

Machine Learning 이해와 활용

가톨릭대학교 미디어기술콘텐츠학과
강호철

강의 내용

- 머신러닝 소개
 - 머신러닝 개요
 - 머신러닝 소개 및 실습
- 지도학습
 - 선형모델
 - 나이브 베이즈 분류기
 - 결정트리
- 비지도 학습
 - 데이터 전처리와 스케일 조정
 - 군집화



머신러닝 개요



1.1.1 기계 학습의 정의

■ 학습이란? <표준국어대사전>

“경험의 결과로 나타나는, 비교적 지속적인 행동의 변화나 그 잠재력의 변화, 또는 지식을 습득하는 과정[국립국어원2017]”

■ 기계 학습이란?

▪ 인공지능 초창기 사무엘의 정의

“Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort. 컴퓨터가 경험을 통해 학습할 수 있도록 프로그래밍할 수 있다면, 세세하게 프로그래밍해야 하는 번거로움에서 벗어날 수 있다[Samuel1959].”

1.1.1 기계 학습의 정의

■ 기계 학습이란?

- 현대적 정의

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . 어떤 컴퓨터 프로그램이 T 라는 작업을 수행한다. 이 프로그램의 성능을 P 라는 척도로 평가했을 때 경험 E 를 통해 성능이 개선된다면 이 프로그램은 학습을 한다고 말할 수 있다[Mitchell1997(2쪽)].”

“Programming computers to optimize a performance criterion using example data or past experience 사례 데이터, 즉 과거 경험을 이용하여 성능 기준을 최적화하도록 프로그래밍하는 작업[Alpaydin2010]”

“Computational methods using experience to improve performance or to make accurate predictions 성능을 개선하거나 정확하게 예측하기 위해 경험을 이용하는 계산학 방법들[Mohri2012]”

1.1.2 지식기반 방식에서 기계 학습으로의 대전환

■ 인공지능의 탄생

- 컴퓨터의 뛰어난 능력
 - 사람이 어려워하는 일을 아주 쉽게 함
 - $80932.46789076 * 0.39001324$ 와 같은 곱셈을 고속으로 수행(현재는 초당 수십억개)
 - 복잡한 함수의 미분과 적분 척척
- 컴퓨터에 대한 기대감 (컴퓨터의 능력 과신)
 - 사람이 쉽게 하는 일, 예를 들어 고양이/개 구별하는 일도 잘 하지 않을까
 - 1950년대에 인공지능이라는 분야 등장

■ 초창기는 지식기반 방식이 주류

- 예) "구멍이 2개이고 중간 부분이 훌쭉하며, 맨 위와 아래가 둉근 모양이라면 8이다"

1.1.2 지식기반 방식에서 기계 학습으로의 대전환

■ 큰 깨달음

- 지식기반의 한계
 - 단추를 “가운데 구멍이 몇 개 있는 물체”라고 규정하면 많은 오류 발생



그림 1-2 인식 시스템이 대처해야 하는 심한 변화 양상(8과 단추라는 패턴을 어떻게 기술할 것인가?)

- 사람은 변화가 심한 장면을 아주 쉽게 인식하지만, 왜 그렇게 인식하는지 서술하지는 못함

1.1.2 지식기반 방식에서 기계 학습으로의 대전환

■ 인공지능의 주도권 전환

- 지식기반 → 기계 학습
- 기계 학습: [데이터 중심](#) 접근방식



그림 1-3 기계 학습으로 만든 최첨단 인공지능 제품들

1.1.3 기계 학습 개념

■ 간단한 기계 학습 예제

- 가로축은 시간, 세로축은 이동체의 위치
- 관측한 4개의 점이 데이터

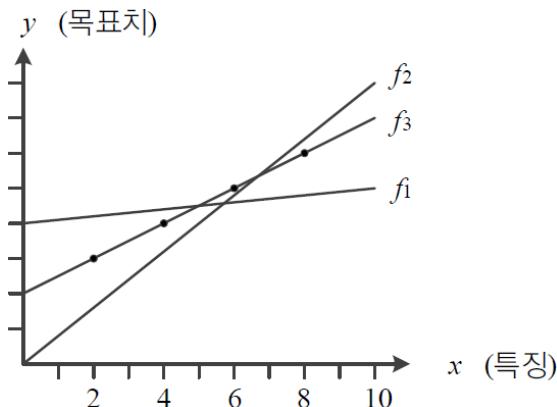


그림 1-4 간단한 기계 학습 예제

■ 예측 prediction 문제

- 임의의 시간이 주어지면 이때 이동체의 위치는?
- 회귀 regression 문제와 분류 classification 문제로 나눔
 - 회귀는 목표치가 실수, 분류는 부류값 ([그림 1-4]는 회귀 문제)

1.1.3 기계 학습 개념

■ 훈련집합

- 가로축은 특징, 세로축은 목표치
- 관측한 4개의 점이 훈련집합을 구성함

훈련집합: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}, \quad \mathbb{Y} = \{y_1, y_2, \dots, y_n\}$ (1.1)

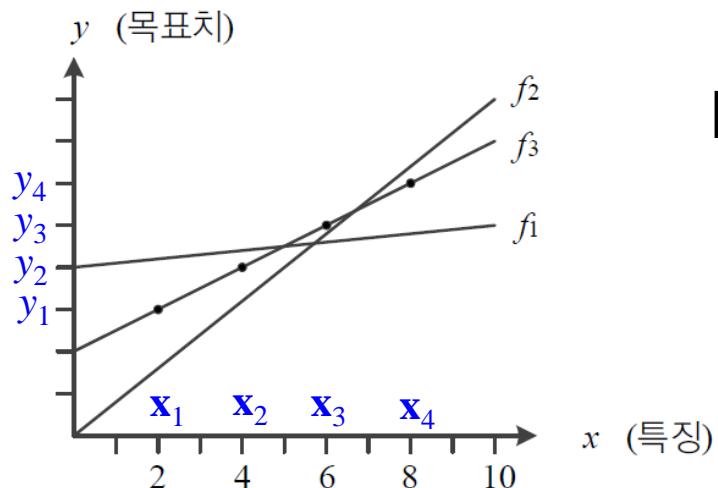


그림 1-4 간단한 기계 학습 예제

[그림 1-4] 예제의 훈련집합

$$\begin{aligned}\mathbb{X} &= \{\mathbf{x}_1 = (2.0), \mathbf{x}_2 = (4.0), \mathbf{x}_3 = (6.0), \mathbf{x}_4 = (8.0)\} \\ \mathbb{Y} &= \{y_1 = 3.0, y_2 = 4.0, y_3 = 5.0, y_4 = 6.0\}\end{aligned}$$

