ -0.5 \end{pmatrix} (1 \quad -0.5) \right] + \left[\begin{pmatrix} -1 \\ 0 \end{pmatrix} (-1 \quad 0) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1 \quad 0) \right] = \begin{pmatrix} 4 & -1 \\ -1 & 0.5 \end{pmatrix}$$

$$\mathbf{S}_W^{-1} = \begin{pmatrix} 0.5 & 1 \\ 1 & 4 \end{pmatrix}$$

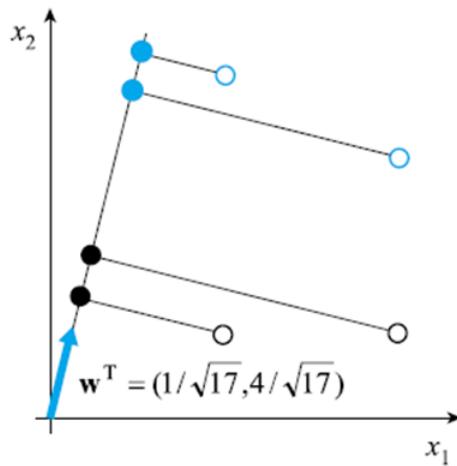
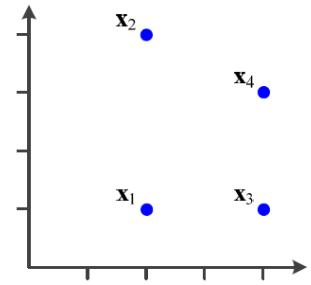


그림 8.16 Fisher의 선형 분별

$$\mathbf{w} = \alpha \begin{pmatrix} 0.5 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} 0 \\ 2.5 \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

$$\mathbf{w} = \left(\frac{1}{\sqrt{17}} \quad \frac{4}{\sqrt{17}} \right)^T = (0.24254, 0.97014)^T$$

이것이 최적의 축이다.
그림 8.15와 비교해 보자.



독립 성분 분석

- ICA, Independent Component Analysis
 - 블라인드 원음 분리 문제 (blind source separation)
 - 실제 세계에서는 여러 신호가 섞여 나타남([그림 6-21]은 음악과 대화가 섞이는 예)

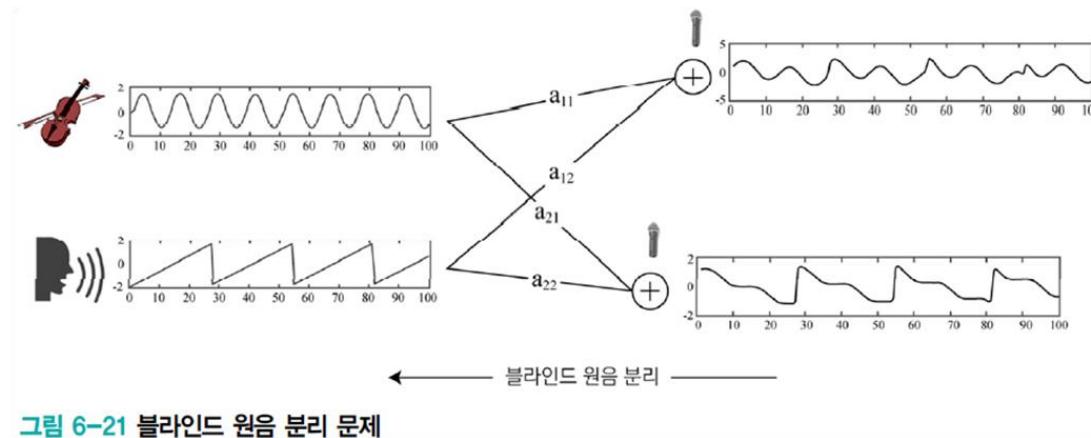


그림 6-21 블라인드 원음 분리 문제

- 마이크로 측정한 혼합 신호로부터 원음(음악과 목소리)을 복원할 수 있나? → 블라인드 원음 분리 문제라 부르며 독립 성분 분석 기법으로 해결 가능
- 아주 많은 예, 뇌파와 다른 장기 신호가 섞인 EEG, 장면과 잡음이 섞인 영상, ...

독립 성분 분석

■ ICA, Independent Component Analysis

■ 문제 정의

▪ 표기

- 원래 신호를 $z_1(t)$ 와 $z_2(t)$, 측정된 혼합 신호를 $x_1(t)$ 와 $x_2(t)$ 로 표기
- t 순간에 획득된 $\mathbf{x}_t = (x_1(t), x_2(t))^T$ 를 훈련 샘플로 취함. 따라서 훈련집합은 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
- 블라인드 원음 분리 문제는 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ 로부터 $\mathbb{Z} = \{z_1, z_2, \dots, z_n\}$ 를 찾는 문제

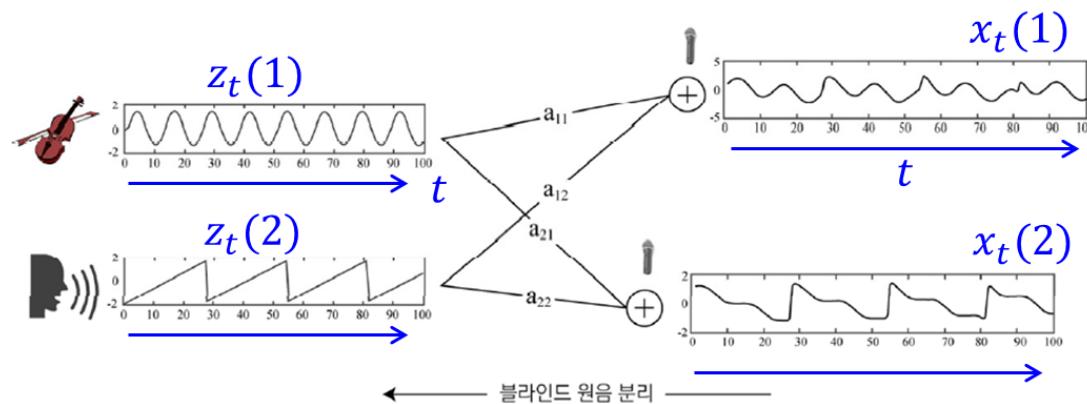


그림 6-21 블라인드 원음 분리 문제

독립 성분 분석

- ICA, Independent Component Analysis

■ 문제 공식화

- 혼합 신호 x 를 원래 신호 z 의 선형 결합으로 표현 가능($z_1(t)$ 와 $z_2(t)$ 가 독립이라는 가정)

$$\begin{aligned} x_1 &= a_{11}z_1 + a_{12}z_2 \\ x_2 &= a_{21}z_1 + a_{22}z_2 \end{aligned} \quad (6.24)$$

- 행렬 표기로 쓰면,

$$\mathbf{x} \equiv \mathbf{A}\mathbf{z} \quad (6.25)$$

- 블라인드 원음 분리 문제란 **A**를 구하는 것. **A**를 알면, 식 (6.26)으로 원음 복원

$$\tilde{\mathbf{z}} = \mathbf{W}\mathbf{x}, \quad \text{or} \quad \mathbf{W} = \mathbf{A}^{-1} \quad (6.26)$$

■ 식 (6.25)는 과소 조건 문제

- 정수 하나를 주고 어떤 두 수의 곱인지 알아내라는 문제와 비슷함 (예를 들어, 32는 1×32 , 2×16 , 4×8 등 여러 답이 가능) ← 추가 조건을 주면 유일 해가 가능
 - 문제도 과소 적합
 - 추가 조건을 이용하여 식 (6.25)의 해를 찾음 → 독립성 가정과 비가우시안 가정

독립 성분 분석

- ICA, Independent Component Analysis

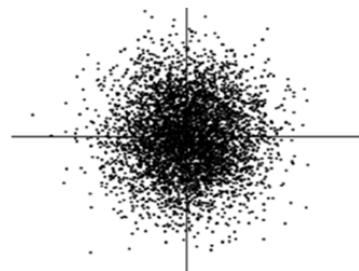
- 독립성 가정

- 원래 신호가 서로 독립이라는 가정(예, 음악과 대화는 서로 무관하게 발생함)

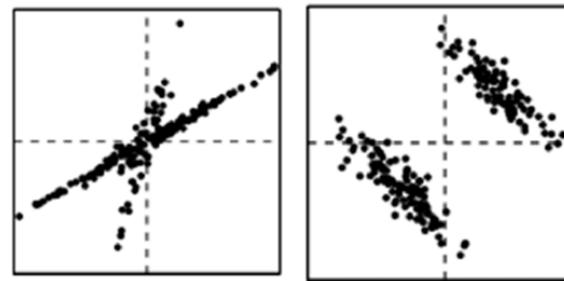
$$P(\mathbf{z}) = P(z_1, z_2, \dots, z_d) = \prod_{j=1}^d P(z_j) \quad (6.27)$$

- 비가우시안 가정

- 원래 신호가 가우시안이라면 혼합 신호도 ([그림 6-22(a)]처럼) 가우시안이 되므로 분리할 실마리 없음. 비가우시안이면 ([그림 6-22(b)]처럼) 실마리가 있음



(a) 확률변수가 가우시안일 때



(b) 확률변수가 비가우시안일 때

그림 6-22 서로 독립인 두 확률변수의 결합 분포

독립 성분 분석

■ ICA, Independent Component Analysis

■ ICA의 문제 풀이

- 원래 신호의 비가우시안인 정도를 최대화하는 가중치를 구하는 전략 사용
 - 원래 신호를 식으로 쓰면,

$$\left. \begin{array}{l} z_j = w_{j1}x_1 + w_{j2}x_2 \\ \text{행렬 형태로 쓰면 } z_j = \mathbf{w}_j \mathbf{x} \end{array} \right\} \quad (6.28)$$

- 비가우시안을 최대화하는 가중치를 구하는 식을 쓰면,

$$\hat{\mathbf{w}}_j = \underset{\mathbf{w}_j}{\operatorname{argmax}} \check{G}(z_j) \quad (6.29)$$

- \check{G} 는 비가우시안 정도를 측정하는 함수
- 주로 식 (6.31)의 첨도를 사용

$$kurtosis(z_j) = \frac{1}{n} \sum_{i=1}^n z_{ji}^4 - 3 \left(\frac{1}{n} \sum_{i=1}^n z_{ji}^2 \right)^2 \quad (6.31)$$

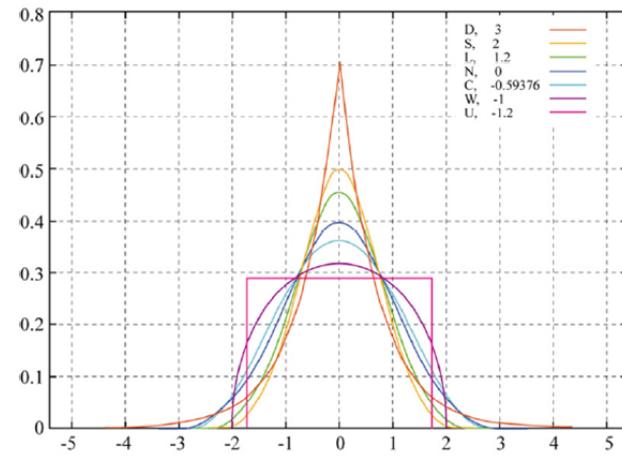


그림 6-23 여러 가지 분포의 첨도 측정

독립 성분 분석

■ ICA, Independent Component Analysis

■ ICA 학습

1. 전처리 수행

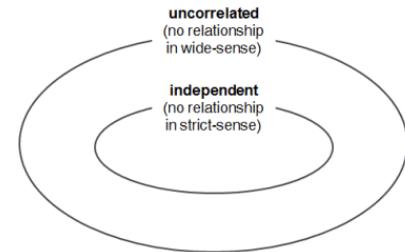
- 훈련집합 \mathbb{X} 의 평균이 $\mathbf{0}$ 이 되도록 이동(식 (6.19) 적용)
- 식 (6.30)의 화이트닝 변환 적용

$$\mathbf{x}'_i = \left(\mathbf{D}^{-\frac{1}{2}} \mathbf{V}^T \right) \mathbf{x}_i, i = 1, 2, \dots, n \quad (6.30)$$

2. 식 (6.29)를 풀어 최적 가중치 구함

■ PCA와 ICA 비교

- ICA는 비가우시안과 독립성 가정, PCA는 가우시안과 비상관을 가정
- ICA로 찾은 축은 수직 아님, PCA로 찾은 축은 서로 수직
- ICA는 주로 블라인드 원음 분리 문제를 푸는데, PCA는 차원 축소 문제를 풁



출처:
<https://iamtaehoon.wordpress.com/2015/01/13/dependence%EC%99%80-correlation%EC%9D%98-%EC%B0%A8%EC%9D%B4/>

독립 성분 분석

- 요약
 - 혼합된 신호 분리
 - 독립성 가정
 - 비 가우시안 가정
 - 전처리 후 최적 가중치 계산

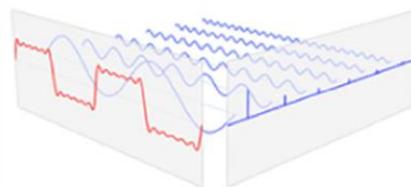
- 실습



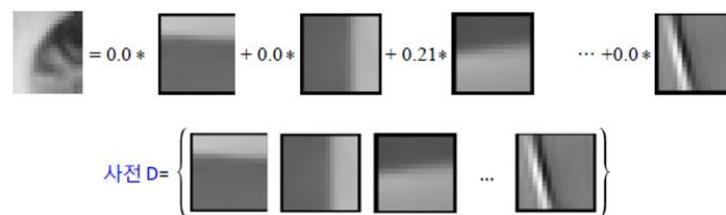
희소 코딩

- (참고) 스파스 코딩, Sparse Coding
 - 기저함수 또는 기저 벡터의 선형 결합으로 신호를 표현
 - 푸리에 변환([그림 6-24(a)]) 또는 웨이블릿 변환 등
 - 희소 코딩
 - 사전 \mathbf{D} 를 구성하는 기저 벡터(단어) $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m$ 의 선형 결합으로 신호(영상) \mathbf{x} 를 표현

$$\mathbf{x} = \mathbf{Da} \\ \textcircled{o} \text{ 때 } \mathbf{D} = (\mathbf{d}_1 \ \mathbf{d}_2 \ \dots \ \mathbf{d}_m) \quad (6.32)$$



(a) 푸리에 변환



희소 코드 $\mathbf{a} = (0.0, 0.0, 0.21, \dots, 0.0)$

(b) 희소 코딩

그림 6-24 신호를 기저함수 또는 기저 벡터의 선형 결합으로 근사 표현

희소 코딩

- 스파스 코딩, Sparse Coding
 - 희소 코딩이 다른 변환 기법과 다른 점
 - 비지도 학습이 사전(즉 기저 벡터)를 자동으로 알아냄(푸리에 변환은 삼각함수를 사용함)
→ 희소 코딩은 데이터에 맞는 기저 벡터를 사용하는 셈
 - 사전의 크기를 과잉 완벽하게 책정($m > d$)
 - 희소 코드 \mathbf{a} 를 구성하는 요소 대부분이 0 값을 가짐
 - 희소 코딩 구현
 - 최적의 사전과 최적의 희소 코드를 알아내야 함
 - ϕ 는 희소 코드의 희소성을 강제하는 규제항

$$\widehat{\mathbf{D}}, \widehat{\mathbf{A}} = \operatorname{argmin}_{\mathbf{D}, \mathbf{A}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda\phi(\mathbf{a}_i) \quad (6.33)$$



특징 활용

- 좋은 특징 추출은 시행 착오가 많이 필요함
 - 외부 요소에 영향을 많이 받음
 - 특징이 좋은 성능을 보이지 못한다면...
 - 특징 결합
 - 특징 전처리



특징 활용

■ 특징 결합

- 특징의 분별력 한계
 - ▣ 그림 8.17은 필기 숫자 예
 - ▣ 그래도 만족스럽다면 그것으로 특징 설계 완료

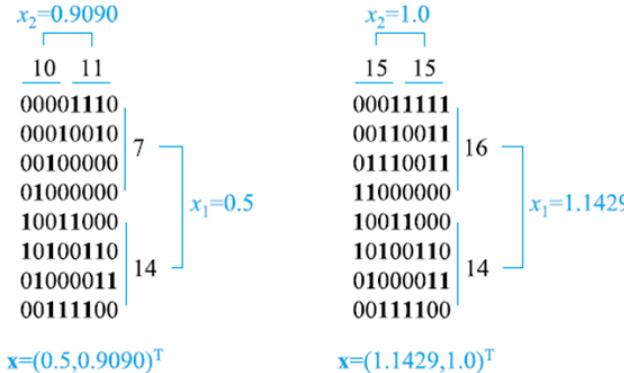


그림 8.17 特징의 성질 (숫자 부류 6)

- 특징이 만족스럽지 않다면,
 - ▣ 버리고 다른 특징을 채택
 - ▣ 또는 기존 특징에 새로운 특징을 추가하는 특징 결합
- 특징이 가지는 정보
 - ▣ 전역 정보
 - 예) 검은 화소 비율
 - ▣ 지역 정보
 - 예) 프로파일

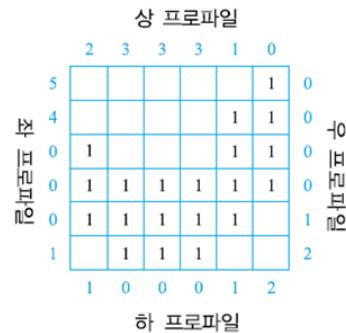
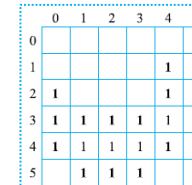


그림 8.7 프로파일 특징

$$R = \{(5,0), (4,1), (5,1), (0,2), (4,2), (5,2), (0,3), (1,3), (2,3), (3,3), (4,3), (5,3), (0,4), (1,4), (2,4), (3,4), (4,4), (1,5), (2,5), (3,5)\}$$

$$C = (0,4) \rightarrow 2 \rightarrow 7 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 6 \rightarrow 6 \rightarrow 5 \rightarrow 5 \rightarrow 4 \rightarrow 3$$



(a) 이진 배열 표현

(b) 집합 표현

(c) 체인 코드 표현

그림 8.3 영역의 표현

특징 활용

■ 특징 전처리

■ 거리 개념이 없는 특징의 변환

- 예) 헐액형을 나타내는 특징 $x \in \{A, B, O, AB\}$
- 거리 개념이 없는 특징 x_i 가 n 개의 값을 갖는다면 x_i 를 $x_{i1}, x_{i2}, \dots, x_{in}$ 으로 확장
- $x_{i1}, x_{i2}, \dots, x_{in}$ 중 하나만 1을 가지고 나머지는 0

■ 특징 값의 정규화

- 선형 변환

$$\tilde{x}_i = low_i + \frac{high_i - low_i}{max_i - min_i} (x_i - min_i) \quad (8.51)$$

- 통계에 의한 변환 (평균은 0, 표준 편차는 1을 가지도록 정규화)

$$\tilde{x}_i = \frac{x_i - \bar{x}_i}{\sigma_i} \quad (8.52)$$



특징 활용

■ 특징 전처리

■ 예제 8.9 특징 정규화

- 사람을 키 (m 단위)와 몸무게 (kg 단위)의 두 개 특징으로 표현

$$\mathbf{a} = (1.60, 70.0)^T, \mathbf{b} = (1.65, 65.5)^T, \mathbf{c} = (1.95, 71.0)^T, \mathbf{d} = (1.68, 72.0)^T$$

- 거리 계산에 따르면 \mathbf{a} 는 \mathbf{b} 보다 \mathbf{c} 에 가깝다.
- 몸무게의 동적 범위가 커서 거리 계산을 주도하기 때문

- (8.51)의 정규화 식을 유도하면,

$$\tilde{x}_1 = 0 + \frac{1-0}{1.95-1.60}(x_1 - 1.60) = \frac{1}{0.35}(x_1 - 1.60)$$

$$\tilde{x}_2 = 0 + \frac{1-0}{72.0-65.5}(x_2 - 65.5) = \frac{1}{6.5}(x_2 - 65.5)$$

- 정규화하고 거리를 계산해 보면,

$$\mathbf{a}' = (0, 0.692)^T, \mathbf{b}' = (0.143, 0)^T, \mathbf{c}' = (1, 0.846)^T, \mathbf{d}' = (0.229, 1)^T$$

$$dist(\mathbf{a}', \mathbf{b}') = \sqrt{(0-0.143)^2 + (0.692-0)^2} = 0.707$$

$$dist(\mathbf{a}', \mathbf{c}') = \sqrt{(0-1)^2 + (0.692-0.846)^2} = 1.012$$



매니폴드 학습

■ 매니폴드

- 고차원 공간에 내재한 저차원 공간
- 자동차 위치를 데이터로 간주하면,
 - $x=(\text{위도}, \text{경도}, \text{고도})^\top$
- 이 3차원(고차원) 공간 데이터를 $x=(\text{기준 점에서의 거리})^\top$ 라는 1차원(저차원) 공간, 즉 매니폴드로 표현할 수 있음
- 보통 매니폴드는 비선형 공간이지만 지역적으로 살피면 선형 구조

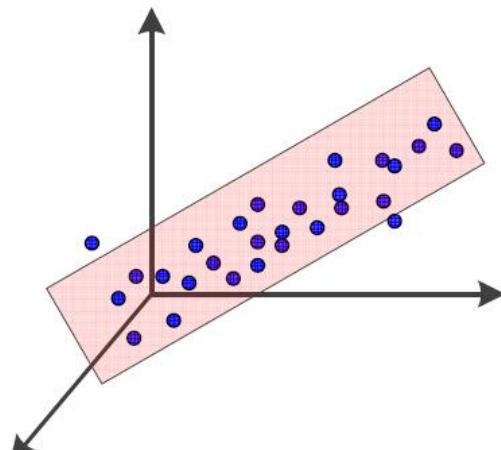


그림 6-30 매니폴드

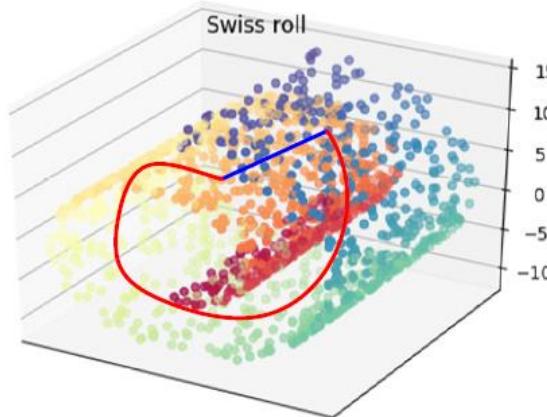
매니폴드 학습

■ 매니폴드 가정

“real-world data presented in high-dimensional spaces are expected to concentrate in the vicinity of a manifold M of much lower dimensionality d_M , embedded in high-dimensional input space \mathbb{R}^d . 고차원 공간에 주어진 실제 세계의 데이터는 고차원 입력 공간 \mathbb{R}^d 에 내재한 훨씬 저차원인 d_M 차원 매니폴드의 인근에 집중되어 있다.”



(a) PCA로 찾은 매니폴드가 적절한 상황



(b) 비선형 매니폴드가 필요한 상황

그림 6-31 매니폴드 가정

■ 매니폴드를 어떻게 찾고 어떻게 표현 할 것인가?

매니폴드 학습

- t-SNE, Stochastic Neighbor Embedding
 - 현재 t-SNE는 매니폴드 공간 변환 기법 중에서 가장 뛰어남

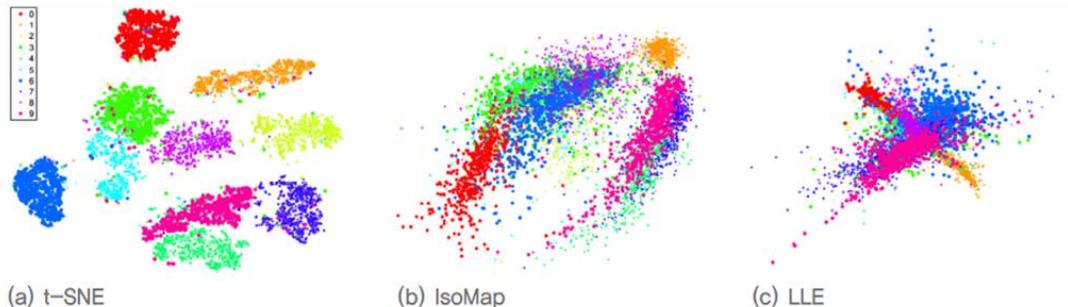


그림 6-33 MNIST를 이용한 매니폴드 학습 기법의 성능 비교

- 원래 공간에서 유사도 측정
 - x_i 와 x_j 의 유사도를 식 (6.45)의 확률로 측정 (가까운 샘플은 높은 확률, 먼 샘플은 0에 가까운 확률 부여)

$$p_{j|i} = \frac{\exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_k\|_2^2}{2\sigma_i^2}\right)} \quad (6.44)$$

$$p_{ij} = p_{ji} = \frac{p_{j|i} + p_{i|j}}{2n} \quad (6.45)$$

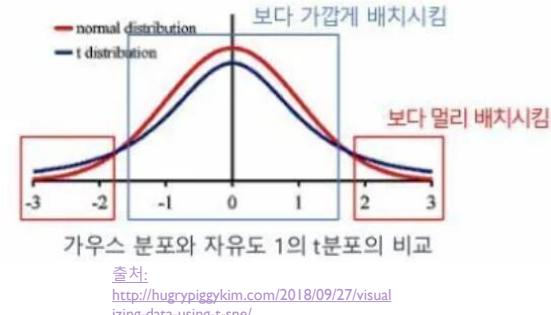
매니폴드 학습

- t-SNE, Stochastic Neighbor Embedding
 - 변환된 공간에서의 유사도는 스튜던트 t 분포로 측정
 - \mathbf{y}_i 와 \mathbf{y}_j 는 변환된 공간에서의 점

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2\right)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|_2^2)^{-1}} \quad (6.46)$$

- 원래 데이터와 변환된 데이터의 구조가 비슷해야 하므로,
 - 확률 분포 P 와 Q 가 비슷할수록 좋음
 - 비슷한 정도를 측정하기 위해 식 (6.47)의 KL 다이버전스를 사용

$$J(\mathbb{X}') = KL(P \parallel Q) = \sum_{i=1}^n \sum_{j=1}^n p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) \quad (6.47)$$



(참고) 정보이론

- 어느 경우가 더 많은 정보를 얻을까?
 - 예) 일기예보
 - 오늘 하루 종일 맑다. 뉴스에서 내일도 맑다고 한다
 - 오늘 하루 종일 맑다. 뉴스에서 내일은 비가 온다고 한다.
- 정보이론
 - 확률이 낮은 사건일수록 더욱 놀랍고 정보량은 크다.
 - 자기 정보 $h(x) = -\log_2 P(x)$
 - Ex) $P(e) = 1/1024, h(e) = 10 \text{ bits}$
 $P(e) = 1, h(e) = 0 \text{ bits}$