1.1.3 기계 학습 개념

■ 데이터를 어떻게 모델링할 것인가

- 눈대중으로 보면 직선을 이루므로 직선을 선택하자 → 모델로 직선을 선택한 셈
- 직선 모델의 수식
 - 2개의 매개변수 w 와 b

$$y = \underline{wx} + \underline{b} \quad (1.2)$$

■ 기계 학습은

- 가장 정확하게 예측할 수 있는, 즉 최적의 매개변수를 찾는 작업
- 처음에는 최적값을 모르므로 임의의 값에서 시작하고, 점점 성능을 개선하여 최적에 도달
- [그림 1-4]의 예에서는 f_1 에서 시작하여 $f_1 \rightarrow f_2 \rightarrow f_3$
 - 최적인 f_3 은 $w=0.5$ 와 $b=2.0$

1.1.3 기계 학습 개념

- 학습을 마치면,
 - 예측에 사용
 - 예) 10.0 순간의 이동체 위치를 알고자 하면, $f_3(10.0)=0.5*10.0+2.0=7.0$ 이라 예측함
- 기계 학습의 궁극적인 목표
 - 훈련집합에 없는 새로운 샘플에 대한 오류를 최소화 (새로운 샘플 집합: 테스트 집합)
 - 테스트 집합에 대한 높은 성능을 일반화^{generalization} 능력이라 부름

1.1.4 사람의 학습과 기계 학습

표 1-1 사람의 학습과 기계 학습의 비교

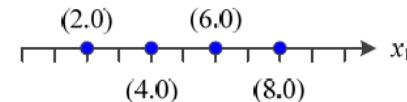
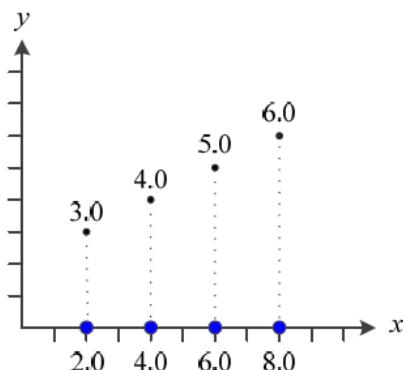
기준	사람의 학습	기계 학습
학습 과정	능동적	수동적
데이터 형식	자연에 존재하는 그대로	일정한 형식에 맞추어 사람이 준비함
동시에 학습 가능한 과업 수	자연스럽게 여러 과업을 학습	하나의 과업만 가능
학습 원리에 대한 지식	매우 제한적으로 알려져 있음	모든 과정이 밝혀져 있음
수학 의존도	매우 낮음	매우 높음
성능 평가	경우에 따라 객관적이거나 주관적	객관적(수치로 평가, 예를 들어 정확률 99.8%)
역사	수백만 년	60년 가량

1.2 특징 공간에 대한 이해

- 1.2.1 1차원과 2차원 특징 공간
- 1.2.2 다차원 특징 공간
- 1.2.3 특징 공간 변환과 표현 학습

1.2.1 1차원과 2차원 특징 공간

■ 1차원 특징 공간



(a) 1차원 특징 공간(왼쪽: 특징과 목푯값을 축으로 표시, 오른쪽: 특징만 축으로 표시)

■ 2차원 특징 공간



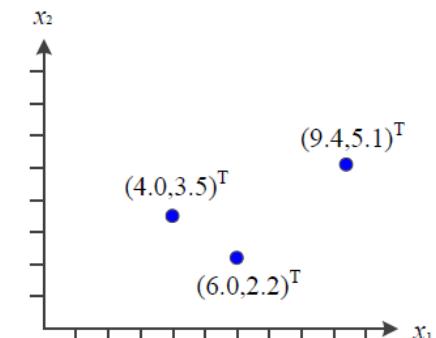
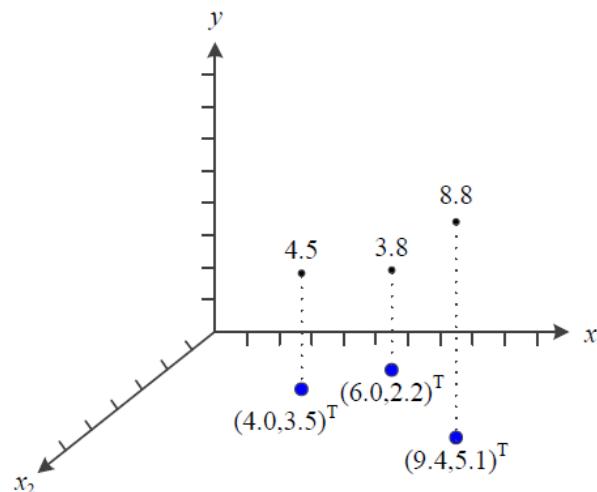
■ 특징 벡터 표기

- $\mathbf{x} = (x_1, x_2)^T$

■ 예시

- $\mathbf{x} = (\text{몸무게}, \text{키})^T, y = \text{장타율}$

- $\mathbf{x} = (\text{체온}, \text{두통})^T, y = \text{감기 여부}$

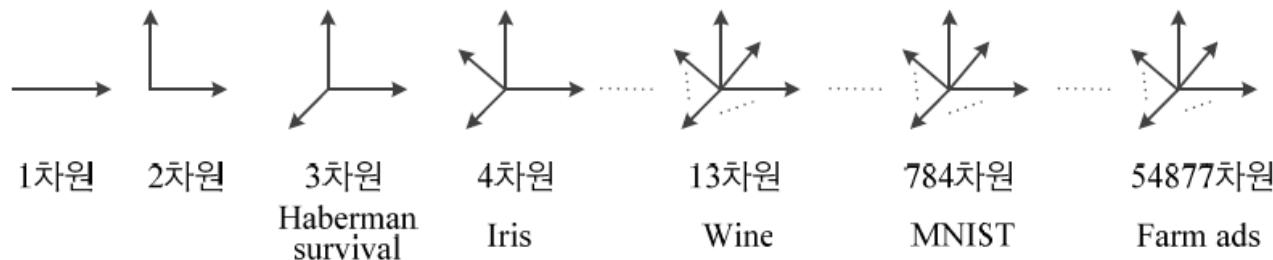


(b) 2차원 특징 공간(왼쪽: 특징 벡터와 목푯값을 축으로 표시, 오른쪽: 특징 벡터만 축으로 표시)

그림 1-5 특징 공간과 데이터의 표현

1.2.2 다차원 특징 공간

■ 다차원 특징 공간 예제



Haberman survival: $\mathbf{x} = (\text{나이}, \text{수술년도}, \text{양성 림프샘 개수})^T$

$$\text{Iris: } \mathbf{x} = (\text{꽃받침 길이}, \text{꽃받침 너비}, \text{꽃잎 길이}, \text{꽃잎 너비})^T$$

Wine: $\mathbf{x} = (\text{Alcohol}, \text{Malic acid}, \text{Ash}, \text{Alcalinity of ash}, \text{Magnesium}, \text{Total phenols}, \text{Flavanoids}, \text{Nonflavanoid phenols})$

Proanthocyanins, Color intensity, Hue, OD₂₈₀ / OD₃₁₅ of diluted wines, Proline)^T

$$\text{MNIST: } \mathbf{x} = (\text{화소1}, \text{화소2}, \dots, \text{화소784})^T$$

그림 1-6 다차원 특징 공간



1.2.2 다차원 특징 공간

■ d -차원 데이터

- 특징 벡터 표기: $\mathbf{x} = (x_1, x_2, \dots, x_d)^T$

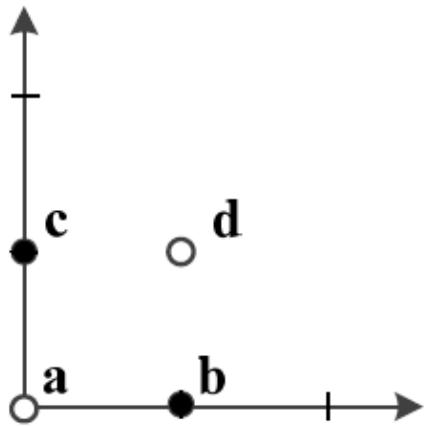
■ d -차원 데이터를 위한 학습 모델

- 직선 모델을 사용하는 경우 매개변수 수 = $d+1$

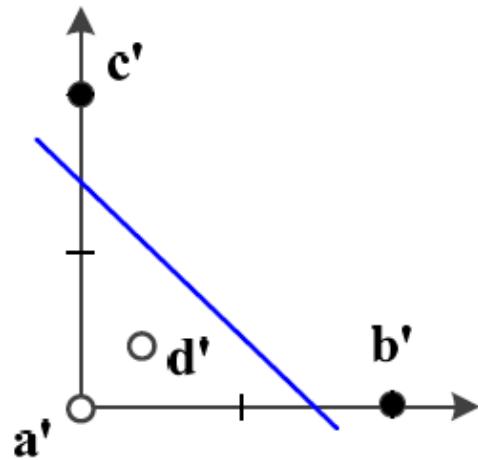
$$y = \underline{w_1}x_1 + \underline{w_2}x_2 + \cdots + \underline{w_d}x_d + \underline{b} \quad (1.3)$$

1.2.3 특징 공간 변환과 표현 학습

- 선형 분리 불가능^{linearly non-separable}한 원래 특징 공간 ([그림 1-7(a)])
 - 직선 모델을 적용하면 75% 정확률이 한계



(a) 원래 특징 공간



(b) 분류에 더 유리하도록 변환된 새로운 특징 공간

그림 1-7 특징 공간 변환

1.2.3 특징 공간 변환과 표현 학습

■ 식 (1.6)으로 변환된 새로운 특징 공간 ([그림 1-7(b)])

- 직선 모델로 100% 정확률

$$\text{원래 특징 벡터 } \mathbf{x} = (x_1, x_2)^T \rightarrow \text{변환된 특징 벡터 } \mathbf{x}' = \left(\frac{x_1}{2x_1x_2 + 0.5}, \frac{x_2}{2x_1x_2 + 0.5} \right)^T \quad (1.6)$$

$$\mathbf{a} = (0,0)^T \rightarrow \mathbf{a}' = (0,0)^T$$

$$\mathbf{b} = (1,0)^T \rightarrow \mathbf{b}' = (2,0)^T$$

$$\mathbf{c} = (0,1)^T \rightarrow \mathbf{c}' = (0,2)^T$$

$$\mathbf{d} = (1,1)^T \rightarrow \mathbf{d}' = (0.4,0.4)^T$$

■ 표현 학습 representation learning

- 좋은 특징 공간을 자동으로 찾는 작업
- 딥러닝은 다수의 은닉층을 가진 신경망을 이용하여 계층적인 특징 공간을 찾아냄
 - 왼쪽 은닉층은 저급 특징(에지, 구석점 등), 오른쪽은 고급 특징(얼굴, 바퀴 등) 추출
- [그림 1-7]은 표현 학습을 사람이 직관으로 수행함

1.2.3 특징 공간 변환과 표현 학습

■ 특징점 간 차이

- 차원에 무관하게 수식 적용 가능함
 - 예) 두 점 $\mathbf{a}=(a_1, a_2, \dots, a_d)^T$ 와 $\mathbf{b}=(b_1, b_2, \dots, b_d)^T$ 사이의 거리는 모든 d 에 대해 성립

$$dist(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^d (a_i - b_i)^2} \quad (1.7)$$

- 보통 2~3차원의 저차원에서 식을 고안한 다음 고차원으로 확장 적용

1.3 데이터에 대한 이해



- 1.3.1 데이터 생성 과정
- 1.3.2 데이터베이스의 중요성
- 1.3.3 데이터베이스 크기와 기계 학습 성능
- 1.3.4 데이터 가시화

1.3 데이터에 대한 이해

■ 과학 기술의 발전 과정

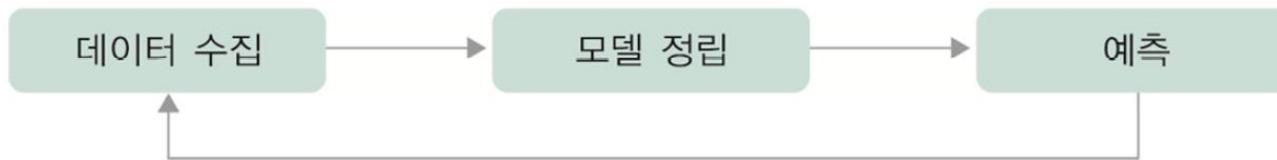


그림 1-8 과학기술의 발전 과정

- 예) 튀코 브라헤는 천동설이라는 틀린 모델을 선택함으로써 자신이 수집한 데이터를 설명하지 못함. 케플러는 지동설 모델을 도입하여 제1, 제2, 제3법칙을 완성함

■ 기계 학습

- 기계 학습이 푸는 문제는 훨씬 복잡함
 - 예) [그림 1-2]의 '8' 숫자 패턴과 '단추' 패턴의 다양한 변화 양상
- 단순한 수학 공식으로 표현 불가능함
- 자동으로 모델을 찾아내는 과정이 필수