(참고) 정보이론

- 엔트로피 (Entropy)
 - 무질서, 불확실성을 나타냄
 - 랜덤 변수 X 가 가질 수 있는 모든 값(사건)에 대한 정보량의 평균

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i).$$

- 모든 사건이 같은 확률을 가질 때 엔트로피가 최대



(참고) 정보이론

- 두 확률 분포간 거리
 - KL 다이버전스

$$KL(P_1(x), P_2(x)) = \sum P_1(x) \log_2 P_1(x) / P_2(x)$$

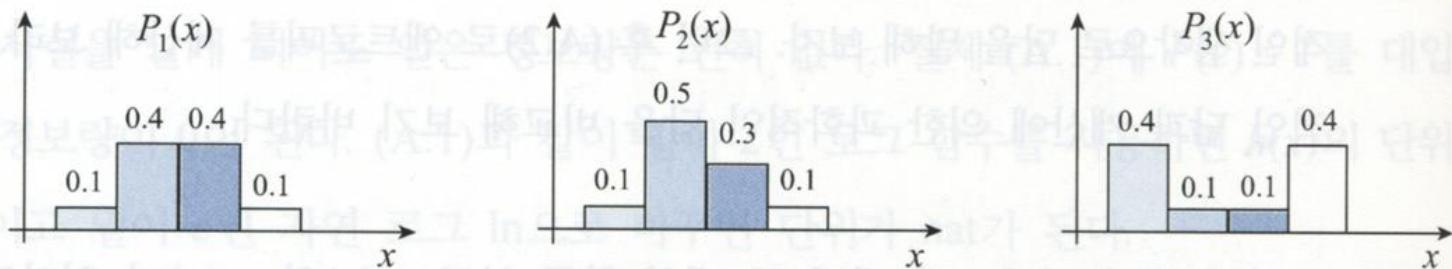


그림 A.2 서로 다른 세 확률 분포

$$KL(P_1(x), P_2(x)) = 0.1 \log_2 \frac{0.1}{0.1} + 0.4 \log_2 \frac{0.4}{0.5} + 0.4 \log_2 \frac{0.4}{0.3} + 0.1 \log_2 \frac{0.1}{0.1} = 0.037$$

$$KL(P_1(x), P_3(x)) = 0.1 \log_2 \frac{0.1}{0.4} + 0.4 \log_2 \frac{0.4}{0.1} + 0.4 \log_2 \frac{0.4}{0.1} + 0.1 \log_2 \frac{0.1}{0.4} = 1.200$$

(참고) 정보이론

- 두 확률 분포간 거리
 - 상호 정보 (Mutual Information)

$$\left. \begin{aligned} I(\mathbf{x}, \mathbf{y}) &= KL(P(\mathbf{x}, \mathbf{y}), P(\mathbf{x})P(\mathbf{y})) \\ &= \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log_2 \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})P(\mathbf{y})} \end{aligned} \right\}$$

- x 와 y 의 의존도 측정 – $P(x), P(y)$
- $P(x, y) = P(x)P(y)$ x, y 는 독립, 의존성이 전혀 없음
- $P(x, y)$ 와 $P(x)P(y)$ 의 차이가 클수록 의존성이 강함 (KL 다이버전스)



매니폴드 학습

■ t-SNE, Stochastic Neighbor Embedding

■ 학습 알고리즘

- 목적함수 J 를 최소로 하는, 즉 P 와 Q 의 KL 다이버전스를 최소로 하는 \mathbb{X}' 를 찾는 문제
- 경사 하강법을 이용(식 (6.48))

$$\frac{\partial J}{\partial \mathbf{y}_i} = 4 \sum_{j=1}^n (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) \left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2\right)^{-1} \quad (6.48)$$

알고리즘 6-5 t-SNE

입력: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 반복 횟수 T , 학습률 ρ , 모멘텀 계수 α

출력: $\mathbb{X}' = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$

- 1 \mathbb{X} 의 모든 샘플 쌍에 대해 식 (6.45)로 p_{ij} 를 계산한다.
- 2 $N(0, 10^{-4}\mathbf{I})$ 가우시안 분포로부터 초기해 $\mathbb{X}'^{(0)} = \{\mathbf{y}_1^{(0)}, \mathbf{y}_2^{(0)}, \dots, \mathbf{y}_n^{(0)}\}$ 을 샘플링한다.
- 3 for ($t=1$ to T)
- 4 식 (6.46)으로 $\mathbb{X}'^{(t-1)}$ 의 모든 쌍에 대해 q_{ij} 를 계산한다.
- 5 for ($i=1$ to n)
- 6 식 (6.48)로 그레이디언트 $\frac{\partial J}{\partial \mathbf{y}_i}$ 를 계산한다.
- 7 if ($t > 1$) $\mathbf{y}_i^{(t)} = \mathbf{y}_i^{(t-1)} + \eta \frac{\partial J}{\partial \mathbf{y}_i} + \alpha(\mathbf{y}_i^{(t-1)} - \mathbf{y}_i^{(t-2)})$ // 학습률과 모멘텀 적용
- 8 else $\mathbf{y}_i^{(t)} = \mathbf{y}_i^{(t-1)} + \eta \frac{\partial J}{\partial \mathbf{y}_i}$

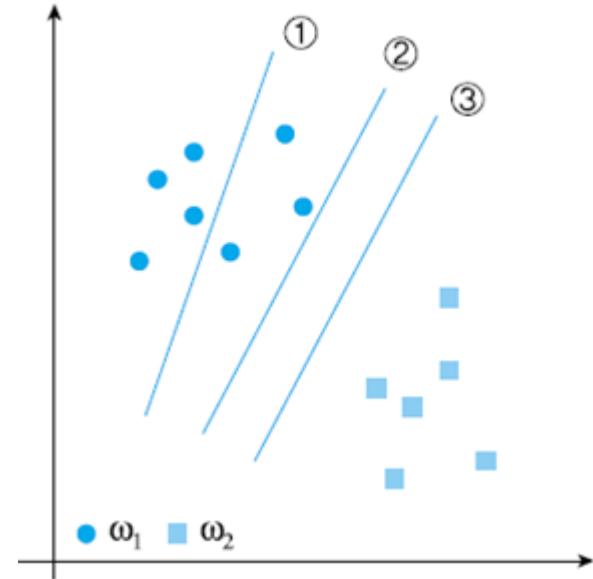


서포트 벡터 머신



SVM 소개

- 서포트 벡터 머신 (Support Vector Machine)
- SVM의 차별성
 - 기존 분류기는 “오류를 최소화”
 - SVM은 여백(margin)을 최대화 하여 일반화 능력의 극대화를 꾀함
- 분류기의 일반화
 - 2보다 3이 여백이 큼
 - 3이 2보다 일반화 능력이 뛰어남
 - 신경망은 초기값 1에서 2를 찾고 멈춤
 - SVM은 3을 찾음
- SVM에 관한 문제
 - 여백이라는 개념 공식화 방법
 - 여백을 최대로 하는 결정 경계 찾는 방법



선형 SVM

- 이진 분류를 위한 결정 초평면(경계) 및 수학적 특성

$$d(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0 \quad (5.1)$$

1. $d(\mathbf{x})$ 는 전체 특징 공간을 두 영역으로 분할하며 한 쪽 영역에 속하는 점 \mathbf{x} 는 $d(\mathbf{x}) > 0$ 이고 다른 쪽에 있는 점은 $d(\mathbf{x}) < 0$ 이다.
2. 하나의 초평면을 표현하는 식은 여럿 있다. (5.1)에 0이 아닌 임의의 상수 c 를 곱하여도 같은 초평면을 나타낸다.
3. \mathbf{w} 는 초평면의 법선 벡터로서 normal vector 초평면의 방향을 나타내고 b 는 위치를 나타낸다.
4. 임의의 점 \mathbf{x} 에서 초평면까지의 거리는 (5.2)와 같다.

$$h = \frac{|d(\mathbf{x})|}{\|\mathbf{w}\|} \quad (5.2)$$

선형 SVM

■ 예제

그림 5.2에 있는 결정 직선의 수학적 특성을 살펴보자. 이 직선의 매개 변수는 $\mathbf{w} = (2,1)^T$ 이고 $b = -4$ 이다.

$$h = \frac{|d(\mathbf{x})|}{\|\mathbf{w}\|}$$

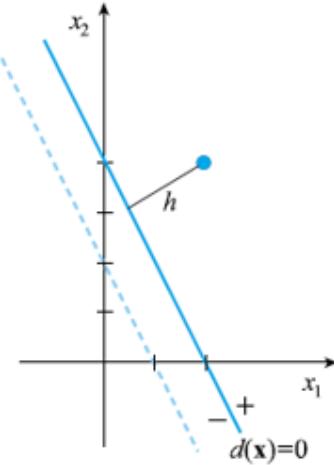


그림 5.2 직선의 수학적 특성

아래는 모두 같은 직선

$$d(\mathbf{x}) = 2x_1 + x_2 - 4 = 0$$

$$d(\mathbf{x}) = x_1 + 0.5x_2 - 2 = 0$$

$$d(\mathbf{x}) = 6x_1 + 3x_2 - 12 = 0$$

점 $\mathbf{x}=(2,4)^T$ 에서 직선까지 거리

$$h = \frac{|2 \times 2 + 1 \times 4 - 4|}{\sqrt{2^2 + 1^2}} = \frac{4}{\sqrt{5}} = 1.78885$$

선형 SVM

- 선형 분리 가능한 상황
 - w (직선의 방향)가 주어진 상황

- ▣ ‘두 부류에 대해 직선으로부터 가장 가까운 샘플까지의 거리가 같게 되는 b 를 결정 (①과 ②는 그렇게 얻은 직선)’
- ▣ **여백**은 그런 직선에서 가장 가까운 샘플까지 거리의 두 배로 정의함
- ▣ 가장 가까운 샘플을 **서포트 벡터**라 부름

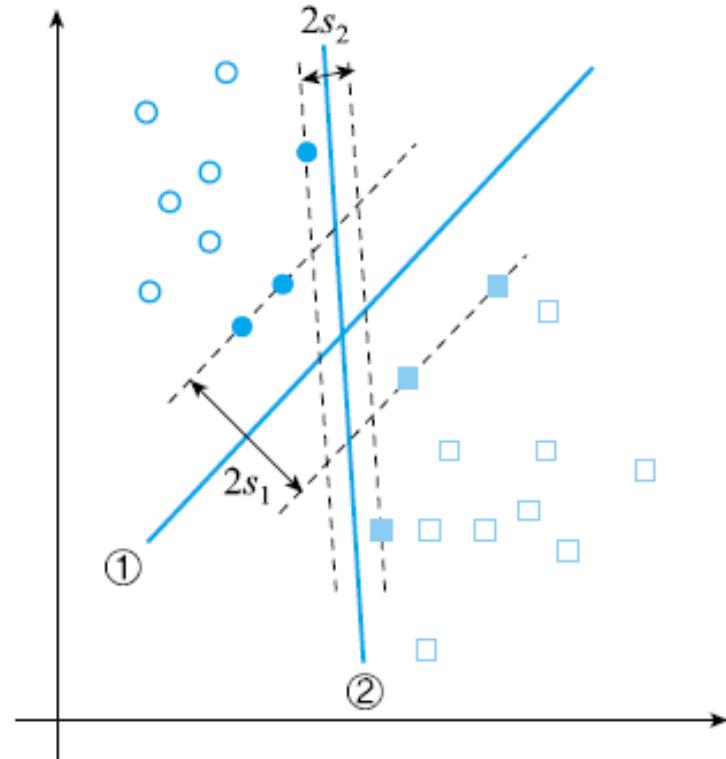


그림 5.3 선형 분리 가능한 상황

선형 SVM

■ 문제 공식화

- 여백을 가장 크게 하는 결정 초평면의 방향, 즉 \mathbf{w} 를 찾아라.

(5.3)²

- 그림 5.3에서 ①과 ②는 어느 것이 최적에 가까운가?
- ①보다 나은 것이 있나?
- 전형적인 최적화 문제임

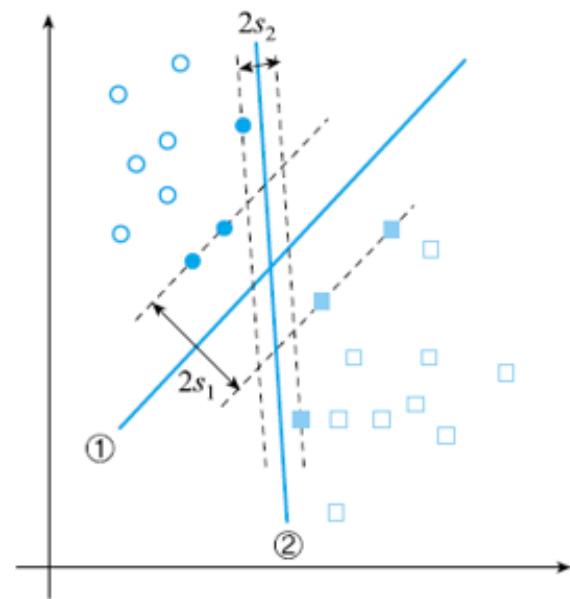


그림 5.3 선형 분리 가능한 상황

선형 SVM

■ 여백의 공식화

$$\text{여백} = 2h = \frac{2|d(\mathbf{x})|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

|이 되도록 적당한 c 를 곱함 (결정 초평면 특성 2번)
이때 \mathbf{x} 는?

■ 조건부 최적화 문제

• 조건부 최적화 문제

아래 조건 하에,

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1, \forall \mathbf{x}_i \in \omega_1$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1, \forall \mathbf{x}_i \in \omega_2$$

$\frac{2}{\|\mathbf{w}\|}$ 를 최대화하라.

훈련 집합 $X=\{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$, $t_i = 1 \text{ or } -1$

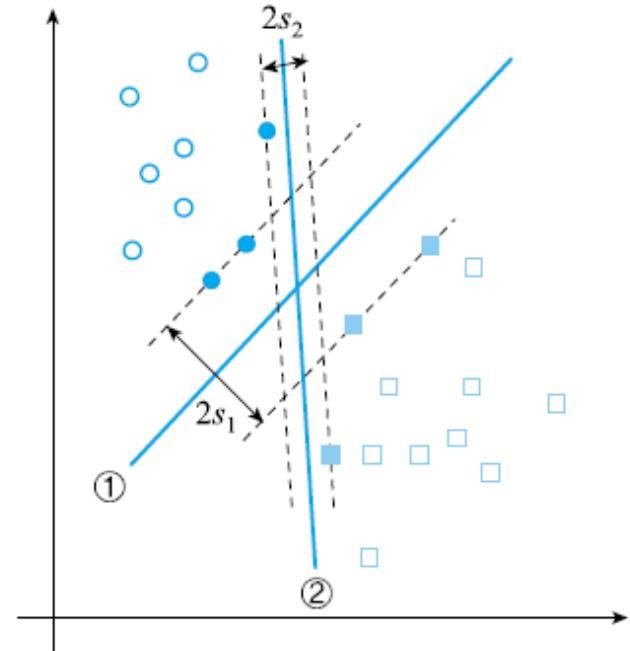


그림 5.3 선형 분리 가능한 상황

선형 SVM

- 조건부 최적화 문제 공식화

- 최소화 문제로 변형

- 조건부 최적화 문제

아래 조건 하에,

$$\left. \begin{array}{l} t_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, i = 1, \dots, N \\ J(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \text{ 을 최소화하라.} \end{array} \right\} \quad (5.6)$$

- 문제의 특성

- 해의 유일성

- 전역 최적 점 보장 (SVM의 장점)

- 문제의 난이도

- N개의 선형 부등식을 조건으로 가진 2차 함수의 최적화 문제

- 조건부 최적화 문제 솔루션: 라그랑제 승수



선형 SVM

- 라그랑제 승수 방법
 - 목적 함수와 조건을 하나의 식(라그랑제 함수 L)으로 만들고 KKT조건을 이용하여 라그랑제 함수를 최적화 하는 해 구함 (α_i 는 라그랑제 승수)

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (t_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1) \quad (5.7)$$

- KKT조건

라그랑제 함수 $L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1,N} \lambda_i f_i(\boldsymbol{\theta})$ 에 대한 KKT 조건은

$$\textcircled{1} \quad \partial L(\boldsymbol{\theta}, \boldsymbol{\lambda}) / \partial \boldsymbol{\theta} = \mathbf{0}$$

$$\textcircled{2} \quad \lambda_i \geq 0, i = 1, \dots, N$$

$$\textcircled{3} \quad \lambda_i f_i(\boldsymbol{\theta}) = 0, i = 1, \dots, N$$



선형 SVM

- 라그랑제 승수 방법

- KKT조건

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} \rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i t_i \mathbf{x}_i \quad (5.8)$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i t_i = 0 \quad (5.9)$$

$$\alpha_i \geq 0, i = 1, \dots, N \quad (5.10)$$

$$\alpha_i (t_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1) = 0, i = 1, \dots, N \quad (5.11)$$

- (5.11)에 의하면 모든 샘플이 $\alpha_i=0$ 또는 $t_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 = 0$ 이어야 함.
 - $\alpha_i \neq 0$ 인 샘플이 서포트 벡터
 - (5.8)에 의하면, 라그랑제 승수 α_i 알면 \mathbf{w} 구할 수 있음
 - 결정 초평면을 구한 셈
 - 이제부터 \mathbf{w} 구하는 대신 라그랑제 승수 구하는 문제로 관심 전환
 - (5.11)로 b 구할 수 있음

선형 SVM

■ Convex 성질을 이용하여 풀기 쉬운 형태로 변환

- ▣ 볼록 성질을 만족하는 조건부 최적화 문제는 Wolfe 듀얼로 변형할 수 있다.

볼록 성질을 만족하는 조건부 최적화 문제는 Wolfe 듀얼 문제로 변형할 수 있다. 원래 문제가 $f_i(\theta) \geq 0, i = 1, \dots, N$ 이라는 조건 하에 $J(\theta)$ 를 최소화하는 것이라 하자. 이때 Wolfe 듀얼 문제는 $\partial L(\theta, \alpha) / \partial \theta = 0$ 과 $\alpha_i \geq 0, i = 1 \dots, N$ 이라는 두 가지 조건 하에 $L(\theta, \alpha) = J(\theta) - \sum_{i=1, N} \alpha_i f_i(\theta)$ 를 최대화하는 것이다. 부등식 조건이 등식 조건으로 바뀌었고 최소화 문제가 최대화 문제로 바뀌었다.

- ▣ (5.6)을 Wolfe 듀얼로 바꾸어 쓰면,

조건부 최적화 문제

$$\left. \begin{array}{l} \text{아래 조건 하에,} \\ t_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, i = 1, \dots, N \\ J(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \text{ 을 최소화하라.} \end{array} \right\}$$



조건부 최적화 문제

$$\left. \begin{array}{l} \text{아래 조건 하에,} \\ \mathbf{w} = \sum_{i=1}^N \alpha_i t_i \mathbf{x}_i \\ \sum_{i=1}^N \alpha_i t_i = 0 \\ \alpha_i \geq 0, i = 1, 2, \dots, N \\ L(\mathbf{w}, b, \alpha) \text{ 를 최대화하라.} \end{array} \right\}$$

- 부등식 조건이 등식 조건이 되어 풀기에 유리함

선형 SVM

■ 수식 정리

■ 5.8, 5.9를 5.7에 대입하여 정리

- 조건부 최적화 문제

아래 조건 하에,

$$\left. \begin{array}{l} \sum_{i=1}^N \alpha_i t_i = 0 \\ \alpha_i \geq 0, i = 1, \dots, N \\ \tilde{L}(\mathbf{a}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j \text{ 를 최대화하라.} \end{array} \right\} \quad (5.13)$$

■ 특성

- 2차 함수의 최대화 문제
- w와 b가 사라지고 alpha를 찾는 문제로 변경
- 특징벡터 \mathbf{x}_i 가 내적 형태로 나타남
- 목적 함수의 두 번째 Σ 는 N^2 개의 항을 갖는다.



선형 SVM

- 예제 I: 두 개의 샘플을 가진 경우의 문제 풀이

- ▣ 훈련집합 $\mathbf{x}_1 = (2,3)^T, t_1 = 1$
 $\mathbf{x}_2 = (4,1)^T, t_2 = -1$

- ▣ (5.13)은

아래 조건 하에,

$$\alpha_1 t_1 + \alpha_2 t_2 = 0$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0$$

$$\tilde{L}(\mathbf{a}) = (\alpha_1 + \alpha_2)$$

$$-\frac{1}{2}(\alpha_1 \alpha_1 t_1 t_1 \mathbf{x}_1^T \mathbf{x}_1 + \alpha_1 \alpha_2 t_1 t_2 \mathbf{x}_1^T \mathbf{x}_2 + \alpha_2 \alpha_1 t_2 t_1 \mathbf{x}_2^T \mathbf{x}_1 + \alpha_2 \alpha_2 t_2 t_2 \mathbf{x}_2^T \mathbf{x}_2)를 최대화하라.$$

- ▣ 실제 값을 대입하면,

아래 조건 하에,

$$\alpha_1 - \alpha_2 = 0$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0$$

$$\tilde{L}(\mathbf{a}) = (\alpha_1 + \alpha_2) - \frac{1}{2}(13\alpha_1^2 + 17\alpha_2^2 - 22\alpha_1\alpha_2)를 최대화하는 \mathbf{a} = (\alpha_1, \alpha_2)^T를 찾아라.$$

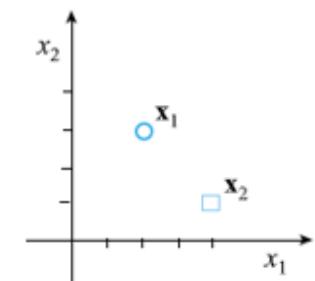
- 조건부 최적화 문제

아래 조건 하에,

$$\sum_{i=1}^N \alpha_i t_i = 0$$

$$\alpha_i \geq 0, i = 1, \dots, N$$

$$\tilde{L}(\mathbf{a}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j 를 최대화하라.$$



(a) 문제

선형 SVM

- 예제 I: 두 개의 샘플을 가진 경우의 문제 풀이

- ▣ 정리하여 풀면,

$$\tilde{L}(\alpha) = -4\alpha_1^2 + 2\alpha_1 = -4\left(\left(\alpha_1 - \frac{1}{4}\right)^2 - \frac{1}{16}\right)$$

$(1/4, 1/4)^T$ 에서 최대값을 가짐

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i t_i \mathbf{x}_i \quad (5.8)$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i t_i = 0 \quad (5.9)$$

$$\alpha_i \geq 0, i = 1, \dots, N \quad (5.10)$$

$$\alpha_i(t_i(\mathbf{w}^T \mathbf{x}_i + b) - 1) = 0, i = 1, \dots, N \quad (5.11)$$

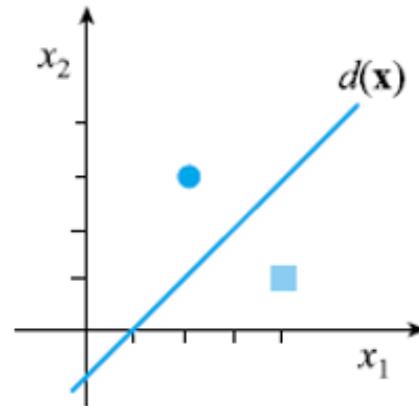
- ▣ (5.8)로 \mathbf{w} , (5.11)로 b 를 구하면

$$\mathbf{w} = \sum_{i=1}^2 \alpha_i t_i \mathbf{x}_i = \frac{1}{4}(2, 3)^T - \frac{1}{4}(4, 1)^T = \left(-\frac{1}{2}, \frac{1}{2}\right)^T$$

$$b = \frac{1}{2}$$

- ▣ 결국 결정 직선은

$$d(\mathbf{x}) = -\frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}$$



(b) SVM (속이 찬 샘플이 서포트 벡터)

선형 SVM

- 예제2: 세 개의 샘플을 가진 경우의 문제 풀이

- ▣ 훈련집합

$$\mathbf{x}_1 = (2,3)^T, t_1 = 1$$

$$\mathbf{x}_2 = (4,1)^T, t_2 = -1$$

$$\mathbf{x}_3 = (5,1)^T, t_3 = -1$$

- ▣ (5.13)에 실제 값을 대입하면,

아래 조건 하에,

$$\alpha_1 - \alpha_2 - \alpha_3 = 0$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$$

$$\tilde{L}(\mathbf{a}) = (\alpha_1 + \alpha_2 + \alpha_3) - \frac{1}{2}(13\alpha_1^2 + 17\alpha_2^2 + 26\alpha_3^2 - 22\alpha_1\alpha_2 - 26\alpha_1\alpha_3 + 42\alpha_2\alpha_3)$$

을
최대화하는 $\mathbf{a} = (\alpha_1, \alpha_2, \alpha_3)^T$ 을 찾아라.

- 이 문제를 어떻게 풀 것인가?

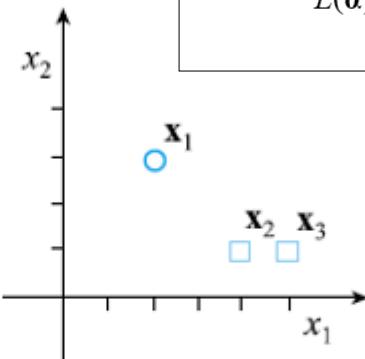
- 조건부 최적화 문제

아래 조건 하에,

$$\sum_{i=1}^N \alpha_i t_i = 0$$

$$\alpha_i \geq 0, i = 1, \dots, N$$

$$\tilde{L}(\mathbf{a}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j$$
 를 최대화하라.



(a) 문제

선형 SVM

- 예제2: 세 개의 샘플을 가진 경우의 문제 풀이

- (review) $\alpha_i \neq 0$ 인 샘플이 서포트 벡터

- ▣ 네 가지 경우로 나누어 분석적 풀이

- ① $\alpha_1=0, \alpha_2 \neq 0, \alpha_3 \neq 0$
 - ② $\alpha_1 \neq 0, \alpha_2=0, \alpha_3 \neq 0$
 - ③ $\alpha_1 \neq 0, \alpha_2 \neq 0, \alpha_3=0$
 - ④ $\alpha_1 \neq 0, \alpha_2 \neq 0, \alpha_3 \neq 0$

- ▣ ① ② ④ 는 모순이므로 버림. 왜 모순?

- ▣ ③ $\alpha_1 \neq 0, \alpha_2 \neq 0, \alpha_3=0$ 인 경우를 풀면,

- 등식 조건으로부터 $\alpha_1=\alpha_2=0$ 이므로

$$\tilde{L}(\alpha) = 2\alpha_1 - 4\alpha_1^2 = -4\left(\left(\alpha_1 - \frac{1}{4}\right)^2 - \frac{1}{16}\right)$$

결국 $\alpha_1=1/4, \alpha_2=1/4, \alpha_3=0$

- (5.8)과 (5.11)로 \mathbf{w} 와 b 를 구하면

아래 조건 하에,

$$\alpha_1 - \alpha_2 - \alpha_3 = 0$$

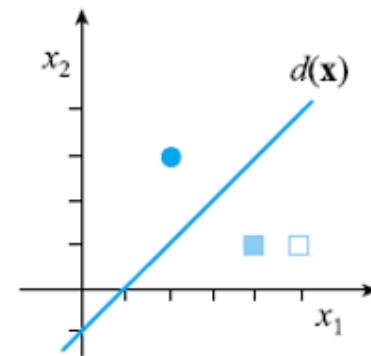
$$\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_3 \geq 0$$

$$\tilde{L}(\alpha) = (\alpha_1 + \alpha_2 + \alpha_3) - \frac{1}{2}(13\alpha_1^2 + 17\alpha_2^2 + 26\alpha_3^2 - 22\alpha_1\alpha_2 - 26\alpha_1\alpha_3 + 42\alpha_2\alpha_3)$$

최대화하는 $\alpha = (\alpha_1, \alpha_2, \alpha_3)^T$ 을 찾아라.

$$\mathbf{w} = \left(-\frac{1}{2}, \frac{1}{2}\right)^T$$

$$b = \frac{1}{2}$$



(b) SVM (속이 찬 샘플이 서포트 벡터)

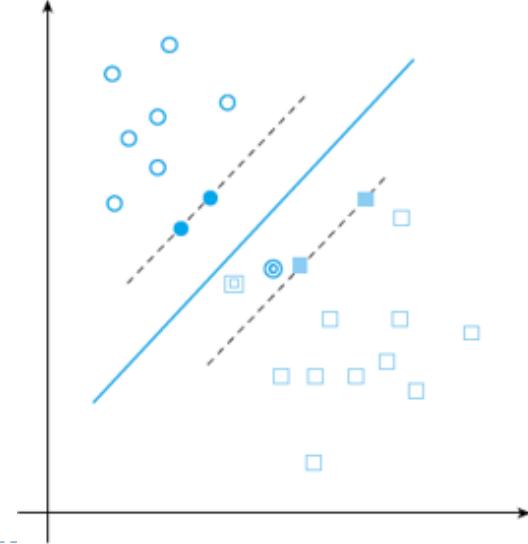
$$d(\mathbf{x}) = -\frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}$$

선형 SVM

■ 선형 분리가 불가능한 상황

▣ 샘플 (\mathbf{x}, t) 의 세 가지 상황

- 경우 1: 분할 띠의 바깥에 있다. $1 \leq t(\mathbf{w}^T \mathbf{x} + b)$ 를 만족한다.
- 경우 2: 분할 띠의 안쪽에 있는데 자기가 속한 부류의 영역에 있다.
 $0 \leq t(\mathbf{w}^T \mathbf{x} + b) < 1$ 을 만족한다.
- 경우 3: 결정 경계를 넘어 자신이 속하지 않은 부류의 영역에 놓여 있다.
 $t(\mathbf{w}^T \mathbf{x} + b) < 0$ 을 만족한다.