1.3.1 데이터 생성 과정

■ 데이터 생성 과정을 완전히 아는 인위적 상황의 예제

- 예) 두 개 주사위를 던져 나온 눈의 합을 x 라 할 때, $y=(x-7)^2+1$ 점을 받는 게임
- 이런 상황을 '데이터 생성 과정을 완전히 알고 있다'고 말함
 - x 를 알면 정확히 y 를 예측할 수 있음
 - 실제 주사위를 던져 $\mathbb{X} = \{3,10,8,5\}$ 를 얻었다면, $\mathbb{Y} = \{17,10,2,5\}$
 - x 의 발생 확률 $P(x)$ 를 정확히 알 수 있음
 - $P(x)$ 를 알고 있으므로, 새로운 데이터 생성 가능

■ [그림 1-6]과 같은 실제 기계 학습 문제

- 데이터 생성 과정을 알 수 없음
- 단지 **주어진 훈련집합 \mathbb{X}, \mathbb{Y} 로 예측 모델 또는 생성 모델을 근사 추정할 수 있을 뿐**

1.3.2 데이터베이스의 중요성

■ 데이터베이스의 품질

- 주어진 응용에 맞는 충분히 다양한 데이터를 충분한 양만큼 수집 → 추정 정확도 높아짐
- 예) 정면 얼굴만 가진 데이터베이스로 학습하고 나면, 기운 얼굴은 매우 낮은 성능
→ 주어진 응용 환경을 자세히 살핀 다음 그에 맞는 데이터베이스 확보는 아주 중요함

■ 아주 많은 공개 데이터베이스

- 기계 학습의 초파리로 여겨지는 3가지 데이터베이스: Iris, MNIST, ImageNet
- 위키피디아에서 'list of datasets for machine learning research'로 검색
- UCI 리퍼지토리 (2017년 11월 기준으로 394개 데이터베이스 제공)

1.3.2 데이터베이스의 중요성

- Iris 데이터베이스는 통계학자인 피셔 교수가 1936년에 캐나다 동부 해안의 가스페 반도에 서식하는 3종의 봇꽃(setosa, versicolor, virginica)을 50송이씩 채취하여 만들었다[Fisher1936]. 150개 샘플 각각에 대해 꽃받침 길이, 꽃받침 너비, 꽃잎 길이, 꽃잎 너비를 측정하여 기록하였다. 따라서 4차원 특징 공간이 형성되며 목푯값은 3종을 숫자로 표시함으로써 1, 2, 3 값 중의 하나이다. <http://archive.ics.uci.edu/ml/datasets/Iris>에 접속하여 내려받을 수 있다.

Sepal length	Sepal width	Petal length	Petal width	Species
5.2	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
7.0	3.2	4.7	1.4	<i>I. versicolor</i>
6.4	3.2	4.5	1.5	<i>I. versicolor</i>
6.9	3.1	4.9	1.5	<i>I. versicolor</i>
5.5	2.3	4.0	1.3	<i>I. versicolor</i>
6.3	3.3	6.0	2.5	<i>I. virginica</i>
5.8	2.7	5.1	1.9	<i>I. virginica</i>
7.1	3.0	5.9	2.1	<i>I. virginica</i>
6.3	2.9	5.6	1.8	<i>I. virginica</i>