선형 SVM

- 선형 분리가 불가능한 상황
 - 슬랙 변수 도입
 - 위의 세 가지 경우를 식 하나로 표현하기 위함

$$t(\mathbf{w}^T \mathbf{x} + b) \geq 1 - \xi$$

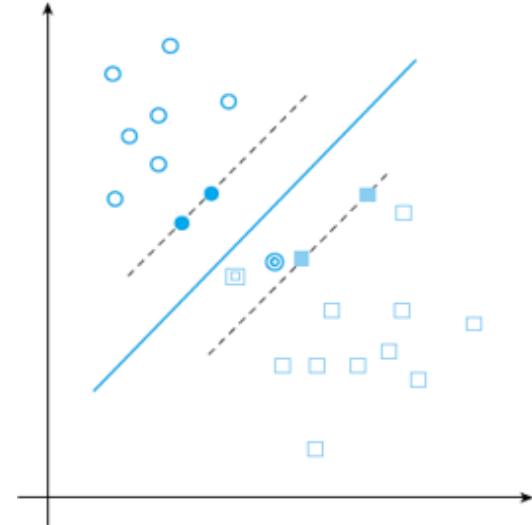


표 5.2 선형 분리 불가능한 상황에서 샘플의 세 가지 경우

| | 샘플 위치 | 분류 | $t(\mathbf{w}^T \mathbf{x} + b)$ 값 | 슬랙 변수 | 그림 5.6에서 기호 |
|------|----------|--------|---|------------------|-------------|
| 경우 1 | 분할 띠 바깥 | 옳게 분류 | $1 \leq t(\mathbf{w}^T \mathbf{x} + b)$ | $\xi = 0$ | □○●■ |
| 경우 2 | 분할 띠 안쪽 | 옳게 분류 | $0 \leq t(\mathbf{w}^T \mathbf{x} + b) < 1$ | $0 < \xi \leq 1$ | □ |
| 경우 3 | 결정 경계 넘음 | 틀리게 분류 | $t(\mathbf{w}^T \mathbf{x} + b) < 0$ | $1 < \xi$ | ◎ |

선형 SVM

- 선형 분리가 불가능한 상황
 - 문제 공식화
 - ▣ 길항^{tradeoff} 관계를 갖는 두 가지 목적을 동시에 달성
 - 여백을 될 수 있는 한 크게 하며 (목적 1), 동시에 $0 < \xi_i$ 인 (즉 경우 2 또는 경우 3에 해당하는) 샘플의 수를 될 수 있는 한 적게 하는 (목적 2) 결정 초평면의 방향 \mathbf{w} 를 찾아라.

- ▣ 목적 함수를 다시 쓰면

$$J(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (5.16)$$

- 첫번째 항은 목적 1, 두번째 항은 목적 2
- C: 사용자 설정 매개 변수 (trade-off)



선형 SVM

- 선형 분리가 불가능한 상황
 - 문제 공식화

- 조건부 최적화 문제

아래 조건 하에,

$$t_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \dots, N$$

$$\xi_i \geq 0, i = 1, \dots, N$$

$$J(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \text{ 를 최소화하라.}$$

} (5.17)



선형 SVM

- 선형 분리가 불가능한 상황
 - 라그랑제 승수로 풀이
 - ▣ 라그랑제 함수

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \right) - \left(\sum_{i=1}^N \alpha_i (t_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) + \sum_{i=1}^N \beta_i \xi_i \right) \quad (5.18)$$

- ▣ KKT 조건

$$\mathbf{w} = \sum_{i=1}^N \alpha_i t_i \mathbf{x}_i \quad (5.19)$$

$$\sum_{i=1}^N \alpha_i t_i = 0 \quad (5.20)$$

$$C = \alpha_i + \beta_i \quad (5.21)$$

$$\alpha_i \geq 0, i = 1, \dots, N \quad (5.22)$$

$$\beta_i \geq 0, i = 1, \dots, N \quad (5.23)$$

$$\alpha_i (t_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) = 0, i = 1, \dots, N \quad (5.24)$$

$$\beta_i \xi_i = 0, i = 1, \dots, N \quad (5.25)$$



선형 SVM

- 선형 분리가 불가능한 상황
 - 라그랑제 승수로 풀이
 - ▣ Wolfe 듀얼로 변형
 - 조건부 최적화 문제

아래 조건 하에,

$$\mathbf{w} = \sum_{i=1}^N \alpha_i t_i \mathbf{x}_i$$

$$\sum_{i=1}^N \alpha_i t_i = 0$$

$$C = \alpha_i + \beta_i$$

$$\alpha_i \geq 0, i = 1, 2, \dots, N$$

$$\beta_i \geq 0, i = 1, 2, \dots, N$$

$L(\mathbf{w}, b, \xi, \alpha, \beta)$ 을 최대화하라.

} (5.26)

선형 SVM

- 선형 분리가 불가능한 상황
 - 라그랑제 승수로 풀이
 - (5.26)을 정리하면
 - 조건부 최적화 문제 (선형 SVM)

아래 조건 하에,

$$\left. \begin{array}{l} \sum_{i=1}^N a_i t_i = 0 \\ 0 \leq a_i \leq C, i = 1, \dots, N \\ \tilde{L}(\alpha) = \sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N a_i a_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j \text{ 를 최대화하라.} \end{array} \right\} \quad (5.27)$$

- (5.13)과 같다! 한 가지만 빼고.
 - $0 \leq a_i$ 가 $0 \leq a_i \leq C$ 로 바뀜

결국 α 만 구하면 w, b 를 구할 수 있음!



선형 SVM

- 선형 분리가 불가능한 상황
 - C의 특징
 - C를 작게 하면 – 여백을 최대한 크게...
 - C를 크게 하면 – 오분류 샘플을 적게...



비선형 SVM

- 비선형 SVM으로 확장
 - 커널을 이용하여 비교적 간단히 해결
- 맵핑 함수를 이용한 분류
 - 더 높은 차원으로 매핑하여 선형 분리 불가능을 가능으로 만들 수 있다.
 - 예제 5.5

원래 공간

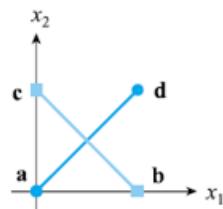
$$\begin{aligned}\mathbf{a} &= (0,0)^T, \quad t_a = 1 \\ \mathbf{b} &= (1,0)^T, \quad t_b = -1 \\ \mathbf{c} &= (0,1)^T, \quad t_c = -1 \\ \mathbf{d} &= (1,1)^T, \quad t_d = 1\end{aligned}$$

매핑 함수

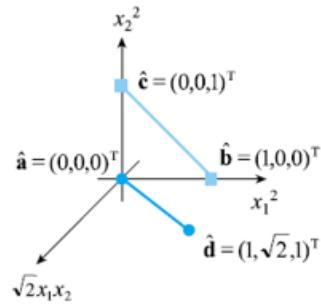
$$\Phi_1(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

매핑 결과

$$\begin{aligned}\mathbf{a} &= (0,0)^T \rightarrow \hat{\mathbf{a}} = (0,0,0)^T \\ \mathbf{b} &= (1,0)^T \rightarrow \hat{\mathbf{b}} = (1,0,0)^T \\ \mathbf{c} &= (0,1)^T \rightarrow \hat{\mathbf{c}} = (0,0,1)^T \\ \mathbf{d} &= (1,1)^T \rightarrow \hat{\mathbf{d}} = (1,\sqrt{2},1)^T\end{aligned}$$



(a) 원래 공간 L



(b) 매핑된 공간 H

그림 5.8 공간 매핑

비선형 SVM

■ 커널 함수

- 공간 매핑 $\Phi: L \rightarrow H$
- SVM이 사용하는 커널 함수

• SVM 커널 함수의 성질 : L 공간 상의 두 벡터 x 와 y 를 매개 변수로 갖는 커널 함수를 $K(x, y)$ 라 하자. 그러면 $K(x, y) = \Phi(x) \cdot \Phi(y)$ 를 만족하는 매핑 함수 $\Phi(\cdot)$ 가 존재해야 한다.⁹ 즉 커널 함수의 값과 H 공간 상으로 매핑된 두 점 $\Phi(x)$ 와 $\Phi(y)$ 의 내적이 같아야 한다.

■ 예제 5.6 커널 함수의 성질

□ 커널 함수 $K(x, y) = (x \cdot y)^2 = x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2$

• 증명

$$\left. \begin{array}{l} K(a, b) = ((0, 0)^T \cdot (1, 0)^T)^2 = 0 \\ \Phi_1(a) \cdot \Phi_1(b) = ((0, 0, 0)^T \cdot (1, 0, 0)^T) = 0 \end{array} \right\} \rightarrow K(a, b) = \Phi_1(a) \cdot \Phi_1(b)$$
$$\left. \begin{array}{l} K(c, d) = ((0, 1)^T \cdot (1, 1)^T)^2 = 1 \\ \Phi_1(c) \cdot \Phi_1(d) = ((0, 0, 1)^T \cdot (1, \sqrt{2}, 1)^T) = 1 \end{array} \right\} \rightarrow K(c, d) = \Phi_1(c) \cdot \Phi_1(d)$$

$$\begin{aligned} K(x, y) &= (x \cdot y)^2 \\ &= x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T \cdot (y_1^2, \sqrt{2}y_1 y_2, y_2^2)^T \\ &= \Phi_1(x) \cdot \Phi_1(y) \end{aligned}$$

$$\Phi_1(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{pmatrix}$$

비선형 SVM

■ (참고) 커널 대치 (커널 트릭)

- ▣ 어떤 수식이 벡터 내적을 포함할 때, 그 내적을 커널 함수로 대치하여 계산하는 기법
 - 실제 계산은 L 공간에서 K()의 계산으로 이루어짐
 - 고차원 공간 H에서 작업하는 효과
 - 적용 예, Fisher LD의 커널 LD로의 확장, PCA를 커널 PCA로 확장
- ▣ SVM에 적용 가능
 - (5.13)과 (5.27)에 벡터 내적만 나타나기 때문
 - 실제 계산은 L에서 이루어지지만 분류는 선형 분류에 유리한 H에서 수행
 - 괴를 부려 차원의 저주를 피한 셈

조건부 최적화 문제

아래 조건 하에,

$$\sum_{i=1}^N \alpha_i t_i = 0$$

$$\alpha_i \geq 0, i = 1, \dots, N$$

$$\tilde{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j \text{ 를 최대화하라.}$$

조건부 최적화 문제 (선형 SVM)

아래 조건 하에,

$$\sum_{i=1}^N \alpha_i t_i = 0$$

$$0 \leq \alpha_i \leq C, i = 1, \dots, N$$

$$\tilde{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\top \mathbf{x}_j \text{ 를 최대화하라.}$$

(5.13)

(5.27)

비선형 SVM

- 커널 대치를 이용한 비선형 SVM

- ▣ 선형 분류가 부적절하면 L 대신 H 공간에서 분류 (그림 5.8)

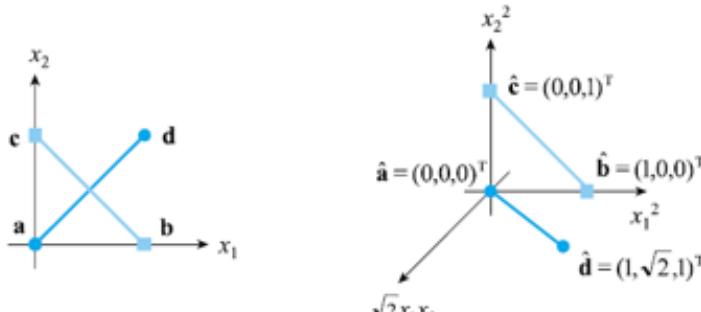


그림 5.8 공간 매핑

- ▣ (5.13)과 (5.27)의 목적 함수를 바꾸어 씀

$$\tilde{L}(\mathbf{a}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \quad (5.30)$$

$$\tilde{L}(\mathbf{a}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (5.31)$$

비선형 SVM

- 커널 대치를 이용한 비선형 SVM

- SVM이 사용하는 대표적인 커널들

다항식 커널 $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$ (5.32)

RBF (Radial Basis Function) 커널 $K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$ (5.33)

하이퍼볼릭 탄젠트 커널 $K(\mathbf{x}, \mathbf{y}) = \tanh(\alpha \mathbf{x} \cdot \mathbf{y} + \beta)$ (5.34)



SVM

■ 정리

표 5.1 조건부 최적화 문제의 변형 과정

| 문제 | | 특성 |
|---------------|-------------------|---|
| 선형 분리 가능 형 | SVM 철학 (그림 5.1) | 여백을 최대로 하여 일반화 능력 극대화 |
| | 문제 (5.3) | SVM 목적과 매개 변수의 관계 설정 |
| | 조건부 최적화 문제 (5.5) | 조건이 등장함 여백을 공식화하여 목적 함수로 삼음 최대화 문제 |
| | 조건부 최적화 문제 (5.6) | t_i 를 이용하여 조건을 하나로 씀 목적 함수 $\frac{2}{\ \mathbf{w}\ }$ 를 $\frac{1}{2}\ \mathbf{w}\ ^2$ 로 바꾸어 최소화 문제로 변형 |
| | 조건부 최적화 문제 (5.12) | 라그랑제 승수를 도입하여 라그랑제 함수 유도함 문제의 볼록 성질을 이용하여 Wolfe 듀얼 문제로 변형 라그랑제 함수를 최대화하는 문제가 됨 부등식 조건이 등식 조건으로 바뀜 |

SVM

■ 정리

| | | |
|-----------|-------------------|---|
| | 조건부 최적화 문제 (5.13) | 라그랑제 함수를 정리하여 w 와 b 가 없어지고 α 만 남음 w 와 b 를 구하는 문제가 α 를 구하는 문제로 바뀜 특징 벡터가 내적 형태로 나타나 비선형 SVM으로 확장하는 토대가 됨 |
| 선형 분리 불가능 | 조건부 최적화 문제 (5.27) | (5.13)을 선형 분리 불가능한 경우로 확장함 $0 \leq \alpha$ 조건이 $0 \leq \alpha \leq C$ 로 변경된 것 빼고는 (5.13)과 같음 |
| 비선형 | 조건부 최적화 문제 (5.35) | (5.27)을 비선형 SVM으로 확장 벡터 내적을 커널 함수 계산으로 대치 (커널 대치 기법) |

SVM

- 실습



분류기 평가

- 분류기 성능

Confusion Matrix

| | | Predicted | |
|--------|----------|------------------------|------------------------|
| | | Positive | Negative |
| Actual | Positive | TP (True Positive) | FN (False Negative) |
| | Negative | FP (False Positive) | TN (True Negative) |

- True Positive(TP) : 실제 True인 정답을 True라고 예측 (정답)
- False Positive(FP) : 실제 False인 정답을 True라고 예측 (오답)
- False Negative(FN) : 실제 True인 정답을 False라고 예측 (오답)
- True Negative(TN) : 실제 False인 정답을 False라고 예측 (정답)

출처: <http://here.deepplus.co.kr/?p=24>

분류기 평가

■ 분류기 성능

1. 정확도(Accuracy)

- 모델이 샘플을 정확히 분류한 비율
- 테스트 데이터가 불균형하거나 특정 클래스의 성능에 편중될 경우 좀 더 세밀한 평가지표가 필요함

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

2. 정밀도(Precision)

- 모델이 긍정으로 분류한 샘플 중 실제 긍정인 비율
- Positive 정답률
- PPV(Positive Predictive Value)

$$Precision = \frac{TP}{TP + FP}$$

3. 재현율(Recall)

- 실제 긍정 샘플 중 모델이 긍정으로 분류한 비율
- Positive 검출률
- sensitivity, hit rate, or TPR(True Positive Rate)

$$Recall = \frac{TP}{TP + FN}$$

4. F1 score

- 정밀도와 재현율의 조화 평균
- 평가를 단일 숫자로 요약. but, 모델의 동작 방식을 잘 이해하려면 정밀도와 재현율을 함께 확인하는 것이 필요

$$F_1 = 2 \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$



인공 신경망



배경

- 뇌의 정보처리 방법 모방
 - 지능 컴퓨터에 도전
 - 인공 신경망(ANN, Artificial Neural Network)
 - 1940대 개발 (디지털 컴퓨터와 탄생 시기가 유사함)
 - 인간 지능에 필적하는 컴퓨터 개발이 목표
- 컴퓨터와 두뇌의 비교
 - 폰 노이만 컴퓨터
 - 순차 명령어 처리기
 - 두뇌
 - 뉴런으로 구성
 - 고도의 병렬 명령어 처리기

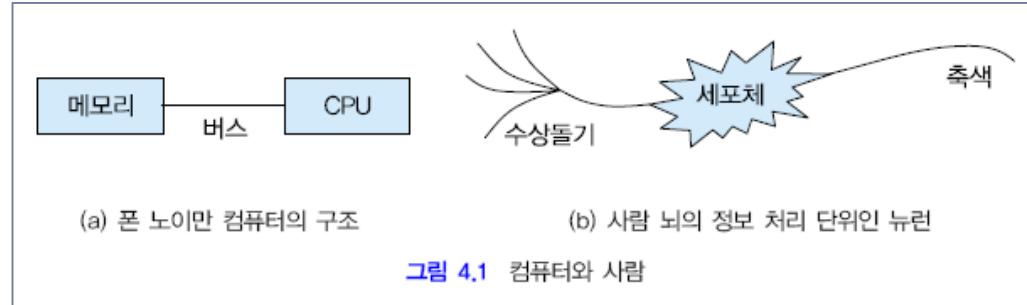


그림 4.1 컴퓨터와 사람

배경

■ 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 역전
파 학습 알고리즘
 - 필기 숫자 인식같은 복잡하고 실용적인 문제에 높은 성능
 - 신경망 연구 다시 활기 찾음
 - 현재 가장 널리 활용되는 문제 해결 도구



수학적 모델의 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)
 - 딥러닝으로 발전



수학적 모델의 신경망

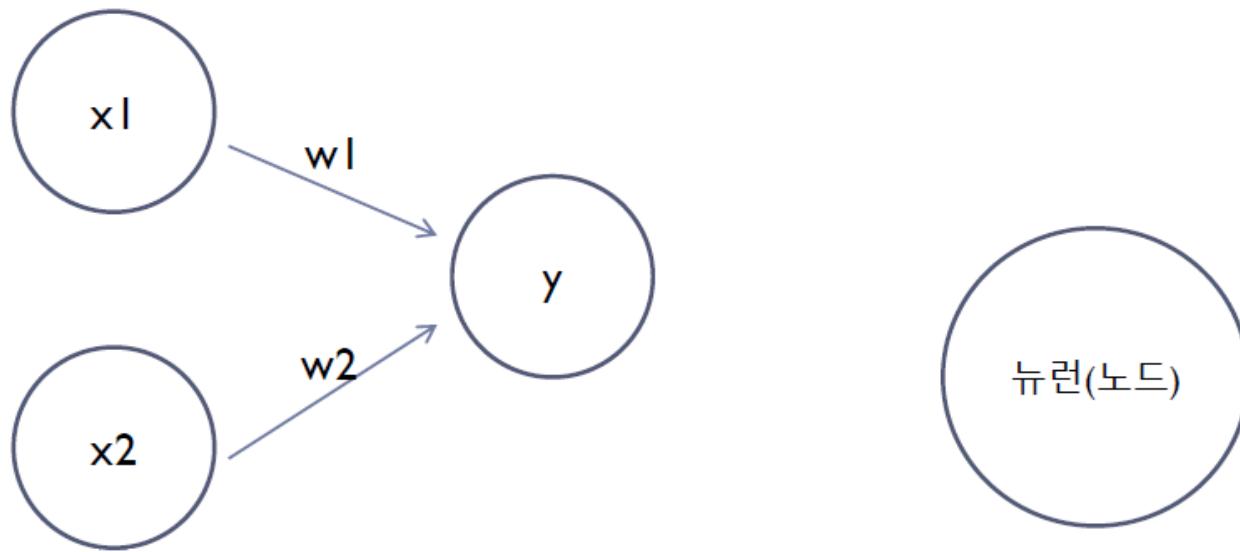
- 새로운 개념들 등장
 - 층
 - 노드와 가중치
 - 학습
 - 활성 함수
- MLP의 초석이 됨



퍼셉트론

- 개념

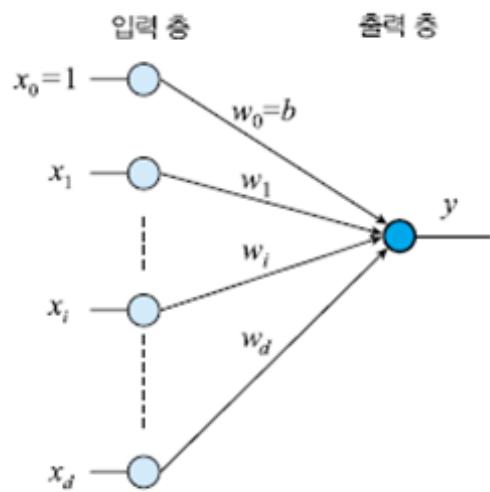
- 다수의 신호를 입력으로 받아 하나 혹은 여러 신호 출력
 - 신호는 흐름을 가짐
 - 신호의 총합이 임계치를 넘으면 1, 아니면 0 출력
 - 가중치는 해당 신호의 중요한 정도



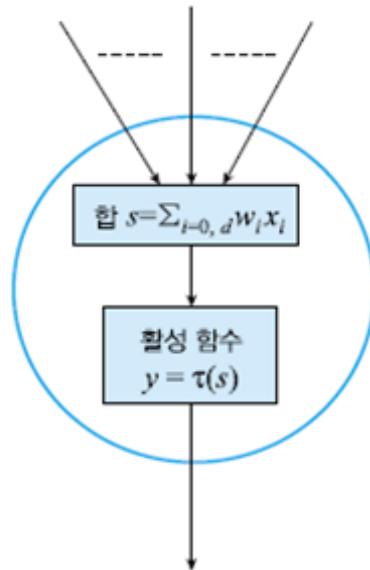
퍼셉트론

■ 구조

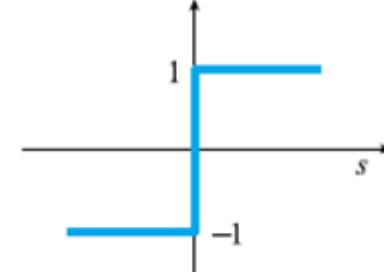
- 입력층: $d+1$ 개의 노드 (특정 벡터 $\mathbf{x}=(x_1, \dots, x_d)^T$)
- 출력층: 한 개의 노드 (따라서 2-부류 분류기)
- 에지와 가중치



(a) 전체 구조



(b) 출력 노드의 연산



(c) 활성 함수

그림 4.2 퍼셉트론의 구조

퍼셉트론

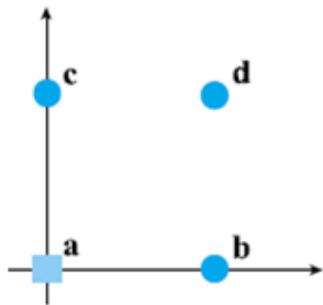
■ 예제

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

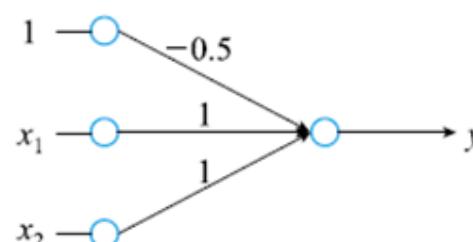
$$\mathbf{b} = (1,0)^T, t_b = 1$$

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

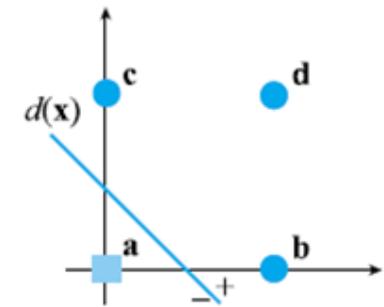
$$\mathbf{d} = (1,1)^T, t_d = 1$$



(a) OR 분류 문제



(b) OR 분류기로서 퍼셉트론



(c) 퍼셉트론은 선형 분류기

그림 4.3 퍼셉트론의 예

0| 퍼셉트론은 $\mathbf{w}=(1,1)^T, b=-0.5$

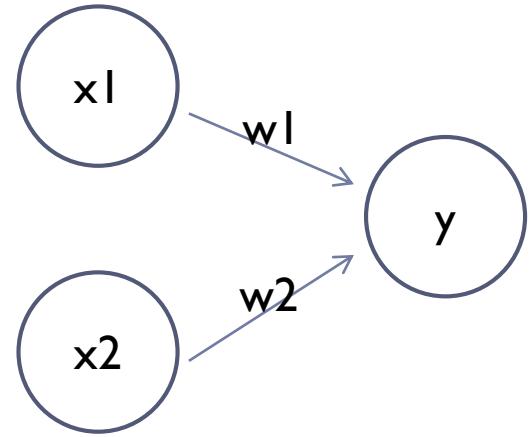
따라서 결정 직선은 $d(\mathbf{x}) = x_1 + x_2 - 0.5$

- 샘플 **c**를 인식해 보자. 맞추나? $y = \tau(\mathbf{w}^T \mathbf{c} + b) = \tau((1,1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 0.5) = \tau(0.5) = 1$
- 나머지 **a**, **b**, **d**는?



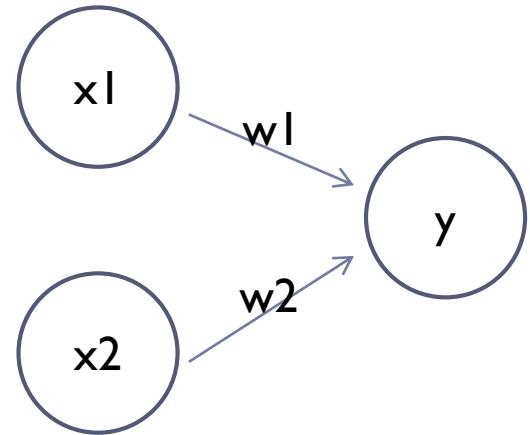
단순 논리 회로

- AND 게이트
 - y 가 참인 경우?
 - 거짓인 경우?
 - AND를 만족하는 매개변수 쌍
 - $(w1=0.5, w2=0.5, \theta=0.7)$
 - $(0.5, 0.5, 0.8)$
 - $(1.0, 1.0, 1.0)$
 -



단순 논리 회로

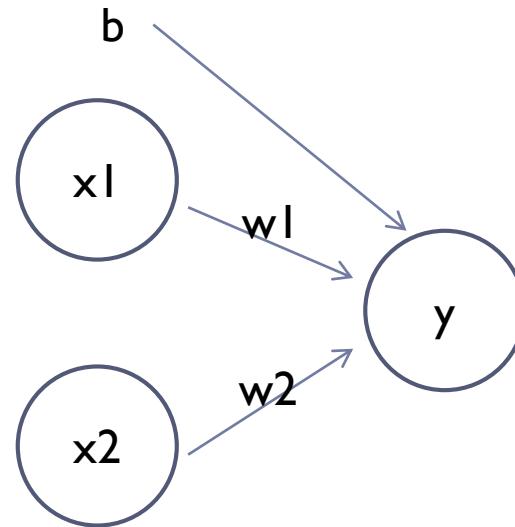
- NAND 게이트
 - y 가 참/거짓인 경우?
 - NAND를 만족하는 매개변수 쌍
 - ($w1 = -0.5, w2 = -0.5, \text{theta} = -0.7$)
 -
- OR 게이트
 - y 가 참/거짓인 경우?
 - OR를 만족하는 매개변수 쌍
 - ($w1 = 1.0, w2 = 1.0, \text{theta} = 0.5$)
 -



실습

- 퍼셉트론 구현하기

- AND
- NAND
- OR
- XOR?



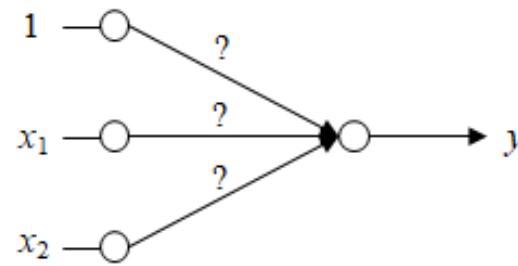
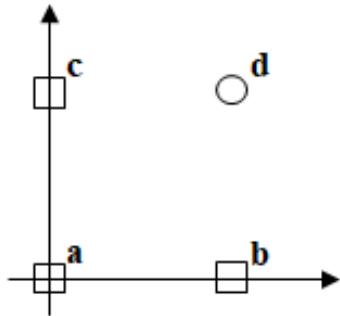
퍼셉트론

■ 퍼셉트론 학습

- 퍼셉트론 학습이란? 훈련 집합 $X = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$ 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_i 는 부류 표지로서 $\mathbf{x}_i \in \omega_1$ 이면 $t_i = 1$ 이고 $\mathbf{x}_i \in \omega_2$ 이면 $t_i = -1$ 이다. X 는 선형 분리 가능하다고 가정한다.³

▣ 예) AND 분류 문제

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



퍼셉트론

■ 노드의 연산

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

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

■ 퍼셉트론은 선형 분류기

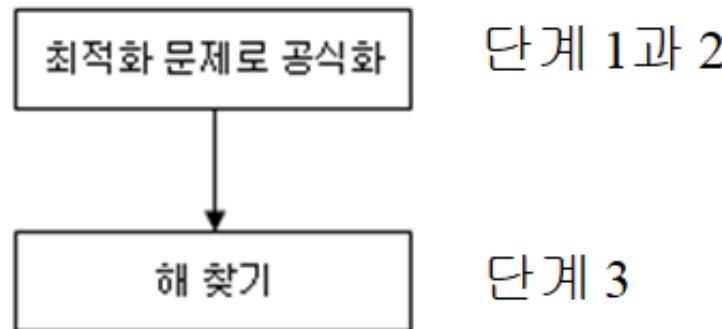
$$\left. \begin{array}{l} d(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b > 0 \text{ 이면 } \mathbf{x} \in \omega_1 \\ d(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b < 0 \text{ 이면 } \mathbf{x} \in \omega_2 \end{array} \right\} \quad (4.3)$$



퍼셉트론 학습

■ (Review) 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 Θ 를 찾는 알고리즘 설계



퍼셉트론 학습

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

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

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

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



퍼셉트론 학습

■ 단계 3

- ▣ $J(\Theta)=0$ 인 Θ 를 찾아라.
- ▣ 내리막 경사법 (Gradient descent method)
 - 현재 해를 $-\frac{\partial J}{\partial \Theta}$ 방향으로 이동
 - 학습률 ρ 를 곱하여 조금씩 이동

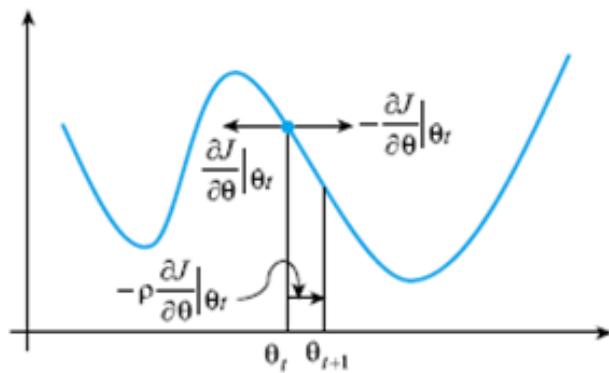
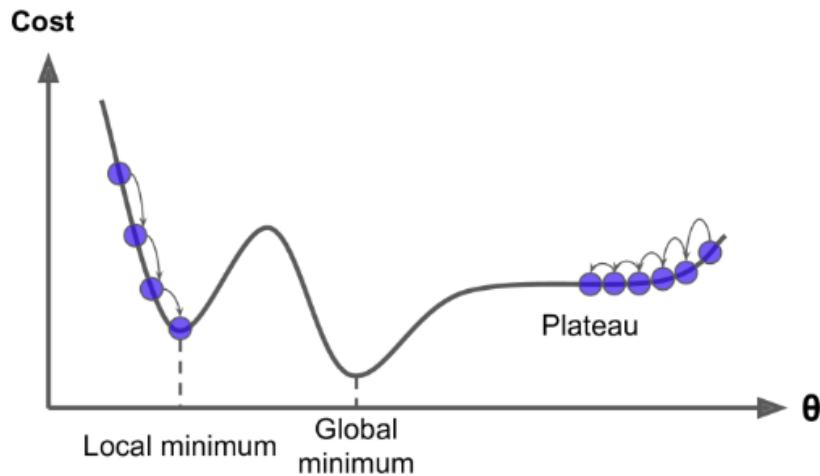


그림 11.9 내리막 경사법

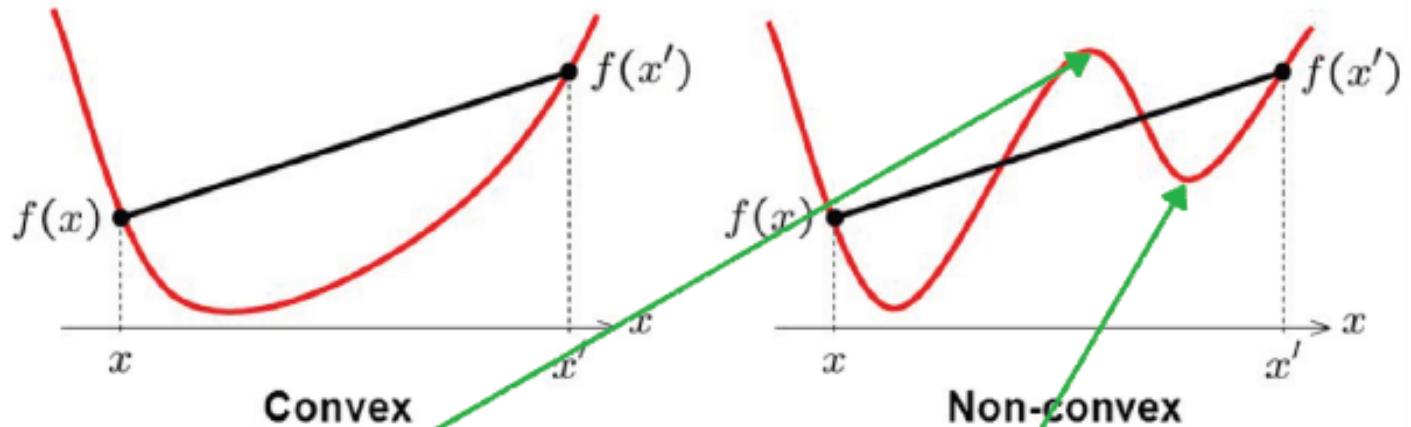
경사하강법 (Gradient Descent)

- 경사도(기울기)
 - 모든 변수의 편미분을 벡터로 정리한 것
 - 경사도가 가리키는 쪽은 각 장소에서 함수의 출력 값을 가장 줄이는 방향임
- 경사하강법
 - 경사도를 이용하여 손실함수의 최소값을 찾는 방법



경사하강법 (Gradient Descent)

■ Convex Function



- 미분값이 0이라고 무조건 최솟값은 아닙니다.
- **최댓값**일 수도 있습니다.
- 최솟값은 아니지만 미분 값이 0인 곳을 **Local Minimum**이라고 합니다.



경사하강법 (Gradient Descent)

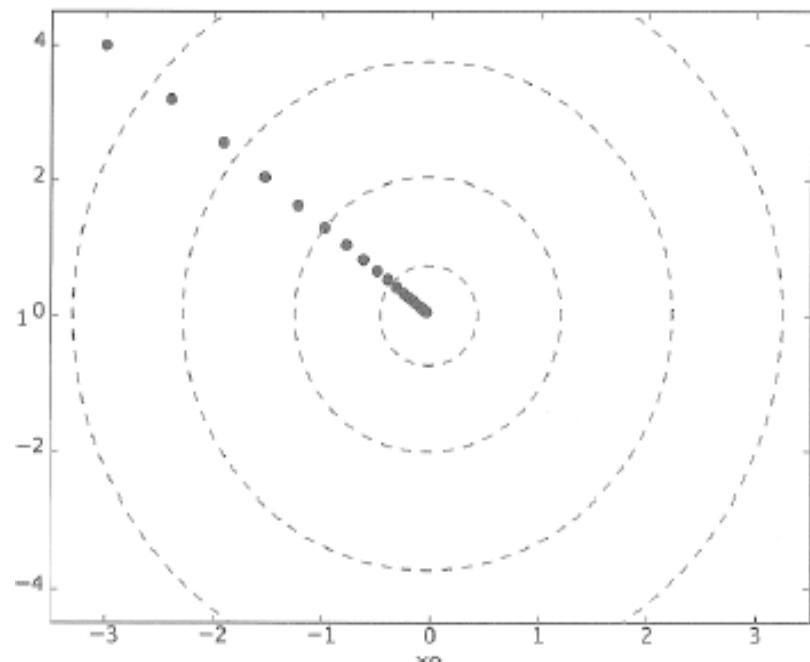
- 개념

- 현 위치에서 기울어진 방향으로 일정 거리만큼 이동
- 이동한 거리에서 다시 경사도 구하고 이동
- 위의 과정을 반복

$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0}$$

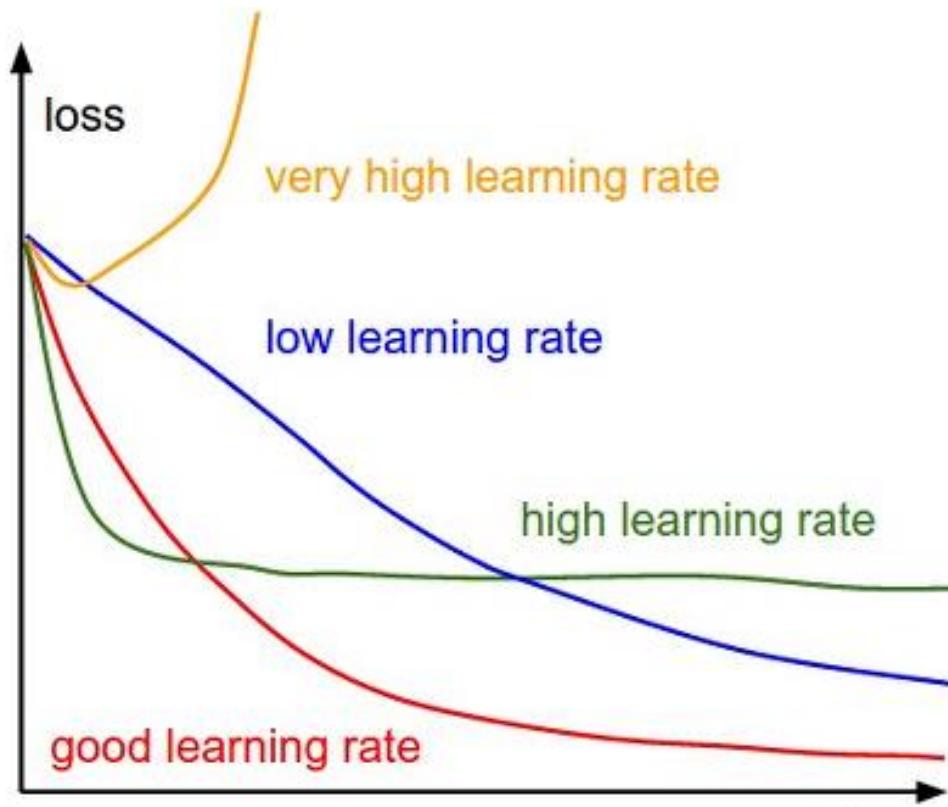
$$x_1 = x_1 - \eta \frac{\partial f}{\partial x_1}$$

- 에타: 학습률(learning rate)
- hyper-parameter



경사하강법 (Gradient Descent)

- 개념
 - 학습률(learning rate)



출처: <https://seamless.tistory.com/38> epoch

퍼셉트론 학습

- 전체 알고리즘
 - 초기해 설정
 - 멈춤 조건이 만족될 때까지 현재 해를 조금씩 이동

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

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

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



퍼셉트론

알고리즘 [4.1]

퍼셉트론 학습 (배치 모드 batch mode)

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

출력: 퍼셉트론 가중치 \mathbf{w} , b

알고리즘:

1. \mathbf{w} 와 b 를 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 b 를 저장한다.



퍼셉트론

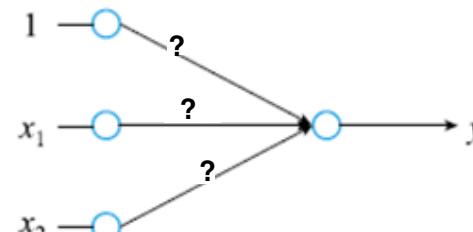
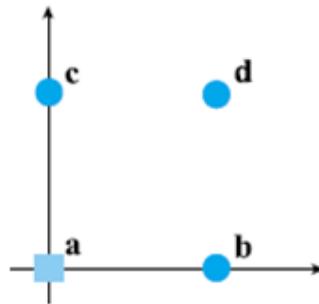
■ 예제 - 학습

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

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

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

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



- $\rho = 0.4$ 로 설정
- 초기값은 $w(0) = (-0.5, 0.75)^T, b(0) = 0.375$
- $d(x) = -0.5x_1 + 0.75x_2 + 0.375$
- $Y(\text{오분류 샘플}) = \{a, b\}$
- 가중치 갱신

} 반복

$$8. \quad \mathbf{w} = \mathbf{w} + \rho \sum_{x_k \in Y} t_k \mathbf{x}_k ;$$

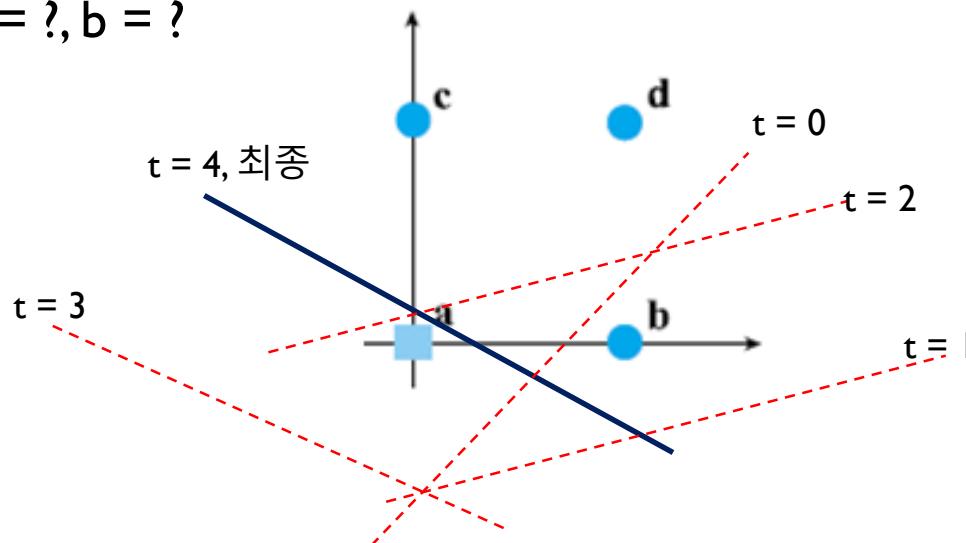
$$9. \quad b = b + \rho \sum_{x_k \in Y} t_k ;$$



퍼셉트론

■ 예제 - 학습

- $w(1) = ?, b(1) = ?, Y = ?$
- $w(2) = ?, b(2) = ?, Y = ?$
- $w(3) = ?, b(3) = ?, Y = ?$
- $w(4) = ?, b(4) = ?, Y = ?$
-
- 최종 $w = ?, b = ?$



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 ρ

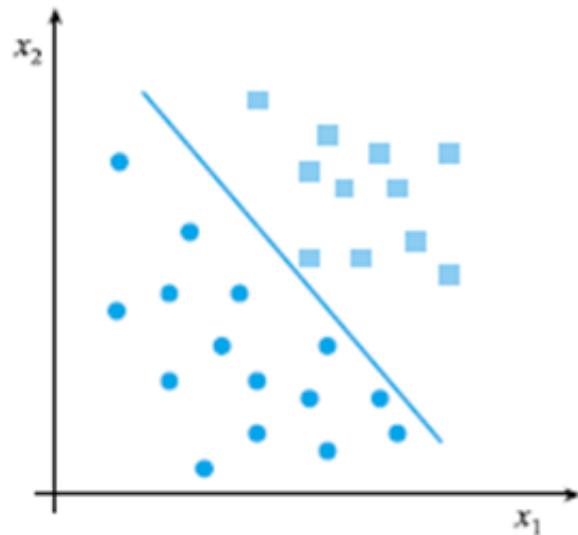
출력: 퍼셉트론 가중치 w, b

알고리즘:

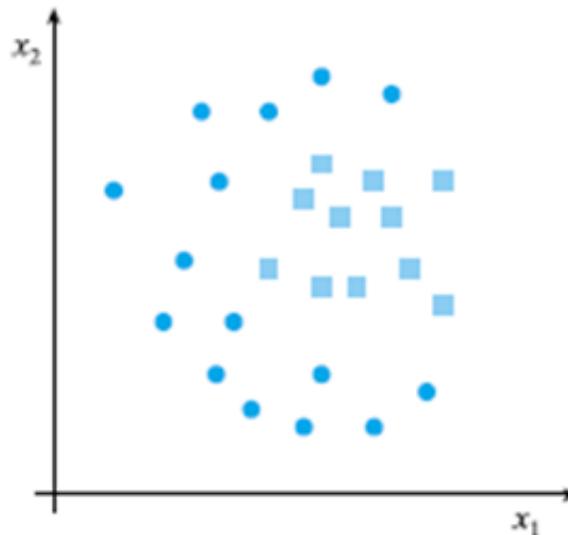
```
1.  $w$ 와  $b$ 를 초기화한다.  
2. repeat {  
3.    $Y = \emptyset$ ;  
4.   for ( $i = 1$  to  $N$ ) {  
5.      $y = \tau(w^T x_i + b)$ ; // (4.2)로 분류를 수행함  
6.     if ( $y \neq t_i$ )  $Y = Y \cup x_i$ ; // 오분류된 샘플 수집  
7.   }  
8.    $w = w + \rho \sum_{x_k \in Y} t_k x_k$ ; // (4.7)로 가중치 갱신  
9. 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

다중 퍼셉트론

- 선형 분리 불가능한 상황
 - ▣ 퍼셉트론의 한계



(a) 선형 분리 가능

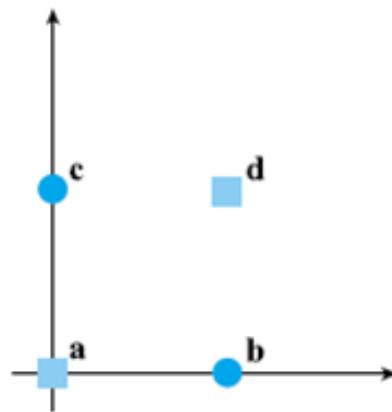


(b) 선형 분리 불가능

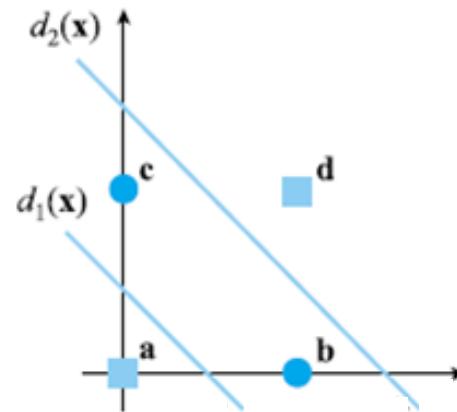
그림 4.5 선형 분리 가능과 불가능

다중 퍼셉트론

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



(a) XOR 분류 문제



(b) 두 개의 직선으로 해결

그림 4.6 XOR 분류 문제의 해결

다중 퍼셉트론

- 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 (\mathbf{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 |

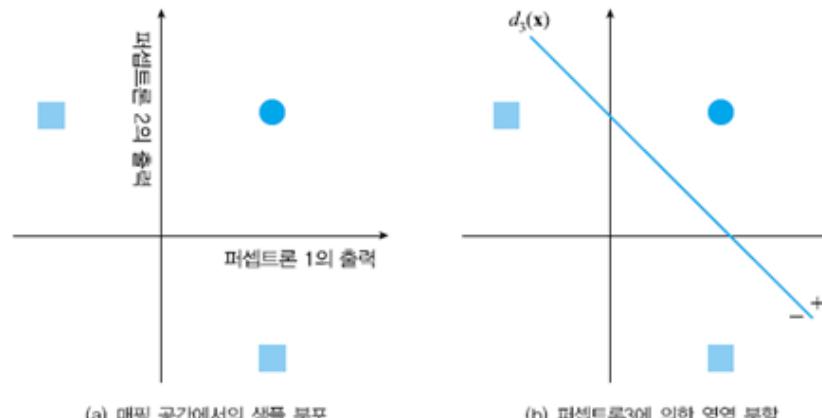
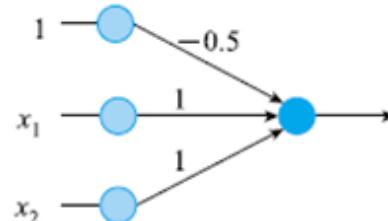


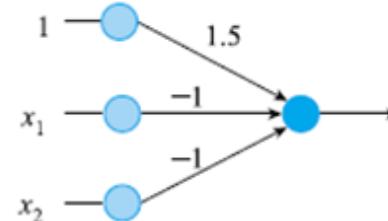
그림 4.7 새로운 공간에서의 샘플 분포와 영역 분할

다중 퍼셉트론

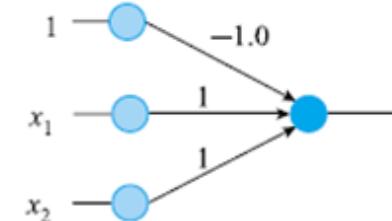
■ 다중 퍼셉트론 (MLP; Multi-layer perceptron)



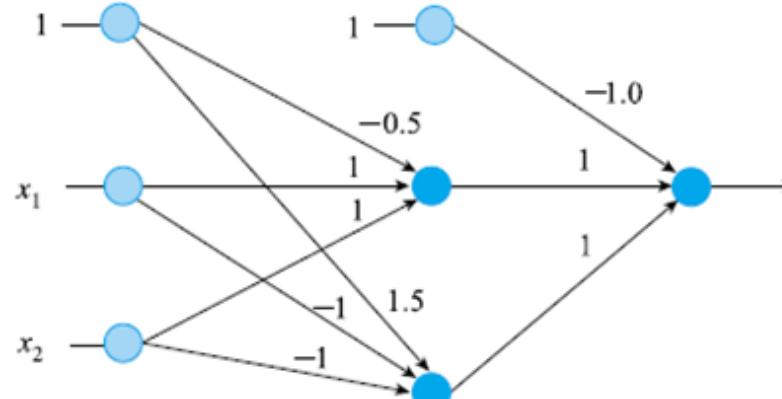
(a) 퍼셉트론1



(b) 퍼셉트론2



(c) 퍼셉트론3

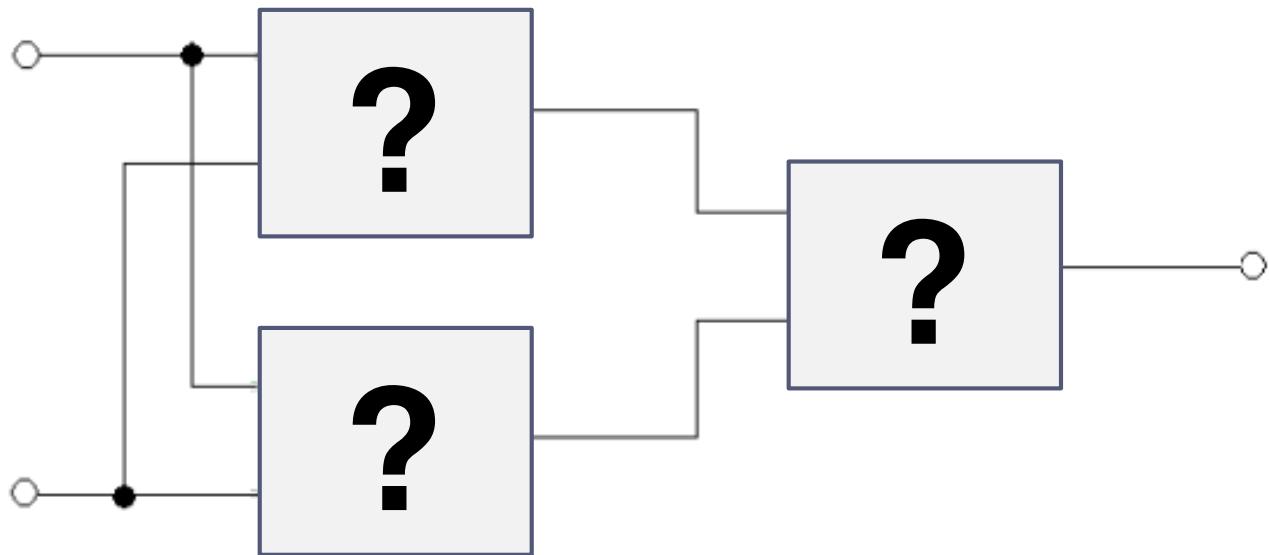
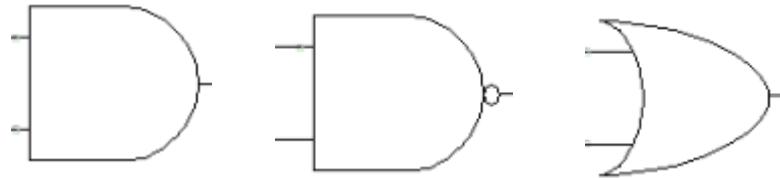


(d) 다중 퍼셉트론

그림 4.8 세 개의 퍼셉트론과 이들을 연결하여 만든 다중 퍼셉트론

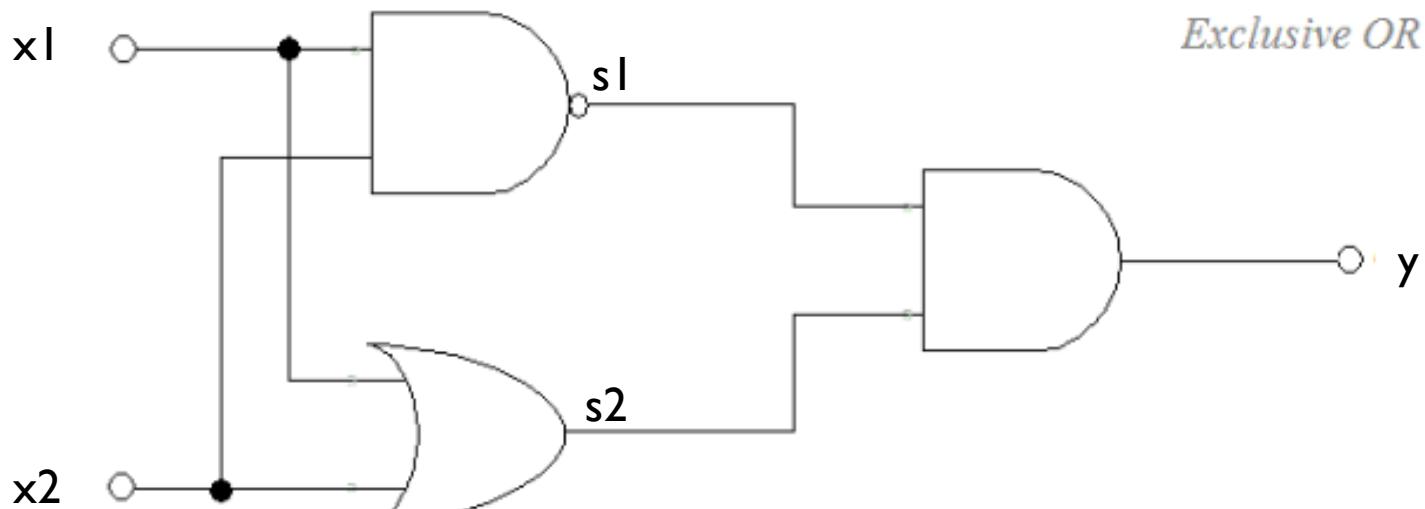
다중 퍼셉트론

- XOR
 - AND, NAND, OR 이용



다중 퍼셉트론

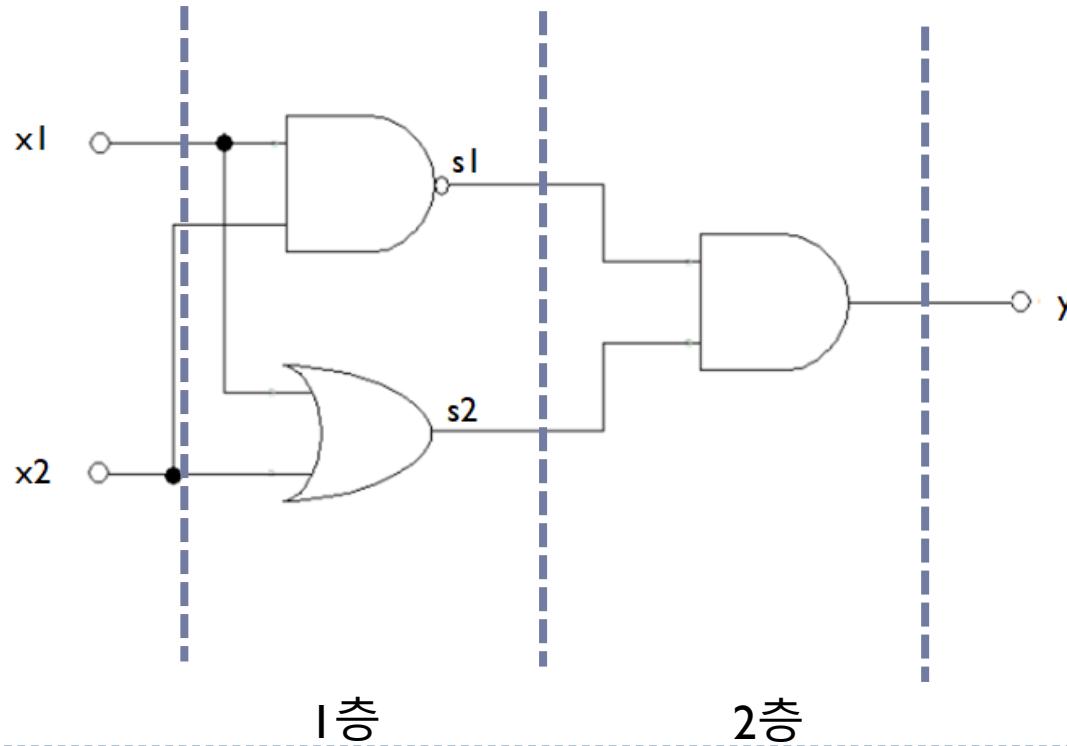
- XOR
 - AND, NAND, OR 이용



© www.petervis.com

실습

- 퍼셉트론 구현하기
 - XOR
 - 다층 퍼셉트론
 - XOR : 2층 퍼셉트론



다중 퍼셉트론

- 다중 퍼셉트론의 아키텍처
 - 입력 층, 은닉 층, 출력 층
 - 가중치: u 와 v

- 신경망은 일종의 함수

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

$$\begin{aligned}\mathbf{z} &= p(\mathbf{x}) \\ \mathbf{o} &= q(\mathbf{z})\end{aligned}$$

또는

$$\mathbf{o} = q(p(\mathbf{x}))$$

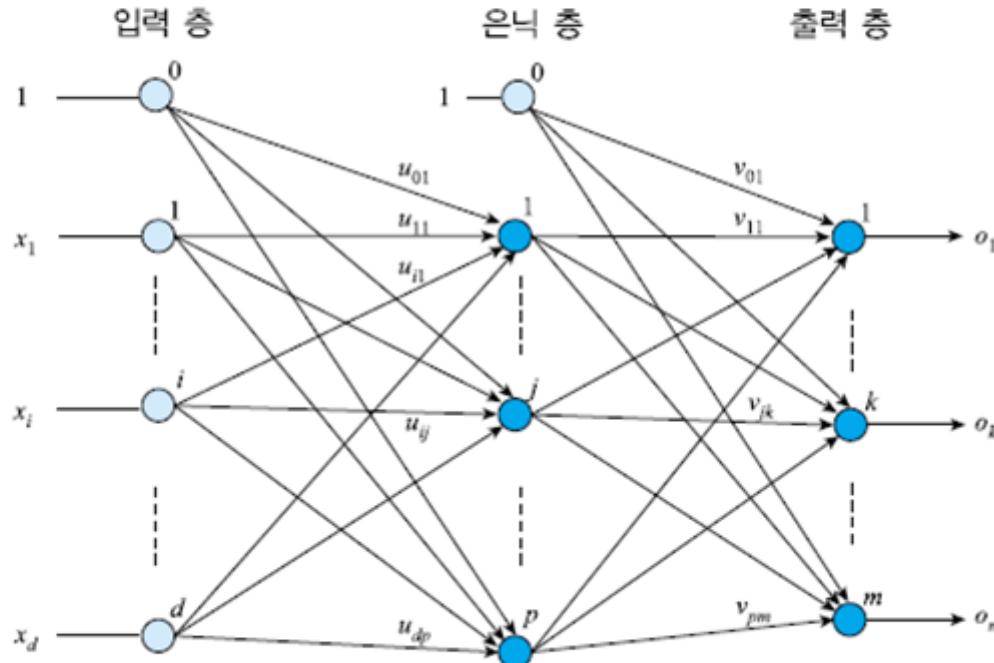


그림 4.9 다중 퍼셉트론의 구조와 표기

다중 퍼셉트론

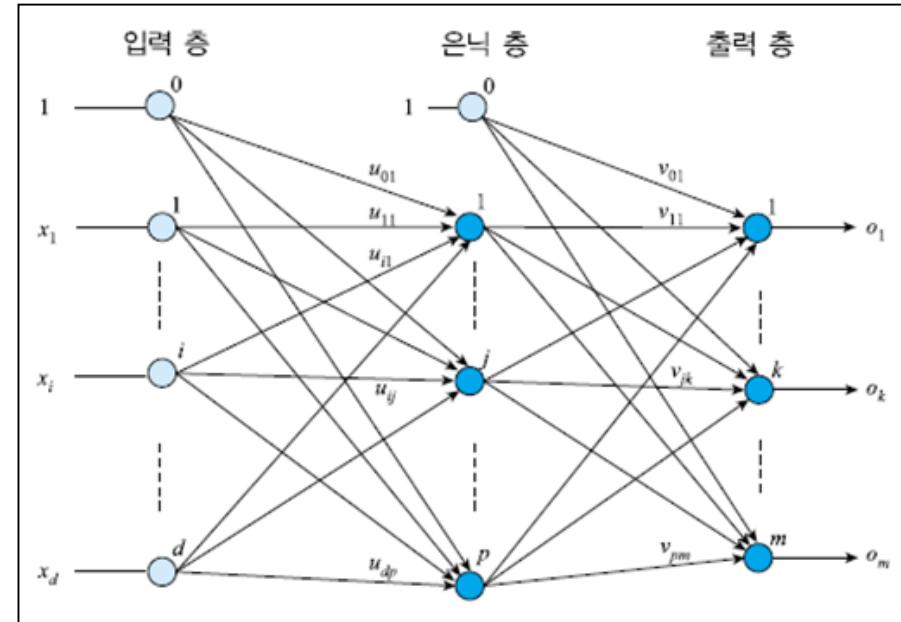
■ 전방 계산 (forward computation)

은닉 층의 j 번째 노드, $1 \leq j \leq p$:

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

출력 층의 k 번째 노드, $1 \leq k \leq m$:

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



다중 퍼셉트론

■ 활성 함수 (activation function)

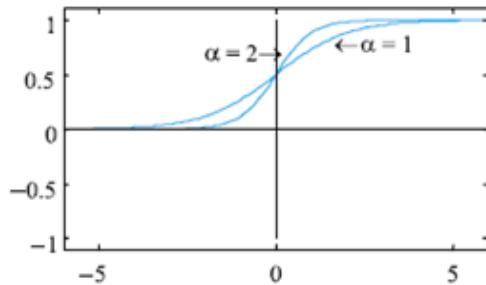
□ 시그모이드 사용

이진 시그모이드 함수:

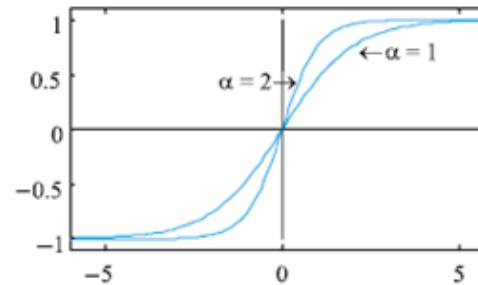
$$\left. \begin{array}{l} \tau_1(x) = \frac{1}{1+e^{-\alpha x}} \\ \tau_1'(x) = \alpha \tau_1(x)(1-\tau_1(x)) \end{array} \right\} \quad (4.14)$$

양극 시그모이드 함수:

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



(a) 이진 시그모이드

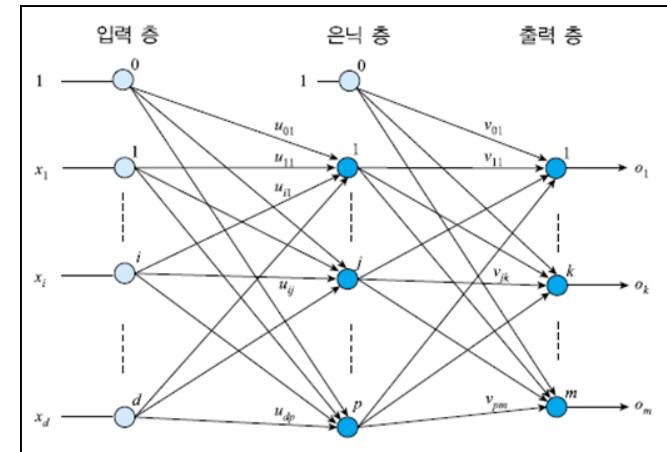


(b) 양극 시그모이드

그림 4.10 활성 함수로 널리 사용되는 두 가지 시그모이드 함수

다중 퍼셉트론

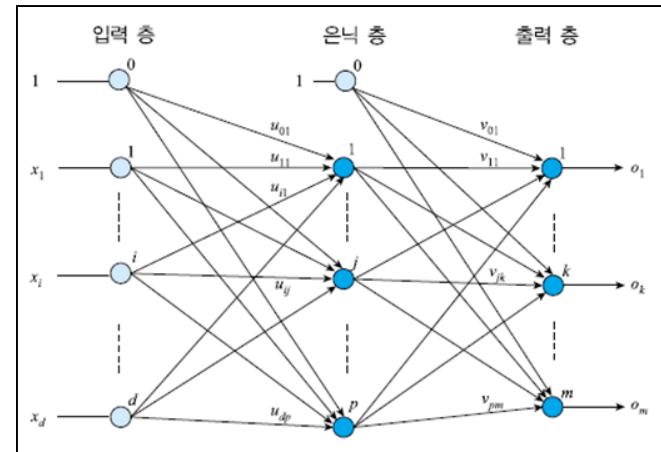
- 신경망 구조 설계 시 고려사항
 - 몇 개의 층을 둘 것인가?
 - 층간의 연결은 어떻게 할 것인가?
 - 각 층에 있는 노드를 몇 개로 할 것인가?
 - 어떤 활성화 함수를 사용할 것인가?



다중 퍼셉트론

■ MLP의 학습이란?

- MLP 학습이란? 훈련 집합 $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ 이 주어졌을 때 이들을 분류하는 다층 퍼셉트론 (즉 \mathbf{u} 와 \mathbf{v})을 찾아라. (x_i, t_i) 에서 x_i 는 특징 벡터이고 t_i 는 부류 표지 벡터로서 class label vector (또는 목적 벡터라고도 target vector 함) $x_i \in \omega_j$ 이면 $t_i = (0, \dots, 1, \dots, 0)^T$ 이다. 즉 j 번째 요소만 1이고 나머지 요소는 모두 0을 갖는다. 이것은 이진 모드를 사용할 때의 값이고 만일 양극 모드를 사용한다면 $t_i = (-1, \dots, 1, \dots, -1)^T$ 로 하면 된다.



다중 퍼셉트론

- 단계 1
 - ▣ (4.12)와 (4.13)의 전방 계산이 분류기의 식
 - ▣ 매개변수 집합 $\Theta = \{\mathbf{u}, \mathbf{v}\}$
- 단계 2 (비용 함수 정의)

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

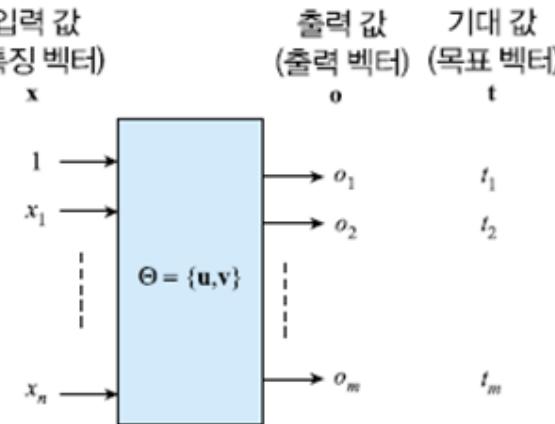


그림 4.12 다중 퍼셉트론의 입력, 출력, 그리고 기대값

다중 퍼셉트론

■ 단계 3 (최적 해 찾음)

- (4.16)의 오류를 줄이는 방향으로 Θ 를 수정해 나감

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

알고리즘 [4.4]

다중 퍼셉트론 (MLP) 학습

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

출력: 가중치 \mathbf{u} 와 \mathbf{v}

알고리즘:

1. \mathbf{u} 와 \mathbf{v} 를 초기화한다.
2. **repeat** {
3. **for** (X 의 샘플 각각에 대해) {
4. (4.12)와 (4.13)으로 전방 계산을 한다.
5. $\frac{\partial E}{\partial \mathbf{v}}$ 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?

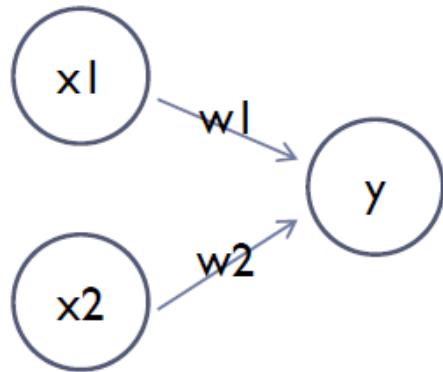
→ 오류 역전파



Review) 퍼셉트론

- 퍼셉트론

- 다수의 신호를 입력으로 받아 하나 혹은 여러 신호 출력

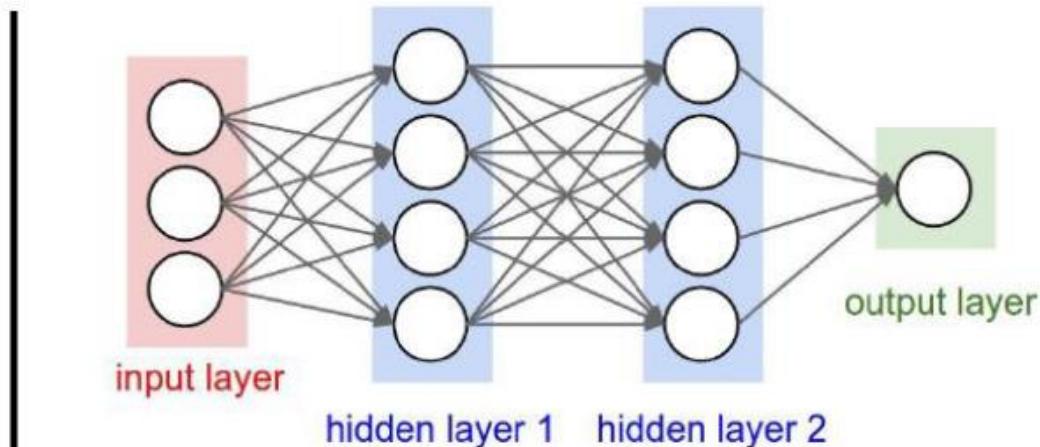
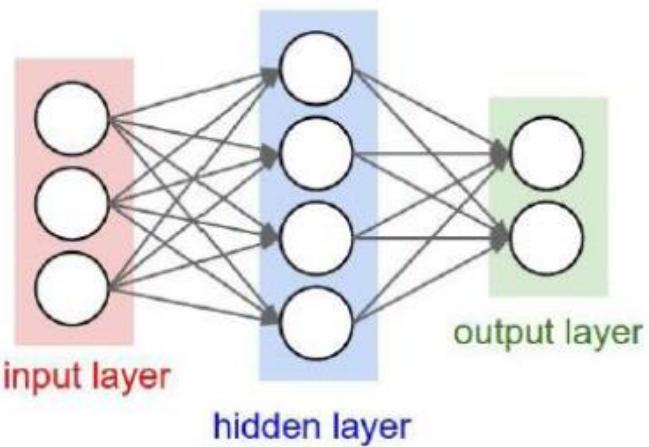
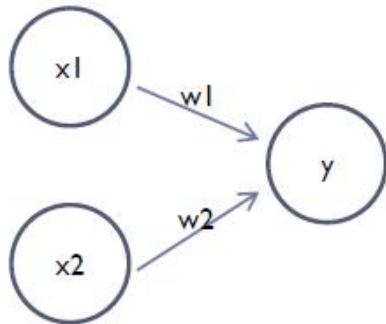


- 한계점

- 퍼셉트론을 이용한 함수 구현시 weight값 설정
 - AND, OR, NAND, ...
 - **weight값 자동 계산 방법 필요**



퍼셉트론 vs. 신경망



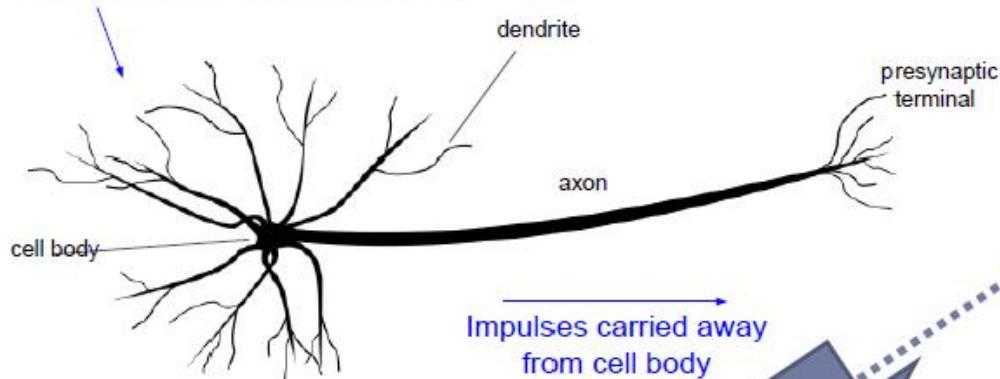
“2-layer Neural Net”, or
“1-hidden-layer Neural Net”

“Fully-connected” layers

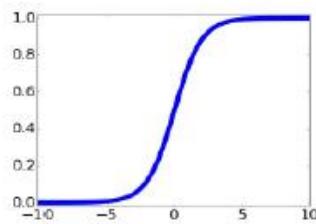
“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

퍼셉트론 vs. 신경망

Impulses carried toward cell body



This image by Felipe Perucco
is licensed under CC-BY 3.0



sigmoid activation function

$$\frac{1}{1 + e^{-x}}$$

x_0

w_0 synapse

axon from a neuron

$w_0 x_0$

dendrite

$w_1 x_1$

$w_2 x_2$

cell body

$$\sum_i w_i x_i + b$$

$$f \left(\sum_i w_i x_i + b \right)$$

activation
function

output axon

활성화함수

- 왜?

(Before) Linear score function: $f = Wx$
(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

The function $\max(0, z)$ is called the **activation function**.

Q: What if we try to build a neural network without one?

$$f = W_2 W_1 x$$



활성화함수

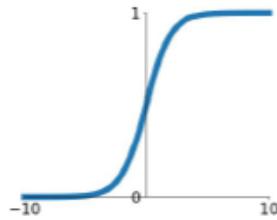
- 개념

- 입력 신호의 총합을 출력 신호로 변환하는 함수

Activation functions

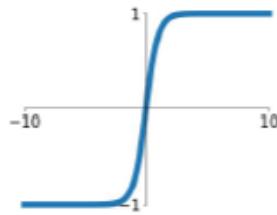
Sigmoid

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



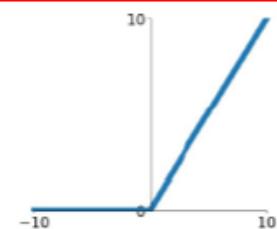
tanh

$$\tanh(x)$$



ReLU

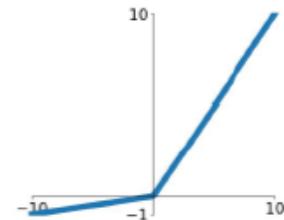
$$\max(0, x)$$



ReLU is a good default choice for most problems

Leaky ReLU

$$\max(0.1x, x)$$

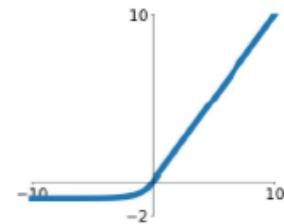


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

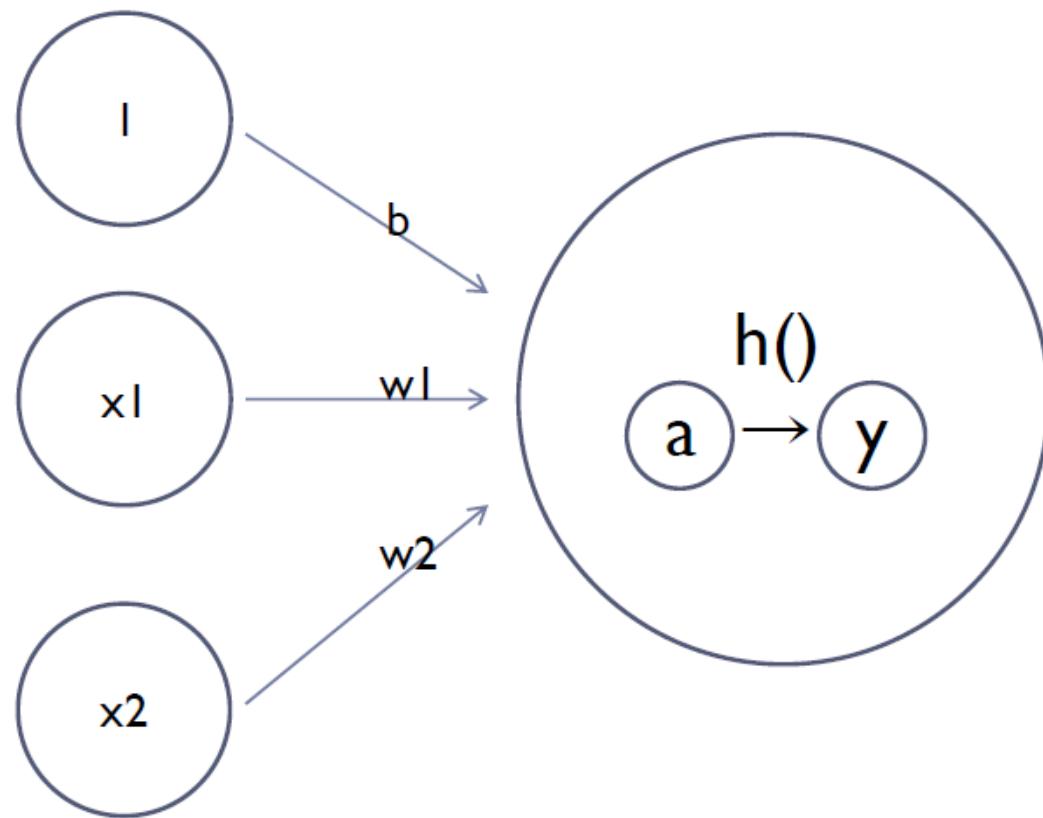
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



활성화함수

- 개념
 - 입력 신호의 총합을 출력 신호로 변환하는 함수



실습

■ 다층 퍼셉트론

■ MLPClassifier

sklearn.neural_network.MLPClassifier