Setosa



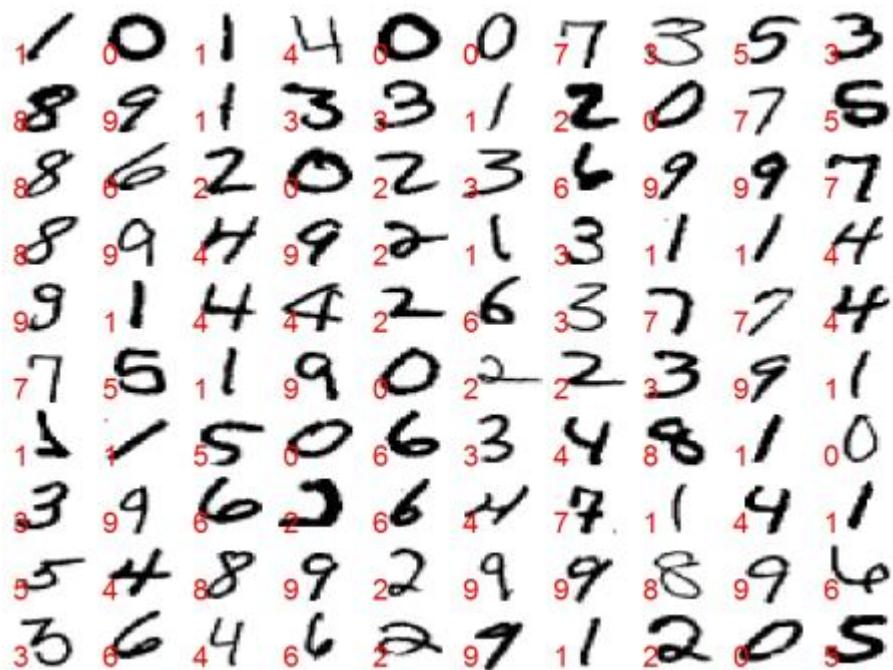
Versicolor



Virginica

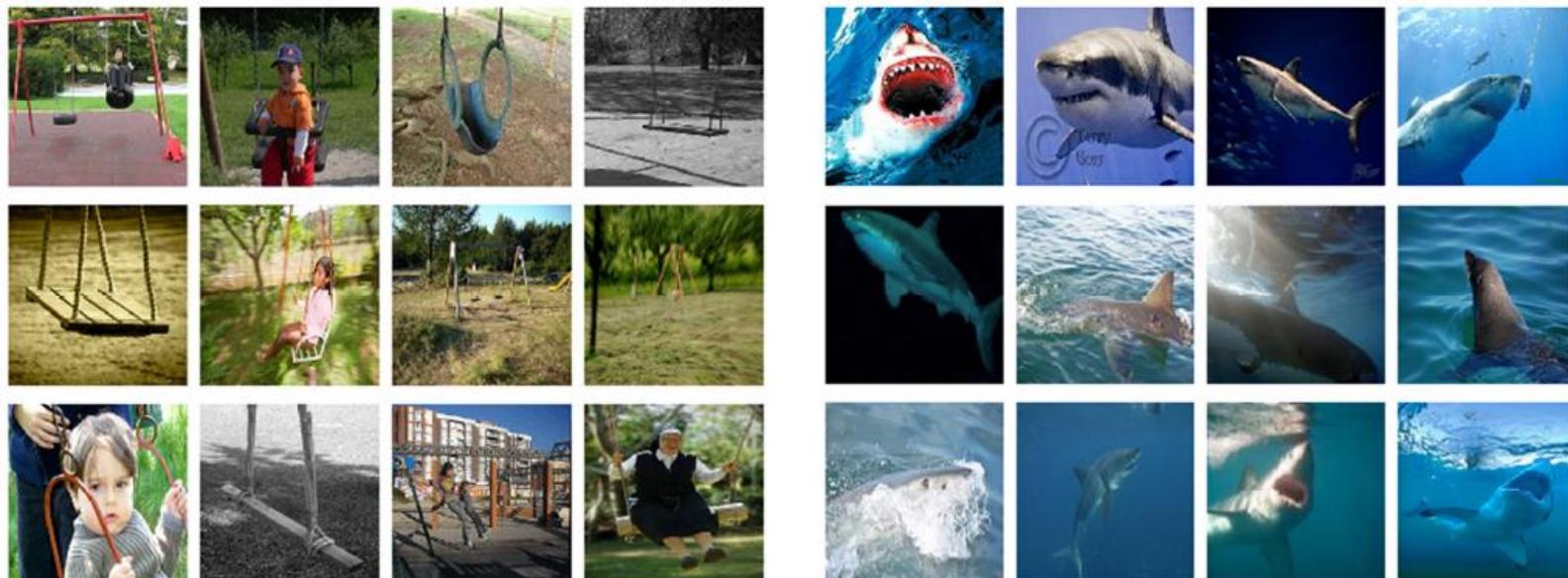
1.3.2 데이터베이스의 중요성

- MNIST 데이터베이스는 미국표준국(NIST)에서 수집한 필기 숫자 데이터베이스로, 훈련집합 60,000자, 테스트집합 10,000자를 제공한다. <http://yann.lecun.com/exdb/mnist>에 접속하면 무료로 내려받을 수 있으며, 1988년부터 시작한 인식률 경쟁 기록도 볼 수 있다. 2017년 8월 기준으로는 [Ciresan2012] 논문이 0.23%의 오류율로 최고 자리를 차지하고 있다. 테스트집합에 있는 10,000개 샘플에서 단지 23개만 틀린 것이다.



1.3.2 데이터베이스의 중요성

- ImageNet 데이터베이스는 정보검색 분야에서 만든 WordNet의 단어 계층 분류를 그대로 따랐고, 부류마다 수백에서 수천 개의 영상을 수집하였다[Deng2009]. 총 21,841개 부류에 대해 총 14,197,122개의 영상을 보유하고 있다. 그중에서 1,000개 부류를 뽑아 ILSVRC(ImageNet Large Scale Visual Recognition Challenge)라는 영상인식 경진대회를 2010년부터 매년 개최하고 있다. 대회 결과에 대한 자세한 내용은 4.4절을 참조하라. <http://image-net.org>에서 내려받을 수 있다.

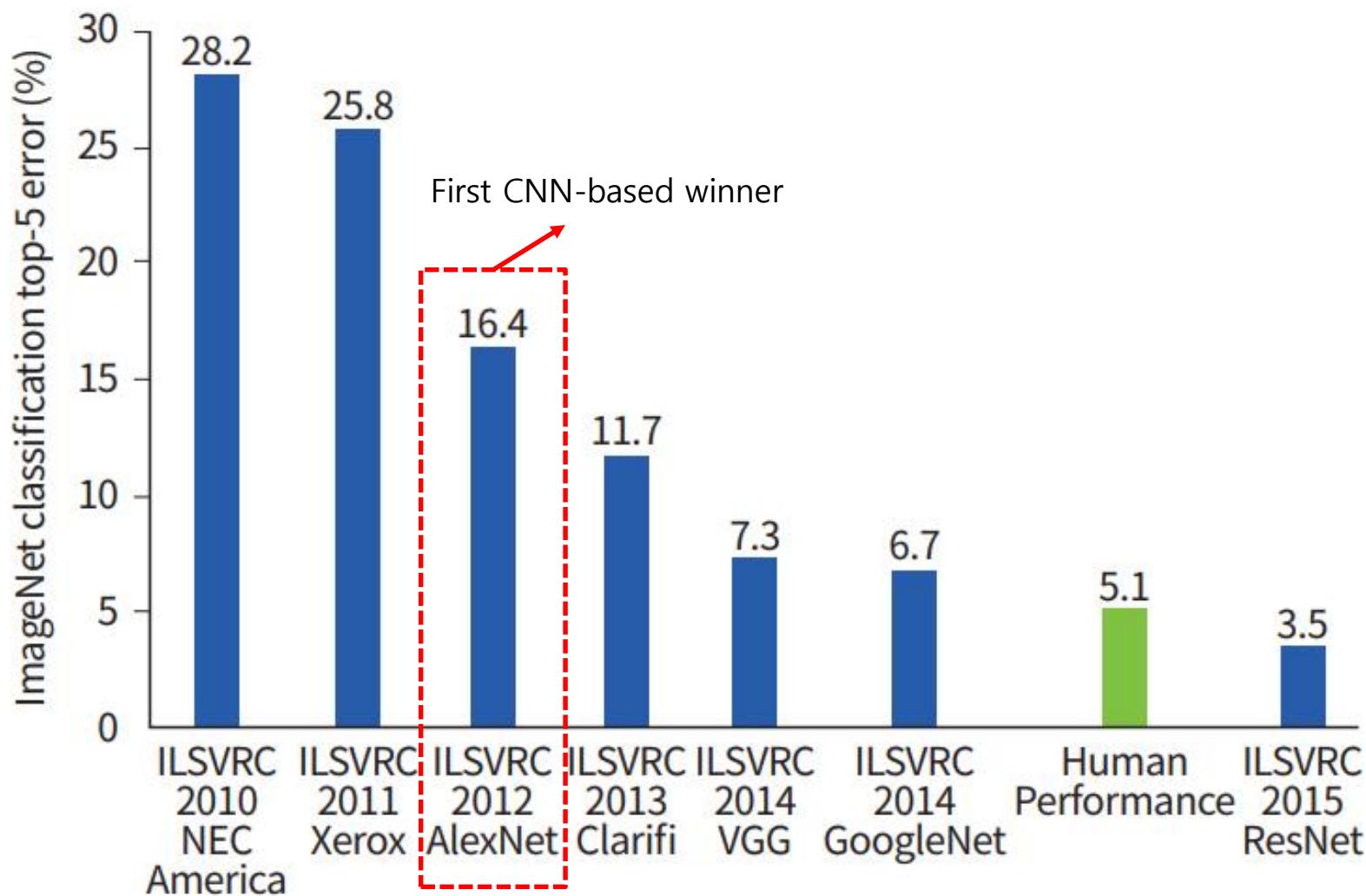


(a) 'swing' 부류

(b) 'Great white shark' 부류

그림 4-20 ImageNet의 예제 영상

1.3.2 데이터베이스의 중요성



1.3.4 데이터 가시화

- 4차원 이상의 초공간은 한꺼번에 가시화 불가능

- 여러 가지 가시화 기법

- 2개씩 조합하여 여러 개의 그래프 그림

그림에 있는 ●=setosa, ○=versicolor, ▲=virginica

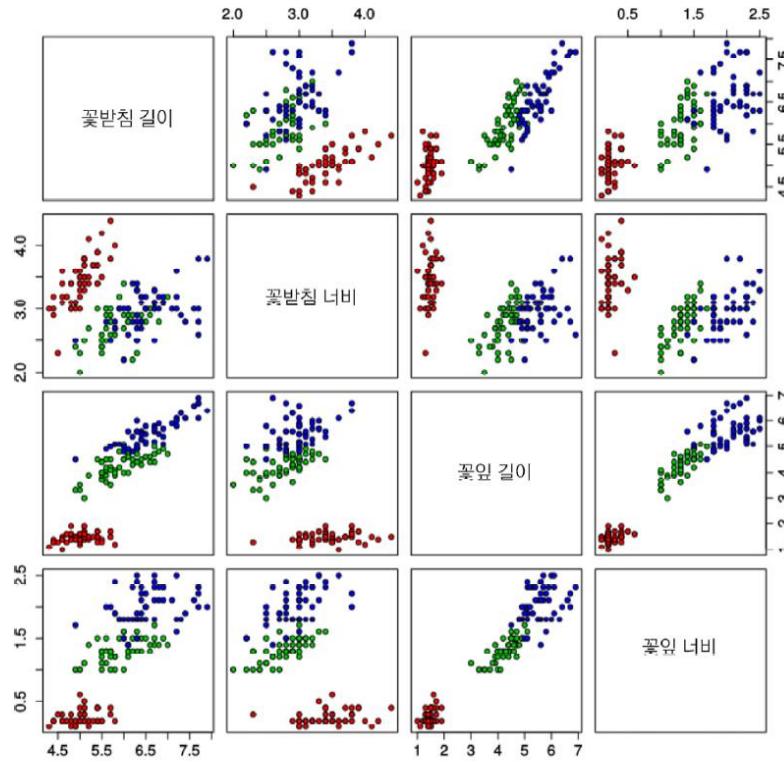


그림 1-10 고차원 특징 공간의 가시화

- 고차원 공간을 저차원으로 변환하는 기법들을 사용하기도 함

1.4 간단한 기계 학습의 예

■ 선형 회귀 문제

- [그림 1-4]: 식 (1.2)의 직선 모델을 사용하므로 두 개의 매개변수 $\Theta = (w, b)^T$

$$y = wx + b \quad (1.2)$$

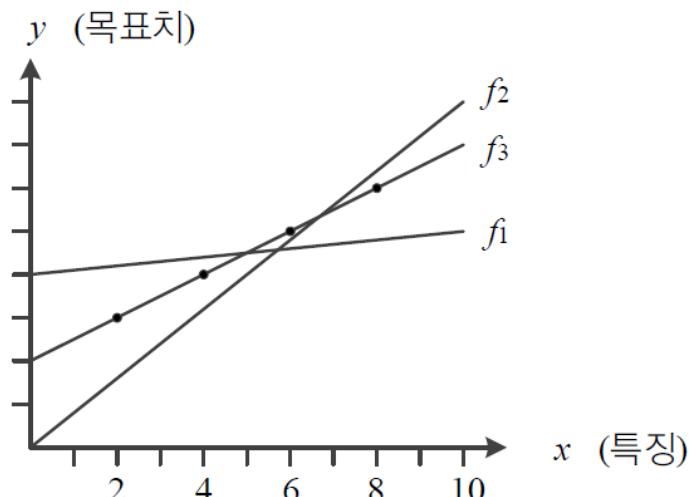


그림 1-4 간단한 기계 학습 예제

1.4 간단한 기계 학습의 예

■ 목적 함수 objective function (또는 비용 함수 cost function, loss function)

- 식 (1.8)은 선형 회귀를 위한 목적 함수

- $f_{\Theta}(\mathbf{x}_i)$ 는 예측함수의 출력, y_i 는 예측함수가 맞추어야 하는 목표값이므로 $f_{\Theta}(\mathbf{x}_i) - y_i$ 는 오차
- 식 (1.8)을 평균제곱오차 MSE(mean squared error)라 부름

$$J(\Theta) = \frac{1}{n} \sum_{i=1}^n (f_{\Theta}(\mathbf{x}_i) - y_i)^2 \quad (1.8)$$

- 처음에는 최적 매개변수 값을 알 수 없으므로 난수로 $\Theta_1 = (w_1, b_1)^T$ 설정 $\rightarrow \Theta_2 = (w_2, b_2)^T$ 로 개선 $\rightarrow \Theta_3 = (w_3, b_3)^T$ 로 개선 $\rightarrow \Theta_3$ 는 최적해 $\hat{\Theta}$
- 이때 $J(\Theta_1) > J(\Theta_2) > J(\Theta_3)$

1.4 간단한 기계 학습의 예

[예제 1-1]

■ 훈련집합

$$\mathbb{X} = \{x_1 = (2.0), x_2 = (4.0), x_3 = (6.0), x_4 = (8.0)\},$$

$$\mathbb{Y} = \{y_1 = 3.0, y_2 = 4.0, y_3 = 5.0, y_4 = 6.0\}$$

■ 초기 직선의 매개변수 $\Theta_1 = (0.1, 4.0)^T$ 라 가정

$$x_1, y_1 \rightarrow (f_{\Theta_1}(2.0) - 3.0)^2 = ((0.1 * 2.0 + 4.0) - 3.0)^2 = 1.44$$

$$x_2, y_2 \rightarrow (f_{\Theta_1}(4.0) - 4.0)^2 = ((0.1 * 4.0 + 4.0) - 4.0)^2 = 0.16$$

$$x_3, y_3 \rightarrow (f_{\Theta_1}(6.0) - 5.0)^2 = ((0.1 * 6.0 + 4.0) - 5.0)^2 = 0.16$$

$$x_4, y_4 \rightarrow (f_{\Theta_1}(8.0) - 6.0)^2 = ((0.1 * 8.0 + 4.0) - 6.0)^2 = 1.44$$

$$\longrightarrow J(\Theta_1) = 0.8$$

1.4 간단한 기계 학습의 예

[예제 1-1] 훈련집합

- Θ_1 을 개선하여 $\Theta_2 = (0.8, 0.0)^T$ 가 되었다고 가정

$$x_1, y_1 \rightarrow (f_{\Theta_2}(2.0) - 3.0)^2 = ((0.8 * 2.0 + 0.0) - 3.0)^2 = 1.96$$

$$x_2, y_2 \rightarrow (f_{\Theta_2}(4.0) - 4.0)^2 = ((0.8 * 4.0 + 0.0) - 4.0)^2 = 0.64$$