```
class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001,
batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True,
early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)
```

[source]

Multi-layer Perceptron classifier.

alpha : float, default=0.0001

L2 penalty (regularization term) parameter.

learning_rate : {'constant', 'invscaling', 'adaptive'}, default='constant'

Learning rate schedule for weight updates.

- 'constant' is a constant learning rate given by 'learning_rate_init'.
- 'invscaling' gradually decreases the learning rate at each time step 't' using an inverse scaling exponent of 'power_t'. effective_learning_rate = learning_rate_init / pow(t, power_t)
- 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if 'early_stopping' is on, the current learning rate is divided by 5.

Only used when `solver='sgd'`.

max_iter : int, default=200

Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations. For stochastic solvers ('sgd', 'adam'), note that this determines the number of epochs

hidden_layer_sizes : tuple, length = n_layers - 2, default=(100,)

The ith element represents the number of neurons in the ith hidden layer.

activation : {'identity', 'logistic', 'tanh', 'relu'}, default='relu'

Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns $f(x) = x$
- 'logistic', the logistic sigmoid function, returns $f(x) = 1 / (1 + \exp(-x))$.
- 'tanh', the hyperbolic tan function, returns $f(x) = \tanh(x)$.
- 'relu', the rectified linear unit function, returns $f(x) = \max(0, x)$

solver : {'lbfgs', 'sgd', 'adam'}, default='adam'

The solver for weight optimization.

- 'lbfgs' is an optimizer in the family of quasi-Newton methods.

- 'sgd' refers to stochastic gradient descent.

- 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

Note: The default solver 'adam' works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, 'lbfgs' can converge faster and perform better.

참고: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

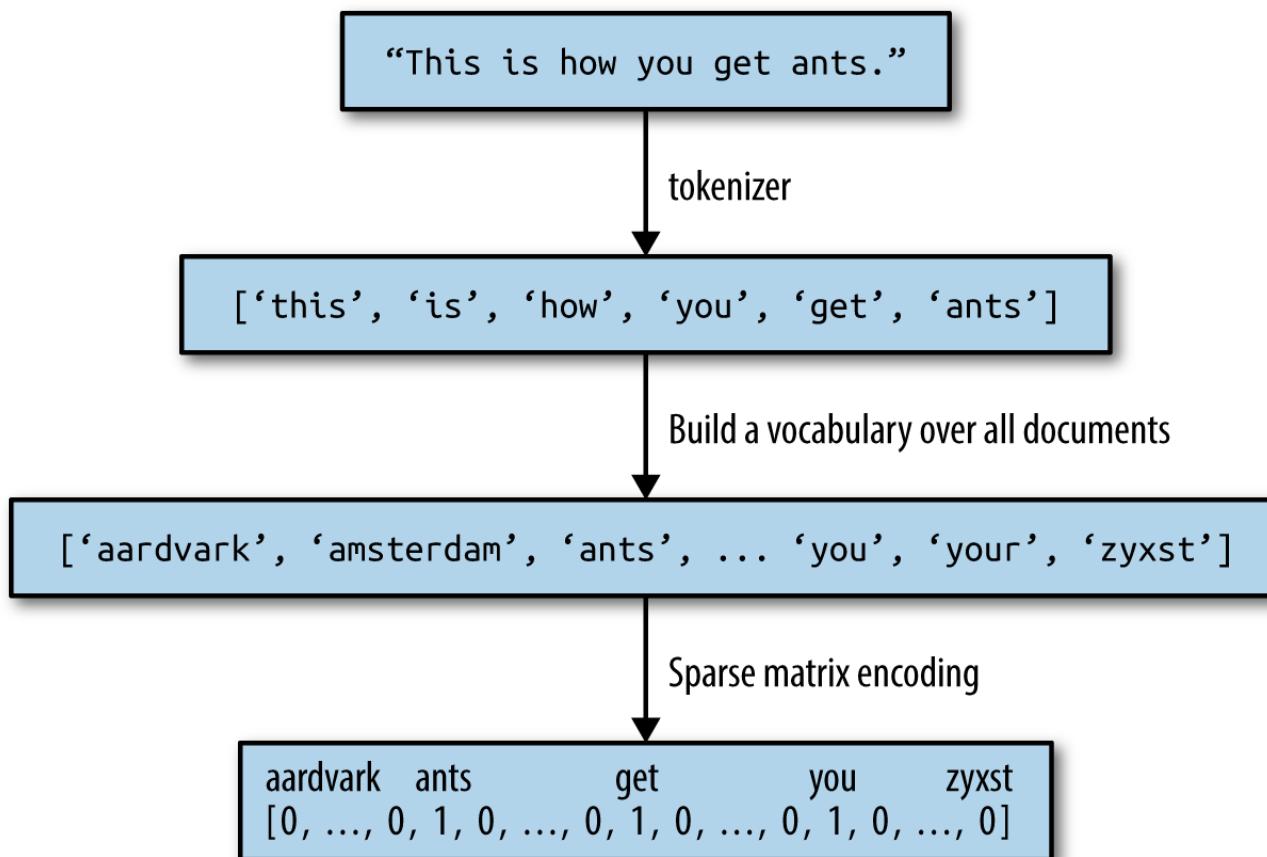
실습

- 영화 리뷰 감성 분석
 - IMDB 데이터셋
 - 영화 리뷰 데이터로 양성, 음성으로 레이블 포함
 - 1 ~ 10 점수가 있고 7점 이상은 양성, 4점 이하는 음성으로 분류
 - Bag-of-words
 - 머신러닝에서 텍스트를 표현하는 가장 간단하고 효과적인 방법
 - 텍스트에서 각 단어에 대한 빈도수 측정
 - 구조와 상관없이 단어의 출현 횟수만 고려
 - 단계
 - 토큰화
 - 어휘사전구축
 - 인코딩



실습

- 영화 리뷰 감성 분석
 - Bag-of-words



그밖에..

- 부연 설명
 - 네트워크 아키텍처
 - 은닉 층 개수, 은닉 노드 개수, ...
 - 가중치 초기화
 - 종료 시점
 - 활성함수
 - 학습률
 -

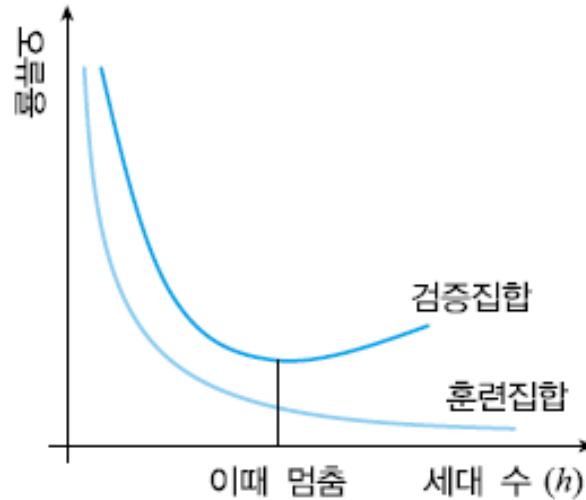


그림 4.15 일반화 기준에 따른 멈춤 조건

양상블



앙상블

- 개념

- 다수의 예측 모형을 결합하여 성능을 높임
 - 단일 모형에 비해 성능 분산 감소 (오버피팅 감소)
 - 성능이 낮은 개별 모형이 모여 성능을 향상 시킴

- 방법

- 취합 (Aggregation)
 - 다수결 (Majority voting)
 - 배깅 (Bagging)
 - 랜덤 포레스트 (Random forests)
- 부스팅 (Boosting)
 - 에이다 부스트 (AdaBoost)
 - 그레디언트 부스트(Gradient Boost)



앙상블

- Voting
 - 가장 단순한 결합 방법
 - 각 독립적인 모형과 결합 가능
 - 방법
 - Hard voting: 단순 투표. 개별 모형의 결과 기준
 - Soft voting: 가중치 투표. 개별 모형의 조건부 확률의 합 기준
 - 실습



앙상블

- 결정 트리 앙상블

- 랜덤 포레스트

- 결정 트리의 단점(과대적합) 보완 할 수 있는 방법
 - 조금씩 다른 여러 결정 트리의 묶음
 - 잘 작동하는 과대적합 트리를 많이 만들고 결과를 평균 내어 과대적합을 줄임 (수학적으로 증명됨)
 - 결정 트리를 많이 만들어야 함
 - 각 트리는 타깃 예측을 잘 해야 함
 - 다른 트리와 구별되어야 함
 - 랜덤 트리 생성 두 가지 방법
 - 트리를 만들 때 사용하는 데이터 포인트를 무작위로 선택
 - 분할 테스트에서 특성을 무작위로 선택



앙상블

- 결정 트리 앙상블
 - 랜덤 포레스트 구축
 - 생성 할 트리 개수 정함 (독립적인 트리)
 - 부트스트랩(bootstrap sample) 샘플 생성
 - $n_{samples}$ 개의 데이터 포인트 중에서 무작위로 $n_{samples}$ 만큼 반복 추출 (중복 추출 허용)
 - 원래 데이터 셋과 크기는 같지만 중복, 누락 될 수 있음

To illustrate, let's say we want to create a bootstrap sample of the list ['a', 'b', 'c', 'd']. A possible bootstrap sample would be ['b', 'd', 'd', 'c']. Another possible sample would be ['d', 'a', 'd', 'a'].



앙상블

- 결정 트리 앙상블
 - 랜덤 포레스트 구축
 - 결정 트리 생성
 - 후보 특성을 무작위로 선택한 후 최선의 tests 찾음
 - max_feature로 몇 개의 특성을 고를지 조정
 - 부트스트랩 샘플링 때문에 다른 데이터 셋으로 트리가 생성됨
 - 각 노드에서 특성의 일부만 사용
 - 랜덤 포레스트의 모든 트리가 서로 다르게 만들어짐
 - max_feature로 트리의 다름을 설정함
 - 값을 크게 하면 트리가 비슷해짐
 - 값을 작게 하면 트리가 달라짐