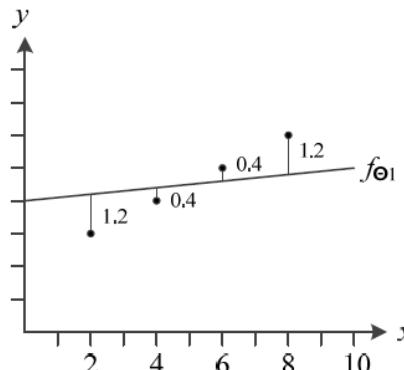
$$x_3, y_3 \rightarrow (f_{\Theta_2}(6.0) - 5.0)^2 = ((0.8 * 6.0 + 0.0) - 5.0)^2 = 0.04$$

$$x_4, y_4 \rightarrow (f_{\Theta_2}(8.0) - 6.0)^2 = ((0.8 * 8.0 + 0.0) - 6.0)^2 = 0.16$$

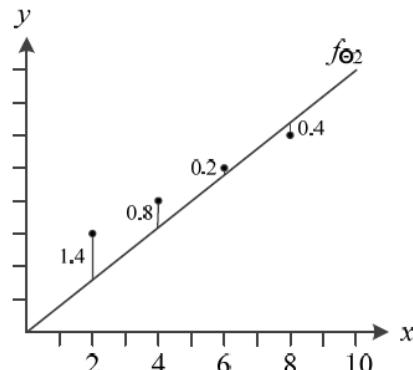
$$\longrightarrow J(\Theta_2) = 0.7$$

- Θ_2 를 개선하여 $\Theta_3 = (0.5, 2.0)^T$ 가 되었다고 가정

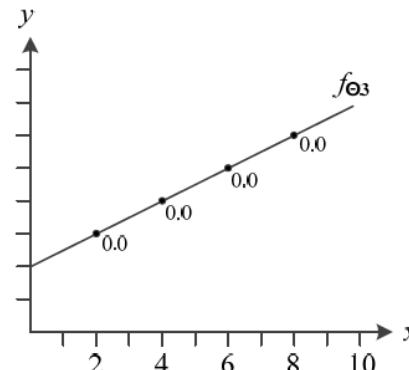
- 이때 $J(\Theta_3) = 0.0$ 이 되어 Θ_3 은 최적값 $\hat{\Theta}$ 이 됨



(a) 초기 매개변수 Θ_1



(b) Θ_1 을 개선하여 Θ_2 가 됨



(c) Θ_2 를 개선하여 최적의 Θ_3 을 찾음

그림 1-11 기계 학습에서 목적함수의 역할

1.4 간단한 기계 학습의 예

- 기계 학습이 할 일을 공식화하면,

$$\hat{\Theta} = \operatorname{argmin}_{\Theta} J(\Theta) \quad (1.9)$$

- 기계 학습은 작은 개선을 반복하여 최적해를 찾아가는 수치적 방법으로 식 (1.9)를 품
- 알고리즘 형식으로 쓰면,

알고리즘 1-1 기계 학습 알고리즘

입력: 훈련집합 \mathbb{X} 와 \mathbb{Y}

출력: 최적의 매개변수 $\hat{\Theta}$

- 1 난수를 생성하여 초기 해 Θ_1 을 설정한다.
- 2 $t=1$
- 3 $\text{while } (J(\Theta_t) \text{가 } 0.0 \text{에 충분히 가깝지 않음}) \quad // \text{수렴 여부 검사}$
- 4 $J(\Theta_t)$ 가 작아지는 방향 $\Delta\Theta_t$ 를 구한다. // $\Delta\Theta_t$ 는 주로 미분을 사용하여 구함
- 5 $\Theta_{t+1} = \Theta_t + \Delta\Theta_t$
- 6 $t=t+1$
- 7 $\hat{\Theta} = \Theta_t$

1.4 간단한 기계 학습의 예

■ 좀더 현실적인 상황

- 지금까지는 데이터가 선형을 이루는 아주 단순한 상황을 고려함
- 실제 세계는 선형이 아니며 잡음이 섞임 → **비선형** 모델이 필요

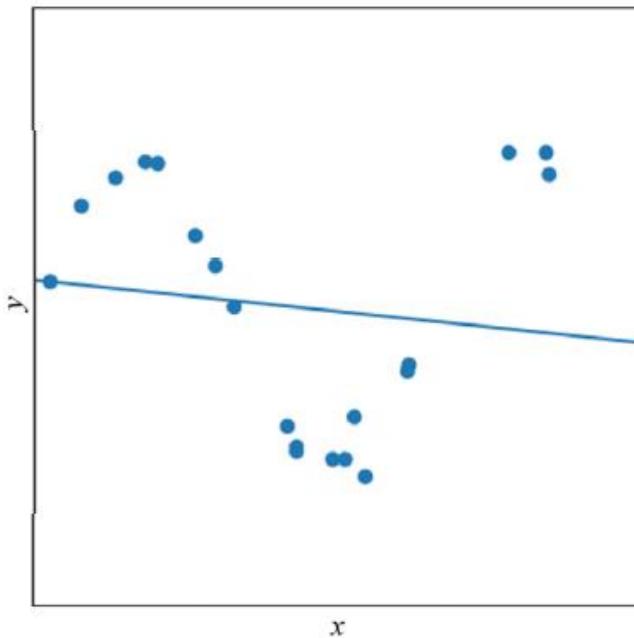


그림 1-12 선형 모델의 한계

1.5 모델 선택

- 1.5.1 과소적합과 과잉적합
- 1.5.2 바이어스와 분산
- 1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘
- 1.5.4 모델 선택의 한계와 현실적인 해결책

1.5.1 과소적합과 과잉적합

■ [그림 1.13]의 1차 모델은 과소적합 (Underfitting)