앙상블

- 결정 트리 앙상블
 - 그래디언트 부스팅 회귀 트리
 - 여러 개의 결정 트리를 묶어 강력한 모델을 만듦
 - 분류, 회귀에 모두 사용
 - 이전 트리의 오차를 보완하는 방식으로 순차적으로 트리 생성
 - 무작위성이 없음
 - 강력한 사전 가지치기 사용
 - 1 ~ 5 depth 정도의 깊지 않은 트리 사용
 - 메모리를 적게 사용
 - 예측 빠름
 - weak learner을 많이 연결하여 성능을 높임
 - 일부 데이터에만 성능이 좋기 때문에 트리가 많을 수록 성능 향상
 - 매개변수에 조금 민감 (learning_rate, n_estimators)



양상블

- 실습

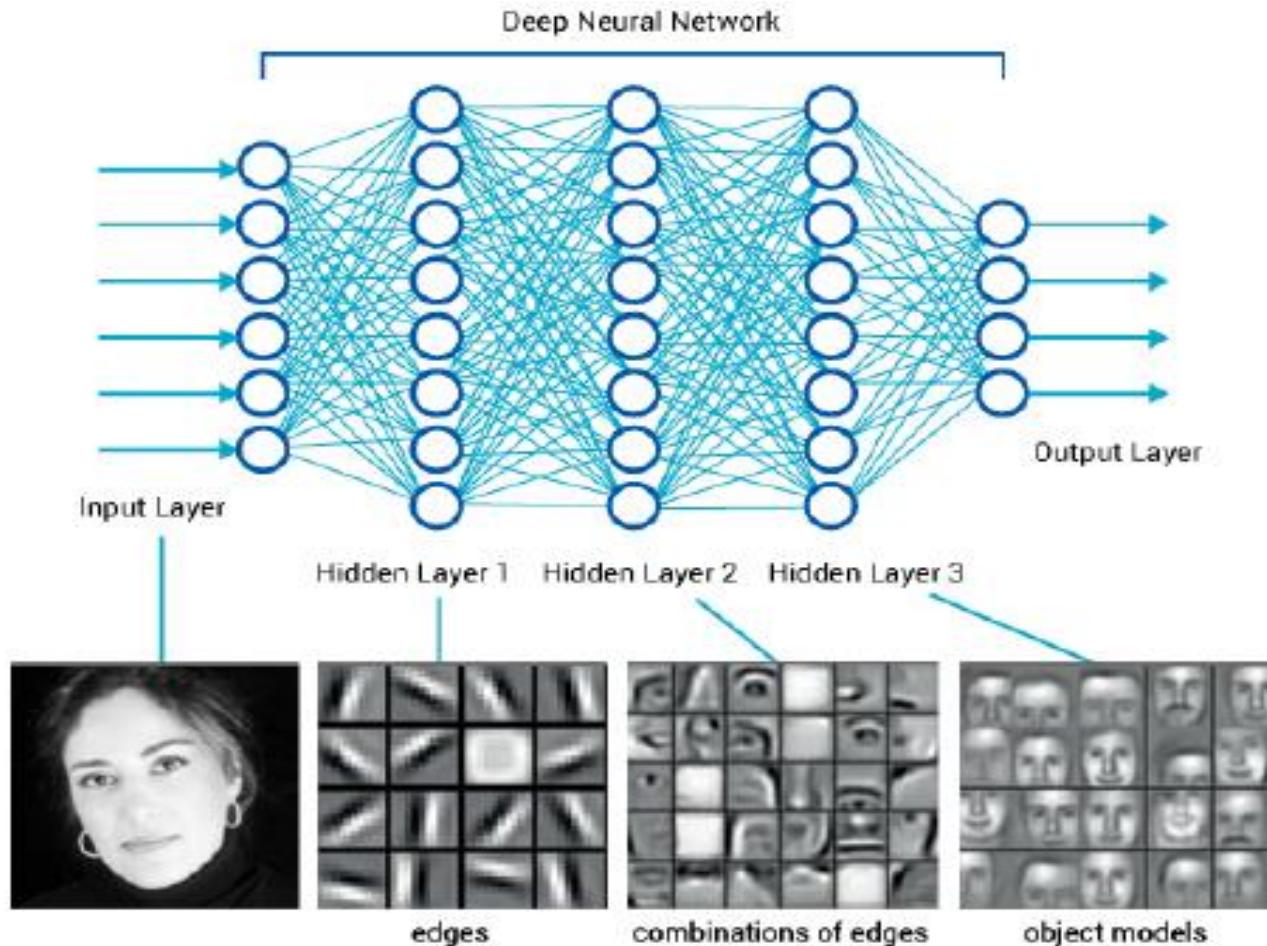


딥러닝 소개



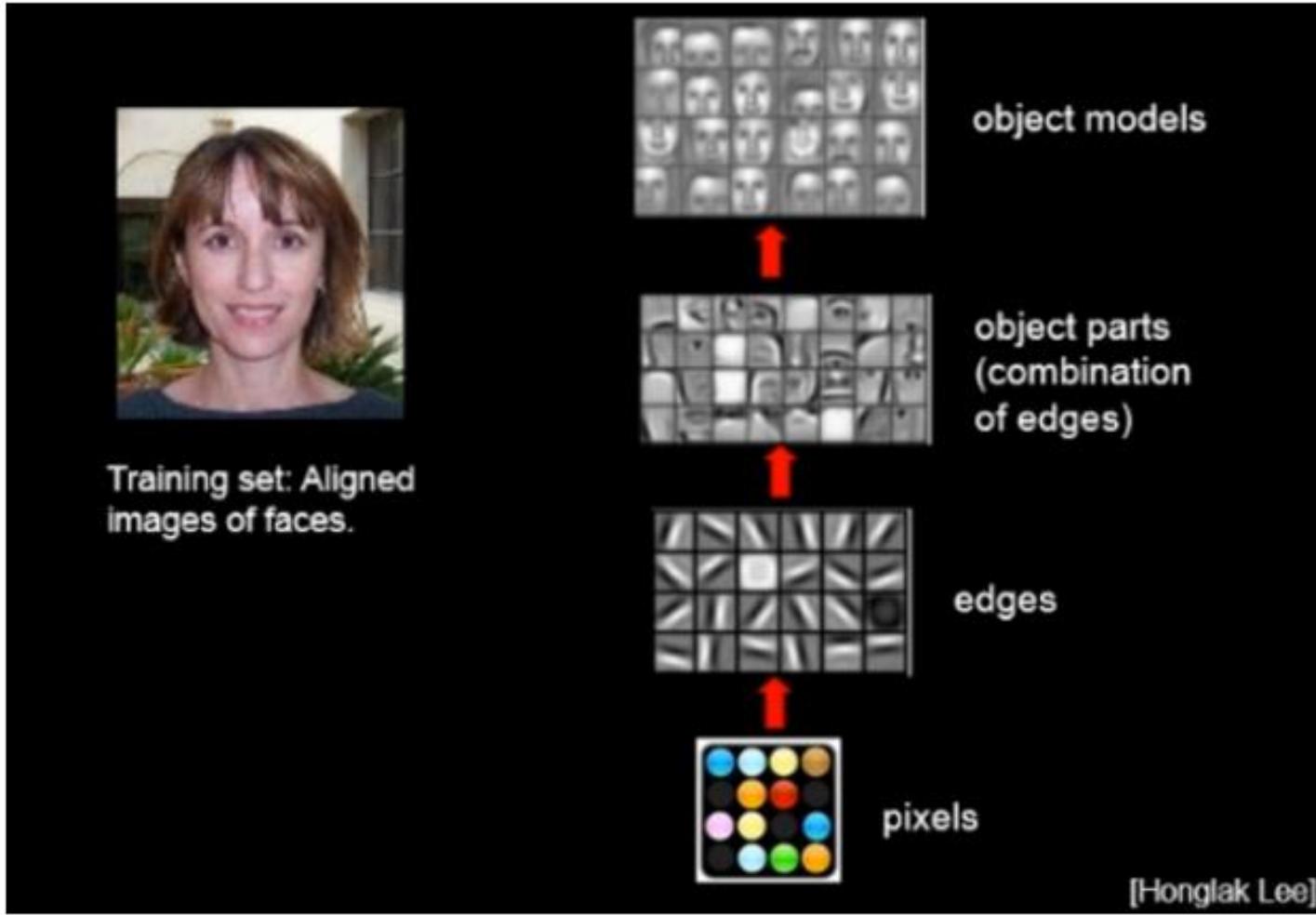
딥러닝?

- Deep Learning == Deep Artificial Neural Networks



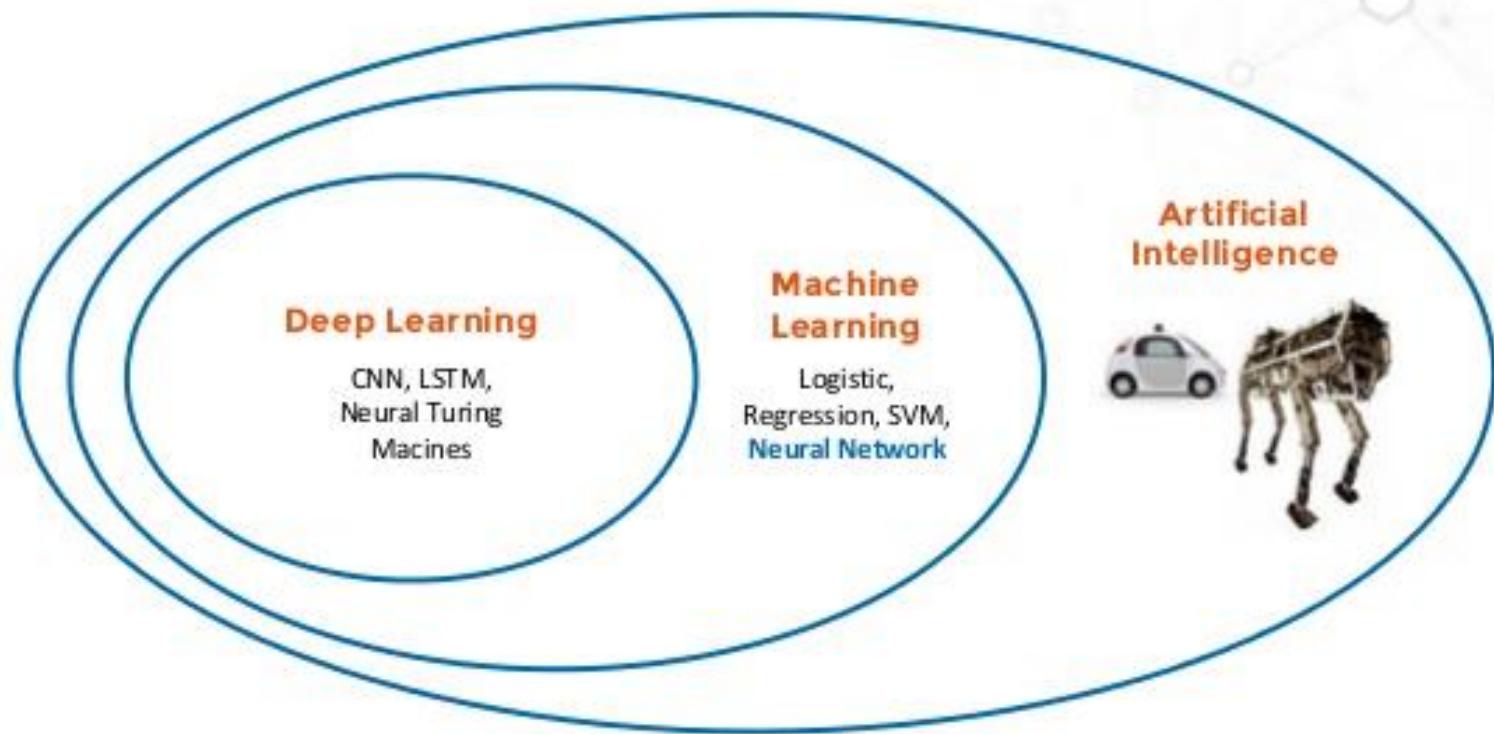
딥러닝?

- Deep Learning == Feature Learning



Review) AI vs. ML vs. DNN

From AI to Deep Learning



[≡]



Review) 머신러닝의 정의

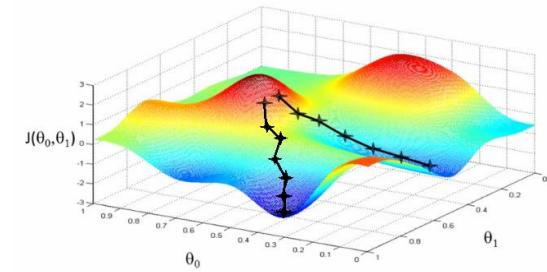
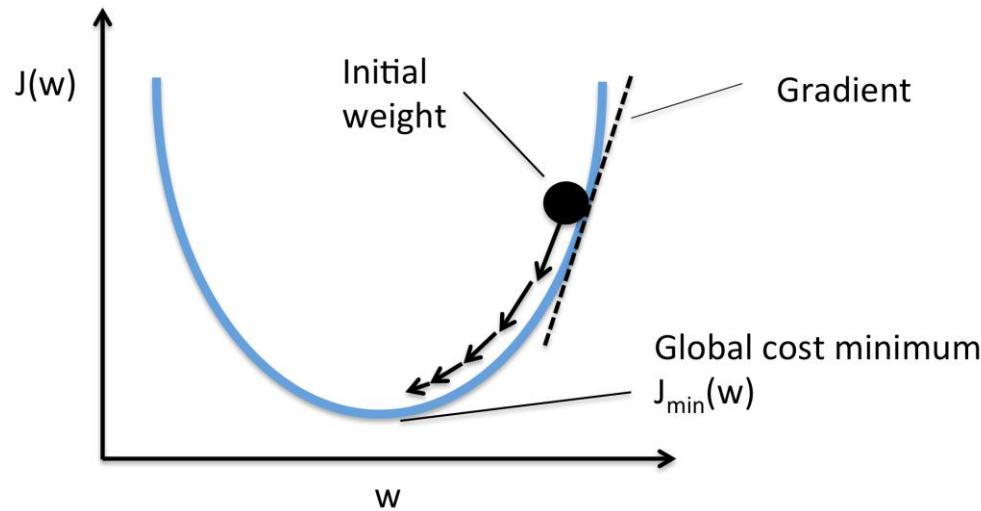
Machine Learning definition

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.



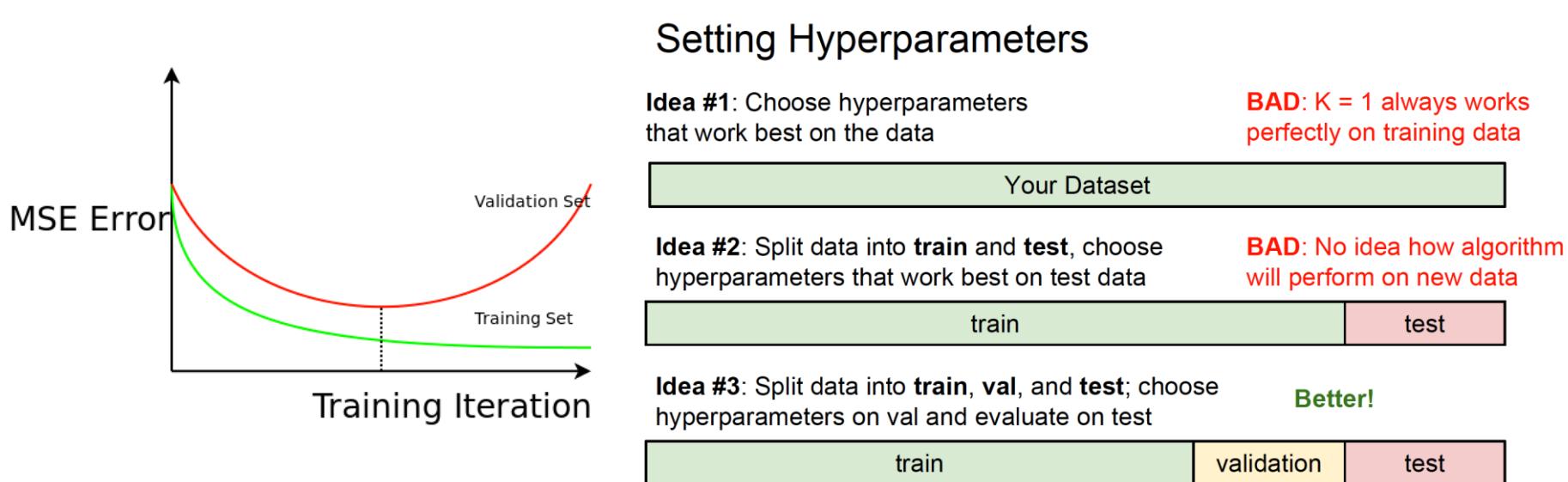
머신러닝 프로세스

- Hypothesis
- Cost Function
- Optimization
 - Local Optima Problem



데이터 셋

- Train data: 학습에 사용되는 데이터
- Validation data: 검증용 데이터. Training error와 validation error를 이용하여 어느 시점에 멈추어야 overfitting을 방지할 수 있을지 결정
- Test data: 머신러닝 알고리즘의 일반화 능력을 측정하기 위한 데이터



History

- 1세대: 1943~1986
- McCulloch, Warren S., and Walter Pitts 인공신경망 아이디어 제안

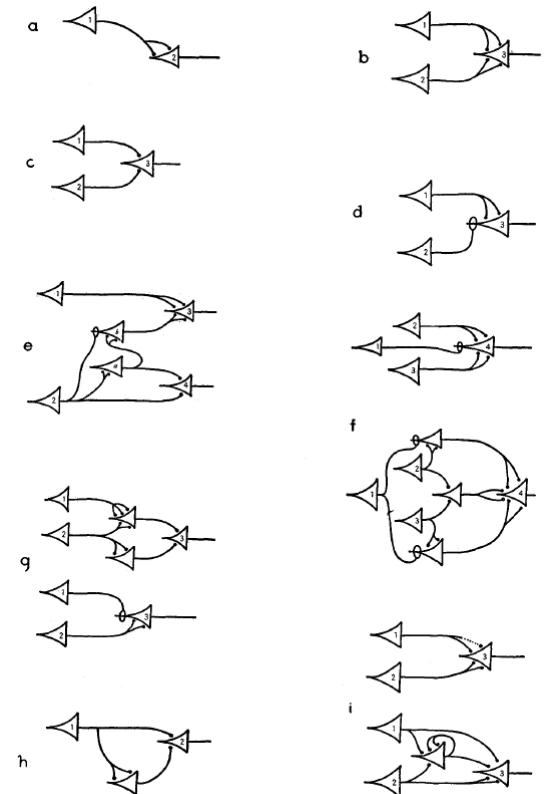
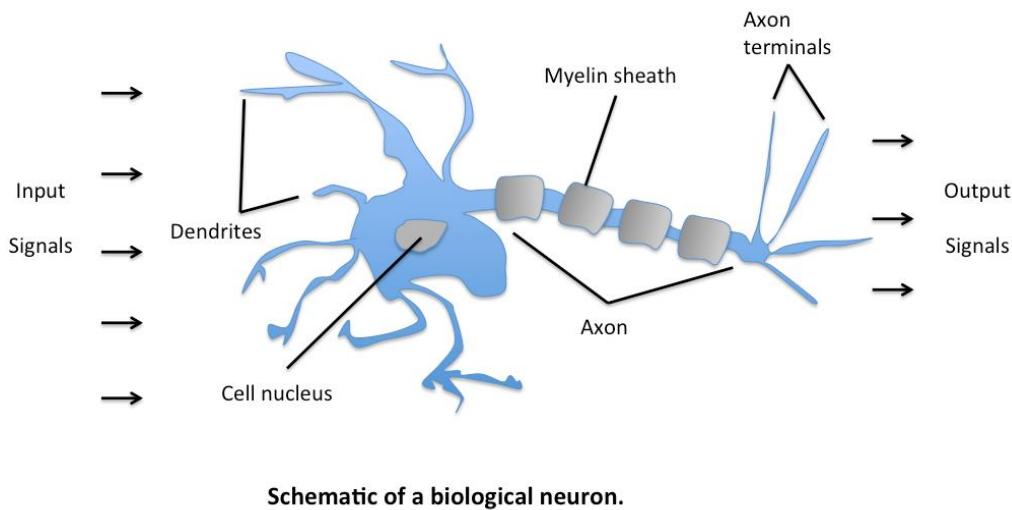
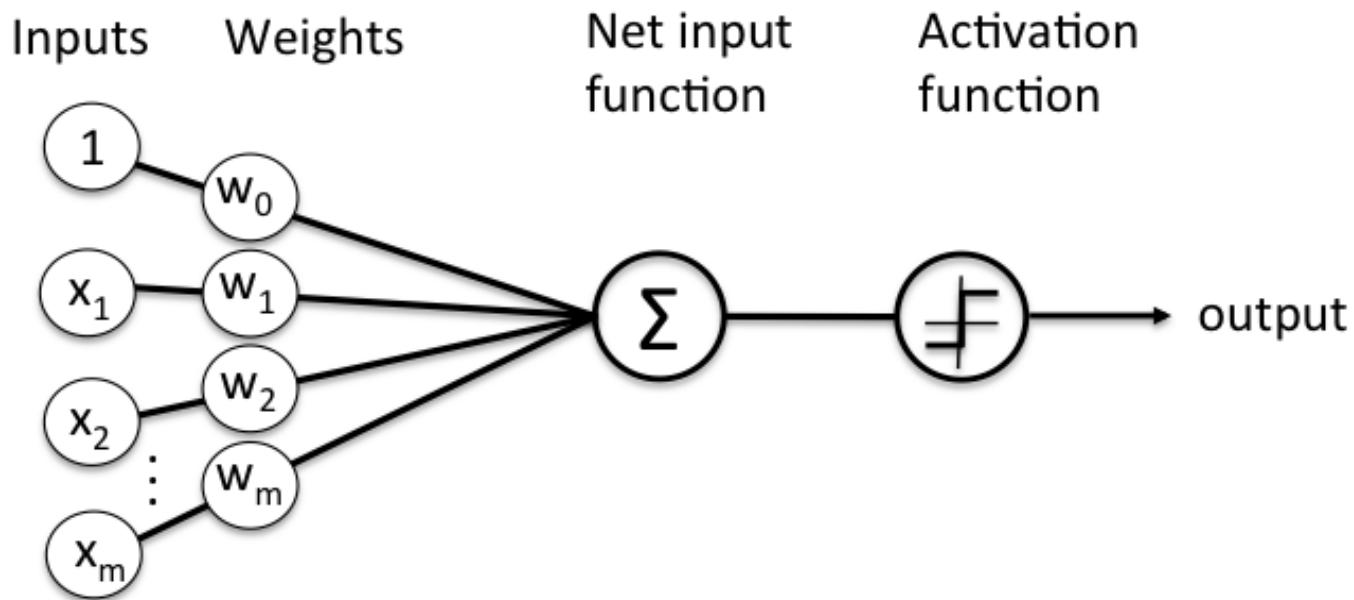


FIGURE 1

History

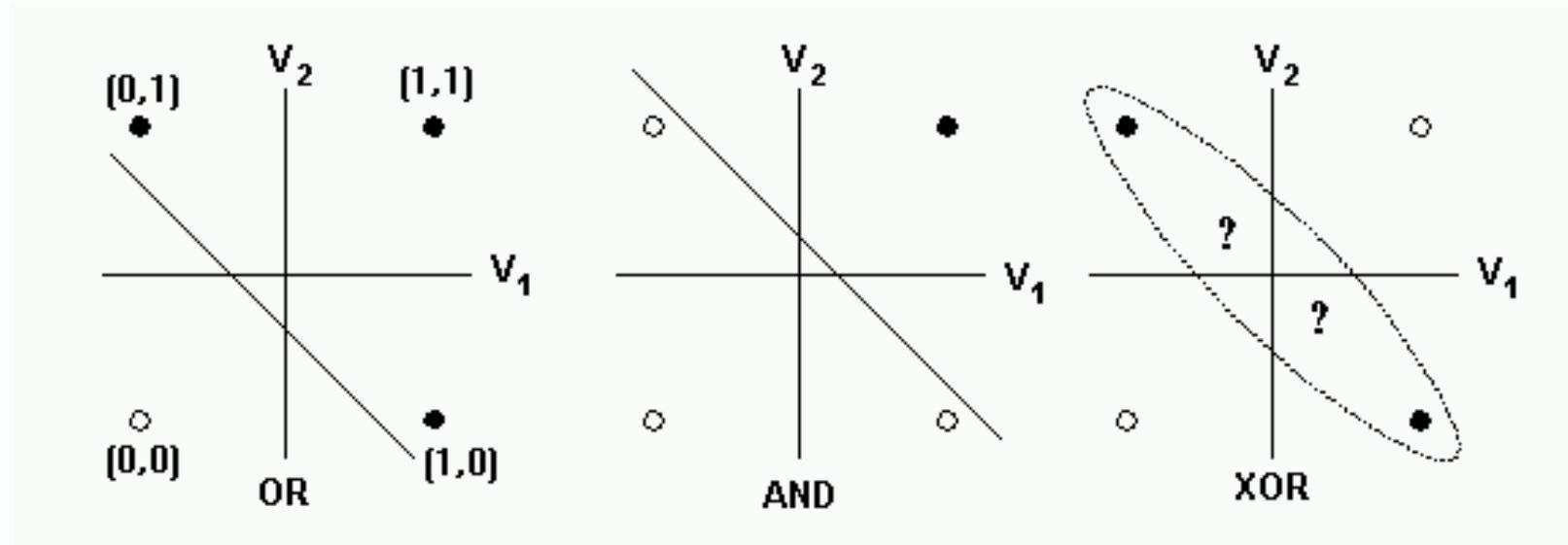
- 1세대: 1943~1986
- Frank Rosenblatt 퍼셉트론 모형 제안



Schematic of Rosenblatt's perceptron.

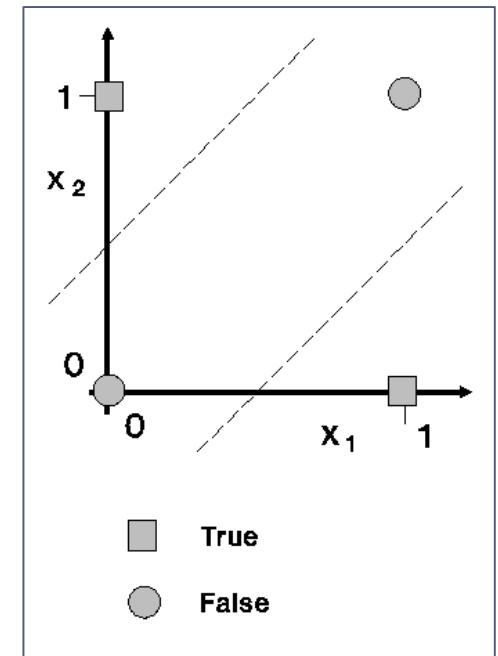
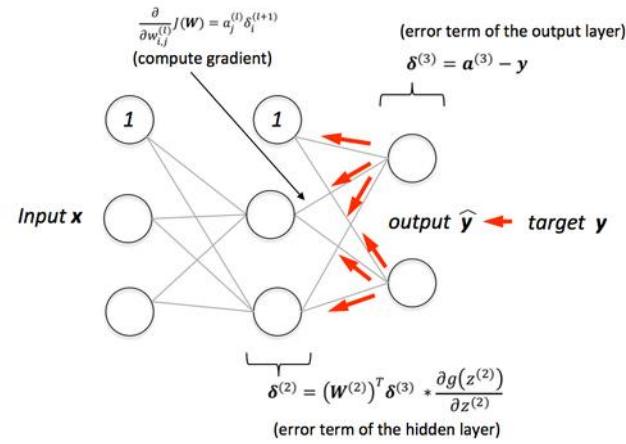
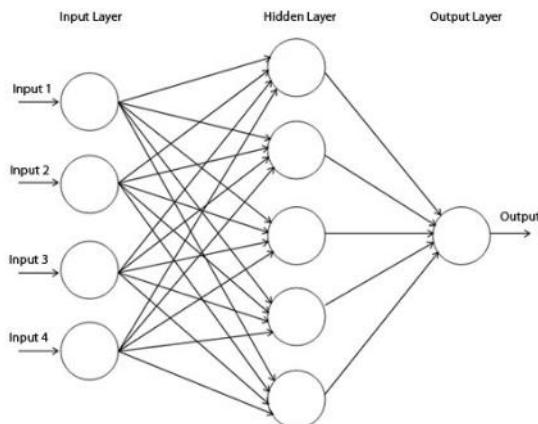
History

- 1세대: 1943~1986
- Marvin Minsky, Seymour Papert 퍼셉트론의 한계 지적
 - 선형 분리 불가능
 - ANN의 암흑기



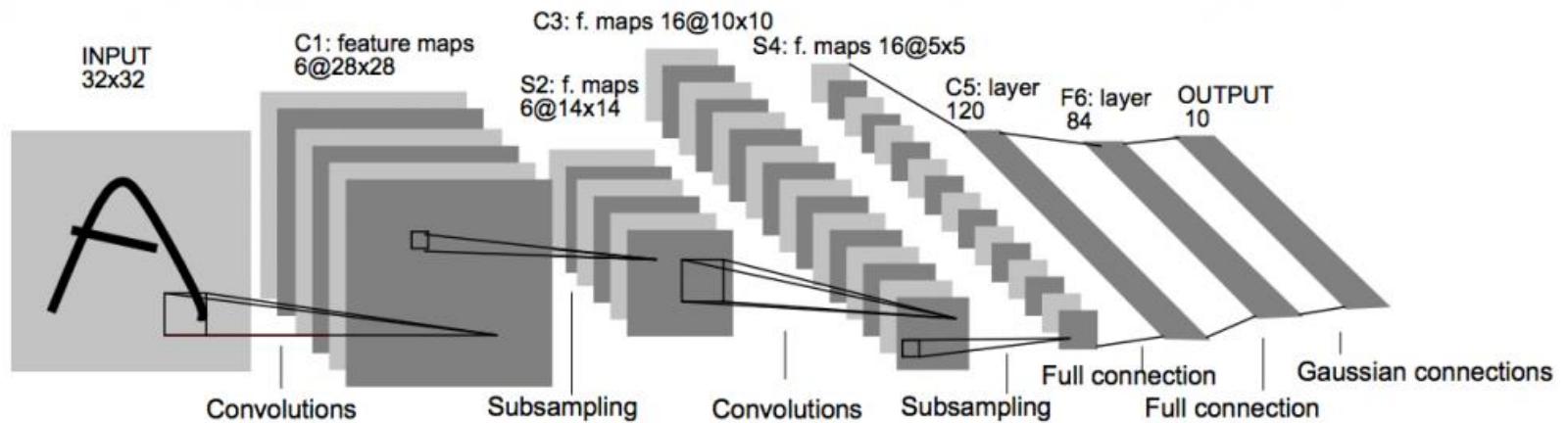
History

- 2세대: 1986~2006
- Multi-Layer Perceptron, Backpropagation 제안
 - XOR 문제 해결



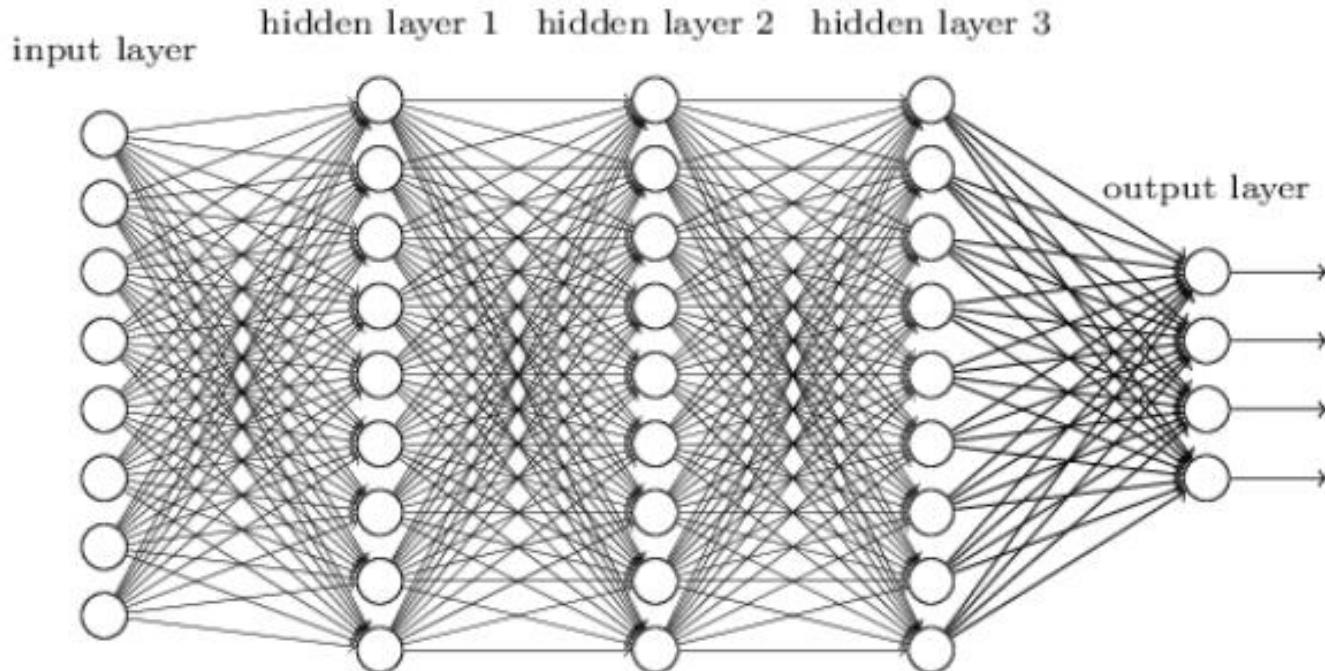
History

- 2세대: 1986~2006
- 1988년 Yann Lecun⁰ | Convolutional Neural Networks(CNN) 제안
 - LeNet-5
 - 수표 인식 문제 등에 실제로 사용



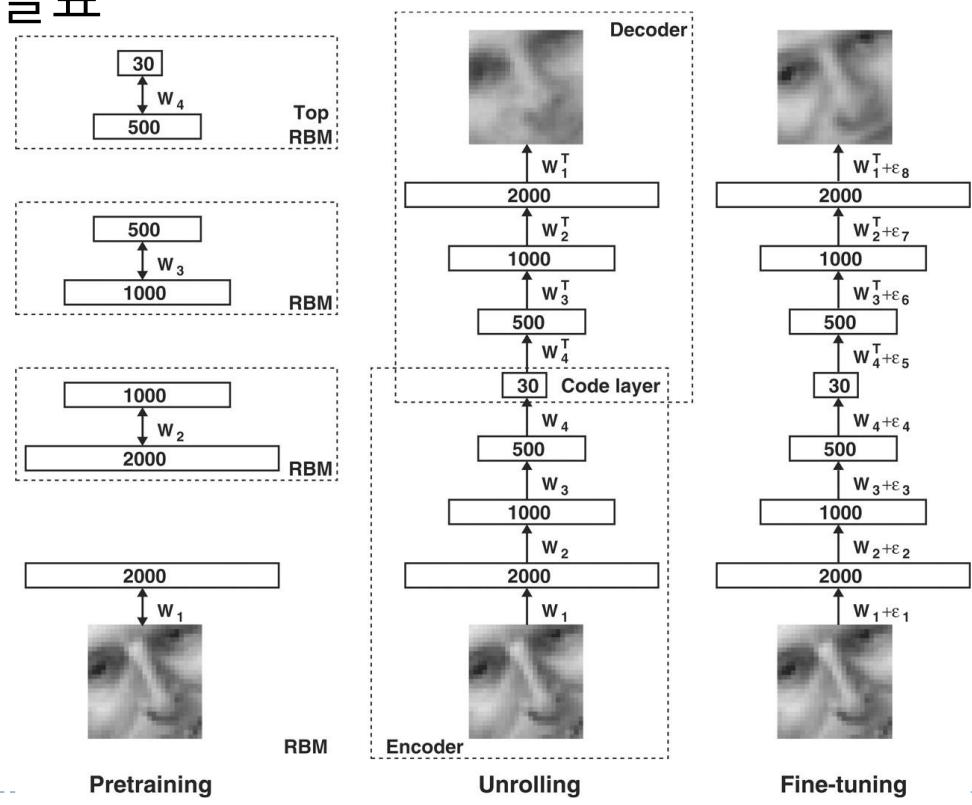
History

- 2세대: 1986~2006
- Layer가 깊어질수록 vanishing gradient problem 문제 발생
 - 딥러닝이 불가능해짐
 - 다른 기법 탐구 (SVM, ...)



History

- 3세대: 2006~2012
- ANN은 다시 암흑기로...
- Geoffrey Hinton과 그 외 딥러닝 선구자들의 연구 지속됨
- 2006년 Deep Autoencoders 발표



History

- 3세대: 2006~2012
- Dropout, ReLU 등 Overfitting을 방지하기 위한 알고리즘 등장
- 인터넷의 발전으로 인한 학습에 사용할 빅데이터를 구할수 있는 환경 도래
- GPU 발전

Big Data

facebook

350 millions
images uploaded
per day

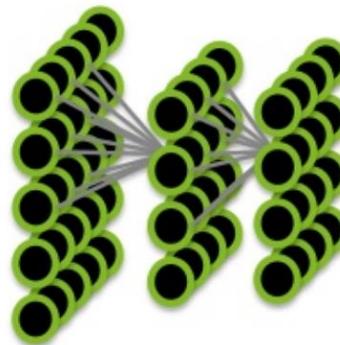
Walmart >*

2.5 Petabytes of
customer data
hourly

YouTube

300 hours of video
uploaded every
minute

Better Algorithms

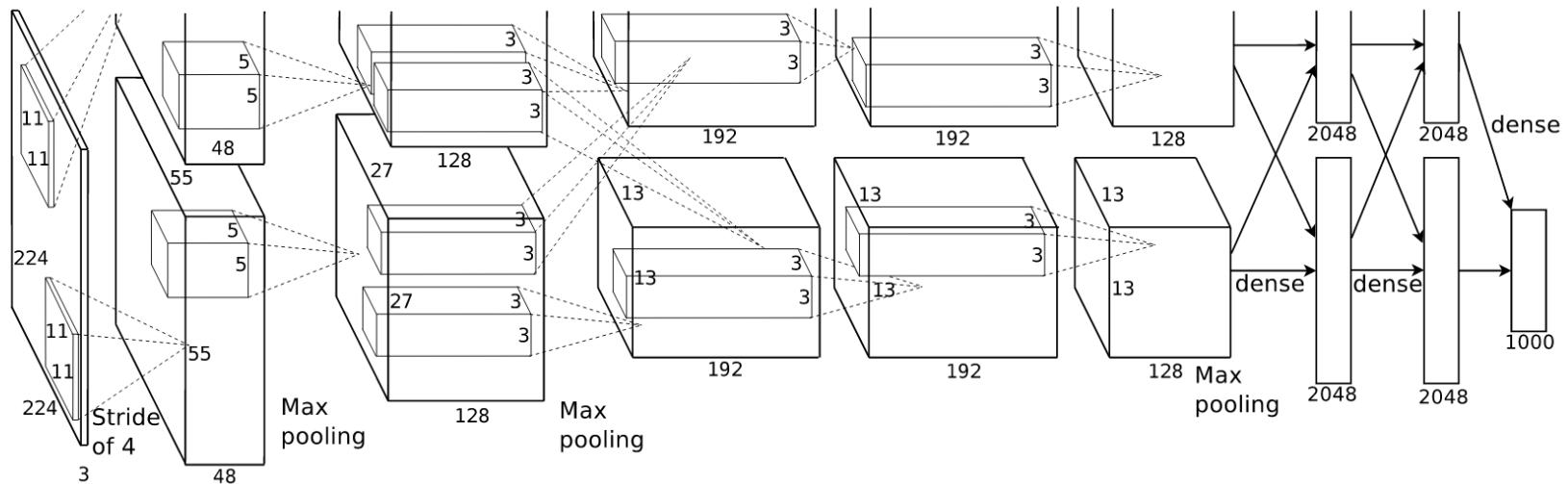


GPU Acceleration



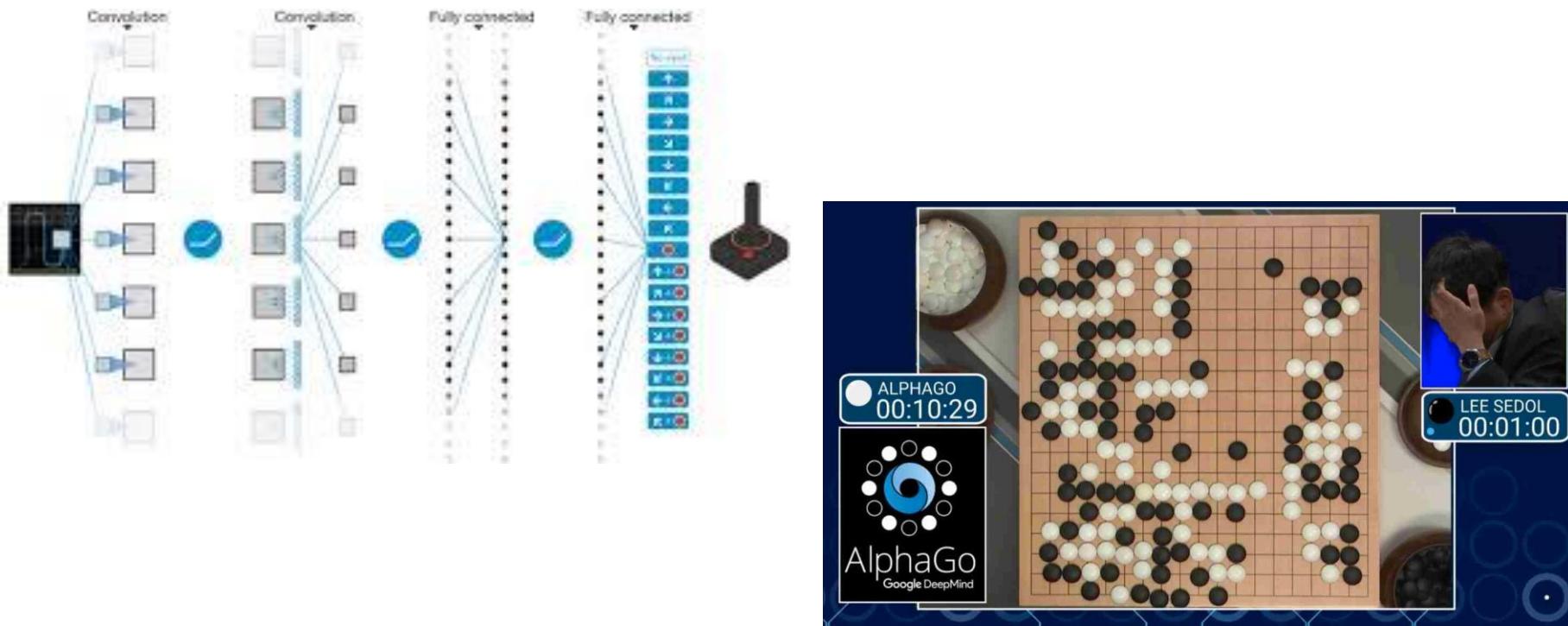
History

- 3세대: 2006~2012
- 2012년 Alex Krizhevsky, Geoffrey Hinton 의 AlexNet 제안
- ILSVRC-2012 대회 압도적 격차로 우승



History

- 4세대: 2012~current
- DeepMind의 DQN(Deep-Q-Networks)
 - DNN + Reinforcement Learning
 - 알파고



History

- 4세대: 2012~current
- Generative Model 연구 활발히 진행 중
 - Generative Adversarial Networks(GAN)



a)



b)



c)



d)



케라스 소개

- 파이썬 딥러닝 프레임 워크
 - 거의 모든 종류의 딥러닝 모델을 간편하게 만들 수 있음
 - 동일한 코드로 CPU와 GPU 실행 가능
 - 사용하기 쉬운 API
 - 딥러닝 모델의 프로토타입을 빠르게 만들 수 있음
 - 합성곱 신경망, 순환 신경망 등 지원
 - 다중 입력, 다중 출력, 층의 공유, 모델 공유 등 어떠한 네트워크 구조도 쉽게 만들 수 있음
 - GAN, 뉴럴 튜닝 머신 등...
 - MIT 라이선스를 따르고 상업적으로 자유롭게 사용 가능



데이터 표현

- 텐서
 - (숫자)데이터를 위한 컨테이너
 - 다차원 배열
 - 최근 머신러닝 시스템은 텐서를 기본 데이터 구조로 사용

- 스칼라
 - 0차원 텐서
 - 숫자 하나만 담고 있음
 - 넘파이 `ndim ==` 텐서 차원 `== rank`



데이터 표현

- 벡터
 - 1차원 텐서
 - 하나의 축만 가짐
 - 최근 머신러닝 시스템은 텐서를 기본 데이터 구조로 사용
- 행렬
 - 2차원 텐서
 - 2개의 축 (행, 열)



데이터 표현

- 3차원 텐서와 고차원 텐서
 - 2차원에 하나의 새로운 배열을 합치면 3차원 텐서가 됨
 - 3차원에 배열을 합치면 4차원이 되고 N차원까지 가능
 - 딥러닝에서는 일반적으로 0차원에서 4차원까지 다름
 - 동영상인 경우 5차원까지 다름
- 텐서의 속성
 - 축의 개수(랭크)
 - 크기(shape)
 - 데이터 타입(dtype)



데이터 표현

- 텐서의 속성
 - 축의 개수(랭크)
 - 크기(shape)
 - 데이터 타입(dtype)

```
from keras.datasets import mnist  
(train_images, train_label), (test_images, test_label) = mnist.load_data()
```

Using TensorFlow backend.

```
print(train_images.ndim)  
print(train_images.shape)  
print(train_images.dtype)
```

3
(60000, 28, 28)
uint8



데이터 표현

- 텐서 연산
 - Dense 층
 - `keras.layers.Dense(512, activation='relu')`
 - $\Rightarrow \text{output} = \text{relu}(\text{dot}(w, \text{input}) + b)$
 - 브로드캐스팅
 - 큰 텐서의 `ndim`에 맞도록 작은 텐서에 축이 추가
 - 작은 텐서가 새 축을 따라서 큰 텐서의 크기에 맞도록 반복
 - 예) $(32, 10) + (10,) \Rightarrow (32, 10) + (32, 10)$
 - 텐서 크기 변환
 - `train_images = train_images.reshape(60000, 28*28)`



신경망 구조

- 훈련 요소
 - 네트워크(혹은 모델)를 구성하는 **층**
 - 입력 데이터와 그에 상응하는 **타깃**
 - 학습에 사용할 피드백 신호를 정의하는 **손실함수**
 - 학습 진행 방식을 결정하는 **옵티마이저**



신경망 구조

■ 훈련 요소

- 네트워크
- 입력
- 학습(교차 엔트로피)
- 학습(최소화)

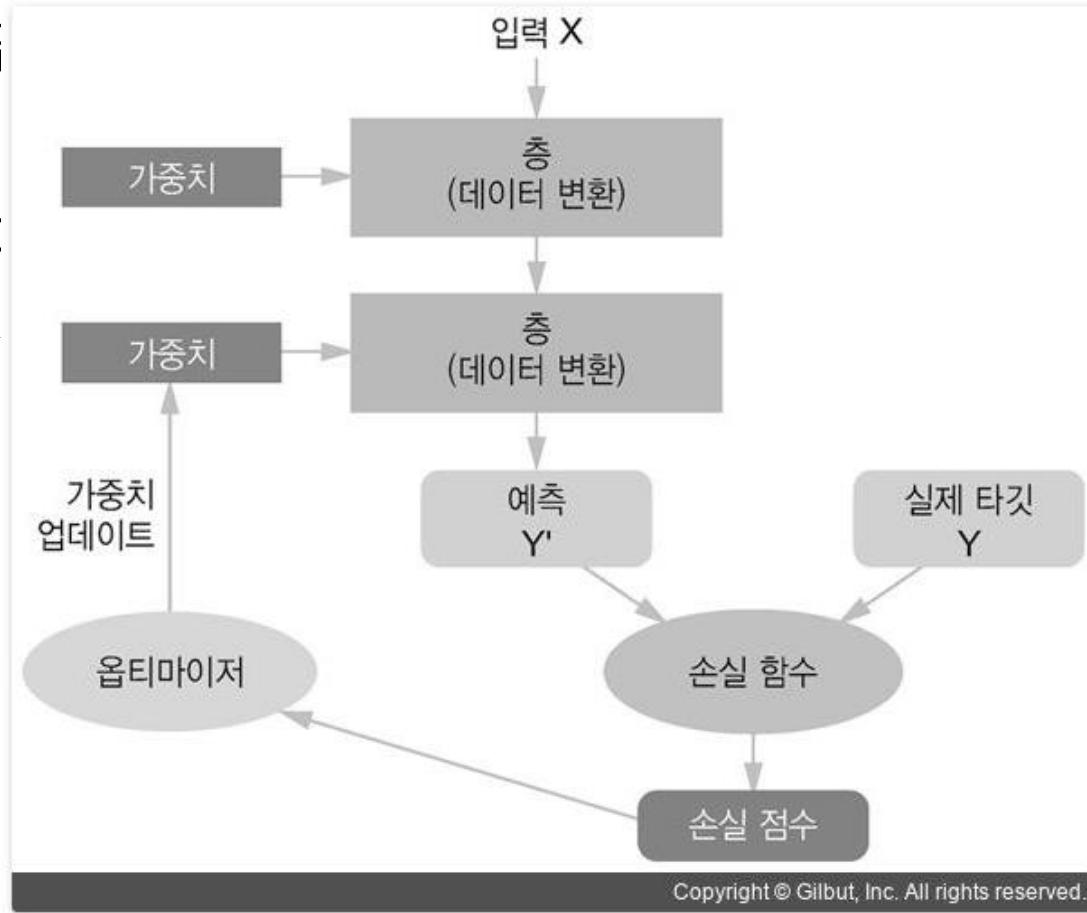


그림 3-1 네트워크, 층, 손실 함수, 옵티마이저 사이의 관계

신경망 구조

- 층

- 딥러닝의 구성 단위
- 하나 이상의 텐서를 입력 받아 하나 이상의 텐서를 출력하는 데이터 처리 모듈
- 대부분 가중치라는 층의 상태를 가짐
 - 상태가 없는 층: flatten, pooling, dropout, ...
- 완전 연결층, 밀집 층(dense layer), 순환층(recurrent layer)
합성곱 층
 - from keras import layers
 - layer = layers.Dense(32, input_shape=(784,))
32개 유닛으로 된 밀집층, 0번째 축 지정하지 않음
어떤 배치 크기도 입력 받을 수 있음



신경망 구조

- 층
 - 완전 연결층, 밀집 층(dense layer), 순환층(recurrent layer) 합성곱 층
 - from keras import layers
 - from keras import models
 - layer = layers.Dense(32, input_shape=(784,))
 - model = models.Sequential()
 - model.add(layers.Dense(32, input_shape=(784,)))
1st layer
 - model.add(layers.Dense(10))
2nd layer, input_shape을 지정하지 않으면 자동으로 채택



신경망 구조

- 모델
 - 딥러닝 모델은 층으로 만든 비순환 유향 그래프 (DAG, Directed Acyclic Graph, 엣지에 방향이 있고 사이클이 없는 그래프)
 - 네트워크 구조는 가설공간 정의
 - 머신러닝이란 결국 **가능성이 있는 공간을 사전에 정의**하고 피드백 신호의 도움을 받아 **입력 데이터에 대한 유용한 변환을 찾는 것**
 - 네트워크 구조 선택 가능성 있는 공간을 입력 데이터에서 출력 데이터로 매핑하는 일련의 특정 텐서 연산으로 제한
 - 이런 텐서 연산에 포함된 가중치 텐서를 찾는 것이 목표



케라스 코드 구조

■ 신경망 구현 순서

- **Sequential** 모형 클래스 객체 생성
- **add(layer)** 모델에 레이어 추가
 - 입력단부터 순차적으로 추가
 - 레이어의 첫 번째 인수는 출력 뉴런 개수
 - 최초의 레이어는 `input_dim` 인수로 입력 크기를 설정해야 함
 - `activation` 인수로 활성화 함수 설정
- **compile** 모형 완성
 - `loss` 비용 함수 설정
 - `optimizer` 최적화 알고리즘 설정
 - `metrics` 훈련 단계의 성능 기준
- **fit** 훈련 실행
 - `nb_epoch` epoch 횟수 설정
 - `batch_size` 배치 크기 설정
 - `verbose` 학습 중 출력되는 문구 설정
 - 주피터 노트북 사용시 `verbose=2`로 설정하여 진행 막대가 나오지 않도록 설정



첫 번째 케라스 예제

■ MNIST 문제

■ 데이터 로딩

```
import keras
import matplotlib.pyplot as plt
from keras import models
from keras import layers
from keras.utils import to_categorical

keras.__version__

Using TensorFlow backend.

'2.2.2'

from keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

print(train_images.shape)
print(len(train_labels))
print(train_labels)

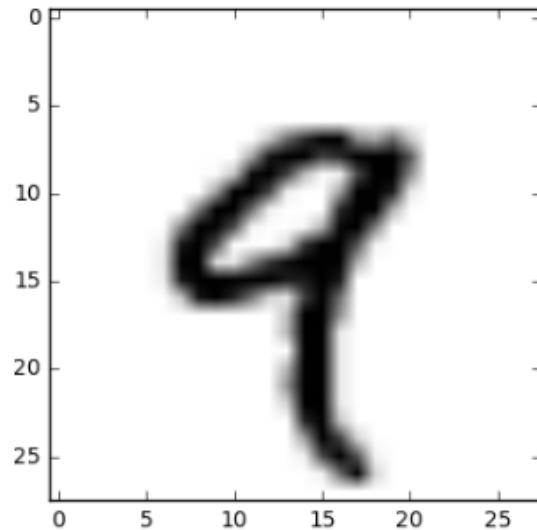
print(test_images.shape)
print(len(test_labels))
print(test_labels)

(60000, 28, 28)
60000
[5 0 4 ... 5 6 8]
(10000, 28, 28)
10000
[7 2 1 ... 4 5 6]
```

첫 번째 케라스 예제

- MNIST 문제
 - 네트워크 모델링

```
digit = train_images[4]  
  
plt.imshow(digit, cmap=plt.cm.binary)  
plt.show()
```



```
network = models.Sequential()  
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))  
network.add(layers.Dense(10, activation='softmax'))  
network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```



첫 번째 케라스 예제

■ MNIST 문제

■ 훈련 및 테스트

```
# 훈련 시 0과 1 사이의 값을 가지는 float32 타입의 (60000, 28 * 28) 배열로 변환
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
```

```
# 레이블을 벙주형으로 인코딩
# 훈련 데이터가 n개, 클래스가 k개 일때 to_categorical 함수는 입력받은 n크기의 1차원 정수 배열을 (n, k)크기의 2차원 배열로 변경
# 이 배열의 두번째 차원의 인덱스가 클래스 값 의미
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

```
# 훈련 데이터로 모델 학습
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
Epoch 1/5
60000/60000 [=====] - 4s 73us/step - loss: 0.2561 - acc: 0.9271
Epoch 2/5
60000/60000 [=====] - 4s 70us/step - loss: 0.1043 - acc: 0.9688
Epoch 3/5
60000/60000 [=====] - 4s 63us/step - loss: 0.0679 - acc: 0.9797
Epoch 4/5
60000/60000 [=====] - 4s 61us/step - loss: 0.0491 - acc: 0.9853
Epoch 5/5
60000/60000 [=====] - 4s 61us/step - loss: 0.0372 - acc: 0.9887
<keras.callbacks.History at 0x1d158557da0>
```

```
# 테스트 데이터로 정확도 측정
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
```

```
10000/10000 [=====] - 0s 37us/step
test_acc: 0.9777
```



두 번째 케라스 예제

■ Iris 문제

■ 데이터 로딩

```
# Multiclass Classification with the Iris Flowers Dataset
import numpy
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from keras.utils import to_categorical
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder

Using TensorFlow backend.
```

```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

# load dataset
dataframe = read_csv("iris.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

print(X.shape)
print(Y.shape)
print(Y[0])

(150, 4)
(150,)
Iris-setosa
```

두 번째 케라스 예제

■ Iris 문제

■ 데이터 로딩

```
# encode class values as integers
# 3개의 클래스 정보를 원핫코딩으로 변환
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)

# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)

# define baseline model
def baseline_model():

    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=4, activation='relu'))
    model.add(Dense(3, activation='softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

estimator = KerasClassifier(build_fn=baseline_model, epochs=200, batch_size=5, verbose=0)
# k-fold 검증
kfold = KFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, dummy_y, cv=kfold)
print("Accuracy: %.2f%%" % (results.mean()*100))
```

Accuracy: 96.67%

Multi classification 예제

- 뉴스 기사 분류
 - 로이터 데이터셋
 - 1986년 로이터에서 공개한 짧은 뉴스 기사화 토픽의 집합
 - 텍스트 분류를 위해 널리 사용되는 간단한 데이터셋
 - 46개 토픽, 토픽 별로 최소 10개 이상 샘플
 - IMDB 문제와 유사하게 문제 해결
 - 다른 점
 - 은닉층 노드 수
 - 출력 노드 수
 - 활성화 함수



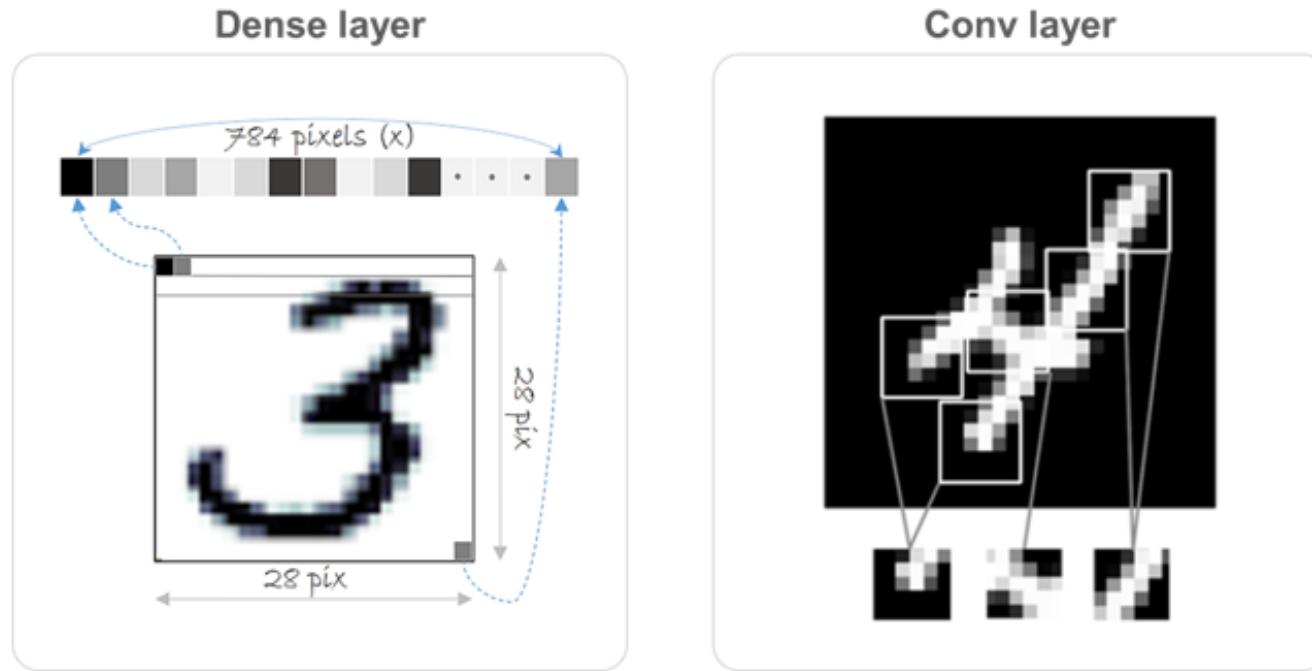
Multi classification 예제

- 뉴스 기사 분류
 - 추가 실험하기
 - 은닉층의 개수 변경 (1 or 3)
 - 층의 유닛(노드) 수 변경 (32 or 128)



합성곱 신경망

- Dense층 vs. 합성곱 층
 - Dense 층: 입력 특성 공간에 있는 전역 패턴 학습
 - 합성곱 층: 지역 패턴 학습



출처: <https://circle.haus/t/chap05-1/121>

합성곱 신경망

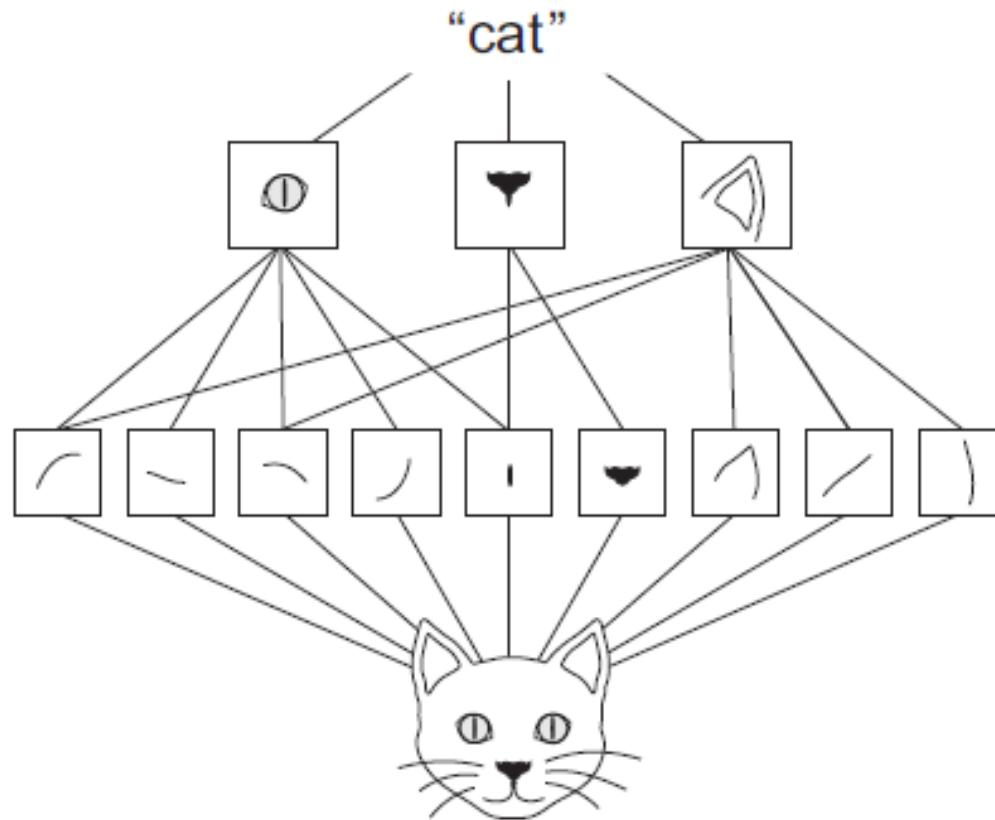
■ 특징

- 학습된 패턴은 평행 이동 불변성을 가짐
 - 컨브넷이 이미지 오른쪽 아래 모서리에서 어떤 패턴을 학습했다면 다른 곳에서도 이 패턴을 인식할 수 있음
 - 완전 연결 네트워크는 새로운 위치에 나타난 것은 새로운 패턴으로 학습해야 함
- 컨브넷은 패턴의 공간적 계층 구조를 학습 할 수 있음
 - 첫 번째 합성곱 층은 엣지 같은 작은 지역 패턴 학습
 - 두 번째 합성곱 층은 첫 번째보다 큰 패턴을 학습
 - 이는 매우 복잡하고 추상적인 시각적 개념을 효과적으로 학습



합성곱 신경망

- 특징



합성곱 신경망

- 특징

- 연산 결과는 특성맵이라고 부르는 3D 텐서에 적용
 - 2개의 공간 축 (높이, 너비), 깊이 축 (채널)
 - 출력 텐서의 깊이 축 채널은 일종의 필터를 의미
 - 입력에 대한 필터의 응답맵 이기도 함
- 핵심 파라미터
 - 패치의 크기: 3×3 or 5×5
 - 특성맵의 출력 깊이: 필터 수

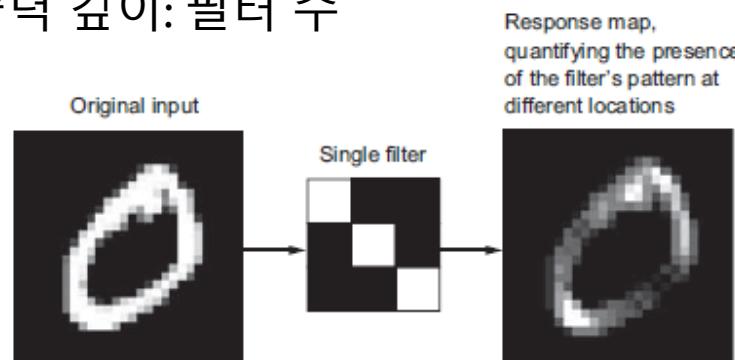


Figure 5.3 The concept of a response map: a 2D map of the presence of a pattern at different locations in an input

합성곱 신경망

■ 특징

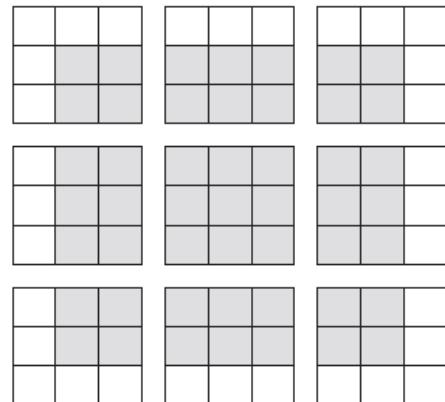
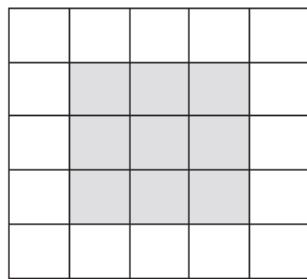


Figure 5.5 Valid locations of 3×3 patches in a 5×5 input feature map

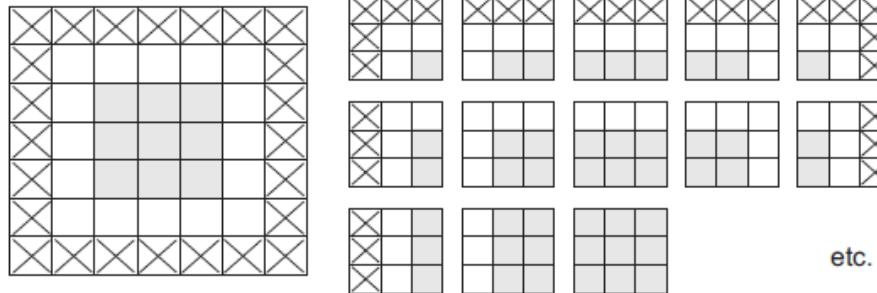


Figure 5.6 Padding a 5×5 input in order to be able to extract $25 3 \times 3$ patches

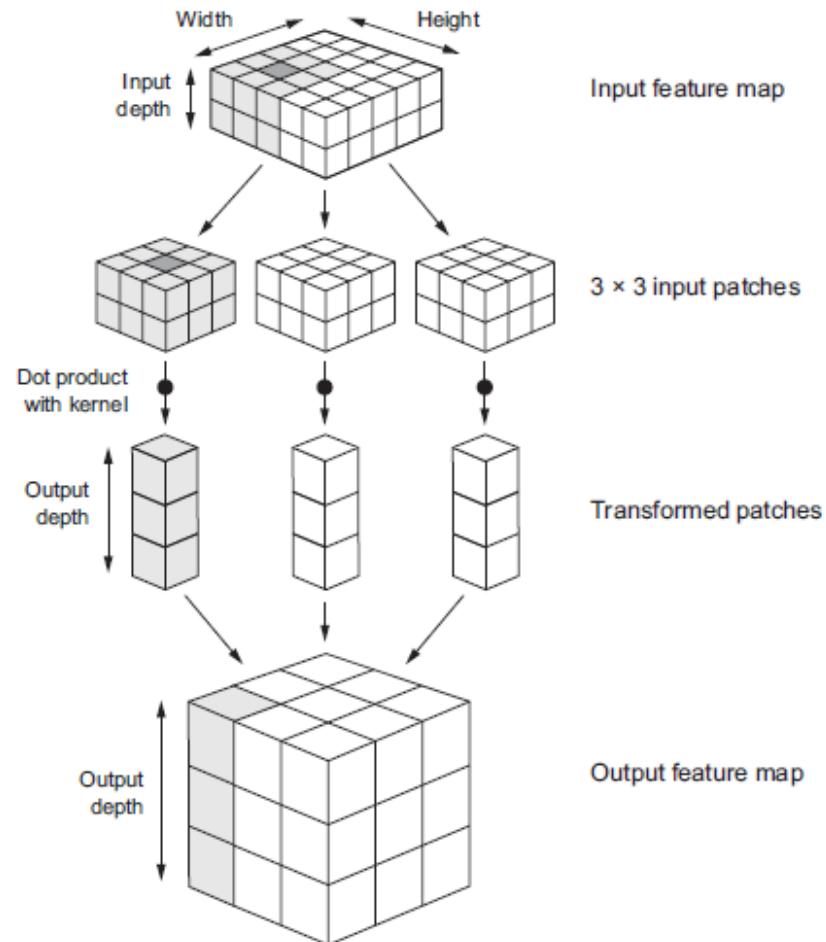


Figure 5.4 How convolution works

합성곱 신경망

- 최대 풀링 연산
 - 최대 풀링 연산을 사용하지 않는다면?
 - 특성의 공간적 계층 구조를 학습하는 데 도움이 되지 않음
 - 가중치가 너무 많아짐(오버피팅)

```
model_no_max_pool = models.Sequential()
model_no_max_pool.add(layers.Conv2D(32, (3, 3), activation='relu',
                                   input_shape=(28, 28, 1)))
model_no_max_pool.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_no_max_pool.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Here's a summary of the model:

```
>>> model_no_max_pool.summary()
```

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------------|---------|
| conv2d_4 (Conv2D) | (None, 26, 26, 32) | 320 |
| conv2d_5 (Conv2D) | (None, 24, 24, 64) | 18496 |
| conv2d_6 (Conv2D) | (None, 22, 22, 64) | 36928 |
| Total params: 55,744 | | |
| Trainable params: 55,744 | | |
| Non-trainable params: 0 | | |

합성곱 신경망

■ MNIST 구현

■ Convnet 빌드

```
import keras
keras.__version__
from keras import layers
from keras import models

model = models.Sequential()

#input_shape = (image_height, image_width, image_channels)
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d_4 (Conv2D) | (None, 26, 26, 32) | 320 |
| max_pooling2d_3 (MaxPooling2D) | (None, 13, 13, 32) | 0 |
| conv2d_5 (Conv2D) | (None, 11, 11, 64) | 18496 |
| max_pooling2d_4 (MaxPooling2D) | (None, 5, 5, 64) | 0 |
| conv2d_6 (Conv2D) | (None, 3, 3, 64) | 36928 |
| flatten_3 (Flatten) | (None, 576) | 0 |
| dense_3 (Dense) | (None, 64) | 36928 |
| dense_4 (Dense) | (None, 10) | 650 |
| Total params: 93,322 | | |
| Trainable params: 93,322 | | |
| Non-trainable params: 0 | | |

합성곱 신경망

■ 구현

■ MNIST 데이터

```
from keras.datasets import mnist
from keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

Epoch 1/5
60000/60000 [=====] - 38s 627us/step - loss: 0.1807 - acc: 0.9445
Epoch 2/5
60000/60000 [=====] - 36s 607us/step - loss: 0.0470 - acc: 0.9847
Epoch 3/5
60000/60000 [=====] - 35s 579us/step - loss: 0.0323 - acc: 0.9907
Epoch 4/5
60000/60000 [=====] - 37s 610us/step - loss: 0.0247 - acc: 0.9925
Epoch 5/5
60000/60000 [=====] - 35s 590us/step - loss: 0.0198 - acc: 0.9941

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)

10000/10000 [=====] - 0s 32us/step
```

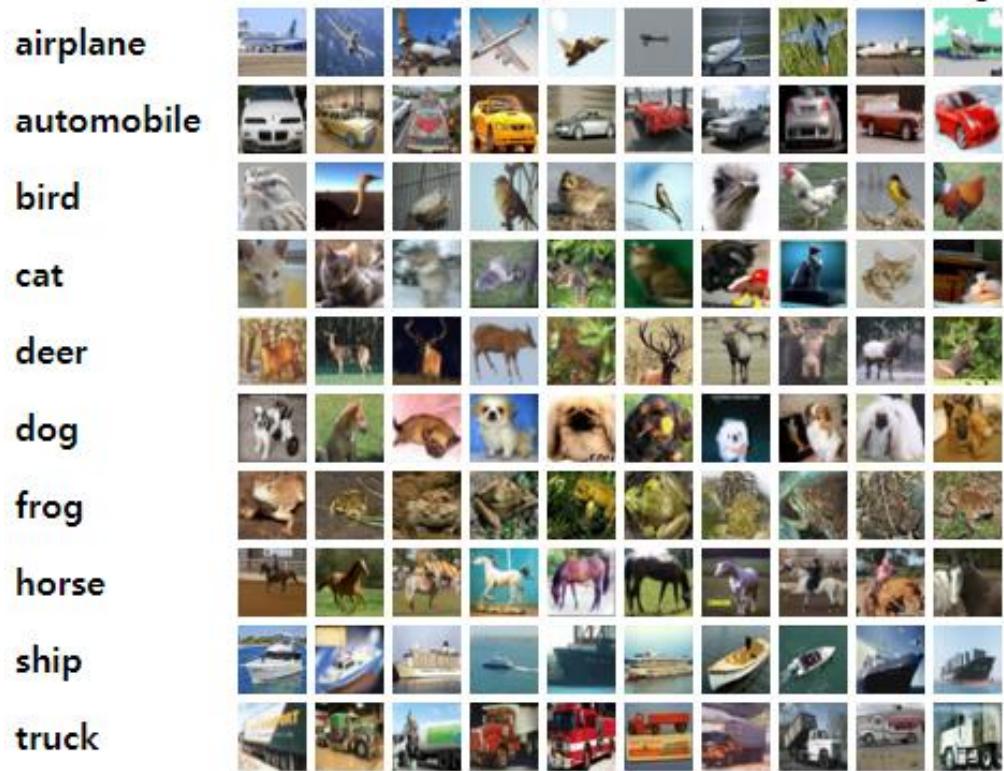
```
test_acc
```

0.9921

CIFAR-10 이미지 분류기

■ CIFAR-10 데이터

- $32 \times 32 \times 3$ 크기의 컬러영상
- 학습용: 50,000개
- 테스트용: 10,000개
- 10개의 클래스
- 정수 레이블 저장



출처: <https://www.cs.toronto.edu/~kriz/cifar.html>

CIFAR-10 인식 문제

■ 데이터 로딩

```
# Simple CNN model for the CIFAR-10 Dataset
import numpy
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
K.set_image_dim_ordering('th')
```

Using TensorFlow backend.

```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```



CIFAR-10 인식 문제

■ 네트워크 모델링, 훈련 및 테스트

```
# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

```
# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.summary()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

화이트 보드



참고 자료

- Introduction to Machine Learning with Python (파이썬 라이브러리를 활용한 머신러닝)
 - 안드레아스 밀러, 세라 가이도 지음 / 박해선 옮김
 - 한빛미디어, 2019
- Hands-On Machine Learning with Scikit-Learn & Tensorflow (핸즈온 머신러닝)
 - 오렐리앙 제롱 / 박해선 옮김
 - 한빛미디어, 2018
- Machine Learning 기계학습
 - 오일석 지음
 - 한빛 아카데미, 2017
- 패턴인식
 - 오일석 지음
 - 교보문고, 2008



참고 자료

- Deep Learning with Python (케라스 창시자에게 배우는 딥러닝)
 - 프랑소와 솔레 / 박해선 옮김
 - 길벗, 2018
- 텐서플로로 배우는 딥러닝
 - 솔라리스 지음
 - 영진닷컴, 2018
- Stanford Course
 - CS231n: Convolutional Neural Networks for Visual Recognition
 - <http://cs231n.stanford.edu/index.html>
- Coursera Course
 - Neural Networks and Deep Learning
 - Andrew Ng
 - <https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome>