- 모델의 '용량이 작아' 오차가 클 수밖에 없는 현상

■ 비선형 모델을 사용하는 대안

- [그림 1-13]의 2차, 3차, 4차, 12차는 다항식 곡선을 선택한 예
- 1차(선형)에 비해 오차가 크게 감소함

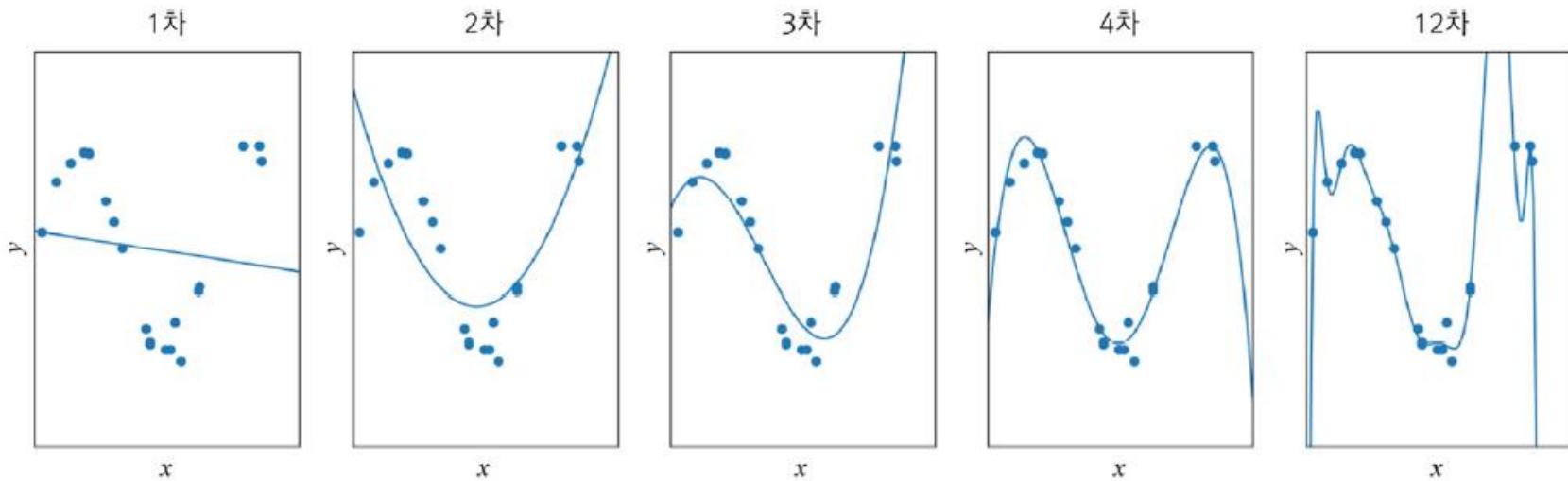


그림 1-13 과소적합과 과잉적합 현상

1.5.1 과소적합과 과잉적합

■ 과잉적합 (Overfitting)

- 12차 다항식 곡선을 채택한다면 훈련집합에 대해 거의 완벽하게 근사화함
- 하지만 '새로운' 데이터를 예측한다면 큰 문제 발생
 - x_0 에서 빨간 막대 근방을 예측해야 하지만 빨간 점을 예측
- 이유는 '용량이 크기' 때문. 학습 과정에서 잡음까지 수용 → 과잉적합 현상
- 적절한 용량의 모델을 선택하는 모델 선택 작업이 필요함

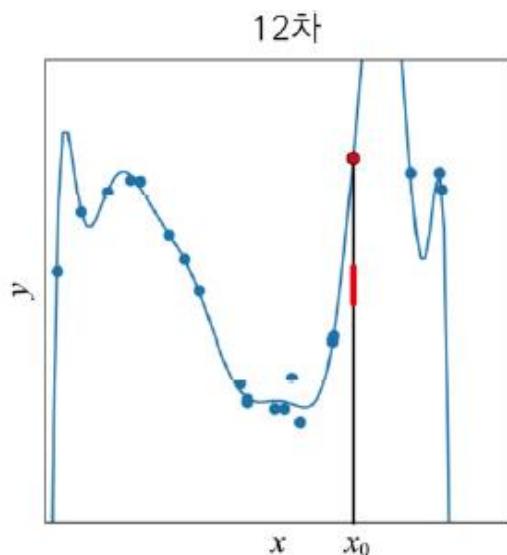


그림 1-14 과잉적합되었을 때 부정확한 예측 현상

1.5.2 바이어스와 분산

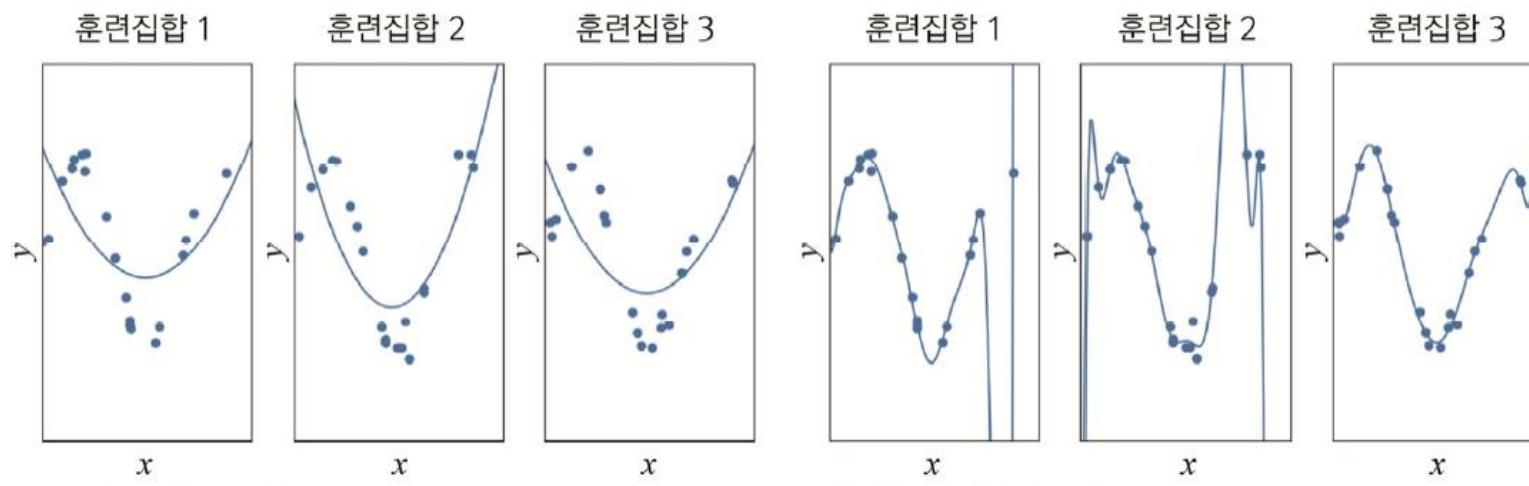
■ 1차~12차 다향식 모델의 비교 관찰

- 1~2차는 훈련집합과 테스트집합 모두 낮은 성능
- 12차는 훈련집합에 높은 성능을 보이나 테스트집합에서는 낮은 성능 → 낮은 일반화 능력
- 3~4차는 훈련집합에 대해 12차보다 낮겠지만 테스트집합에는 높은 성능 → 높은 일반화 능력

1.5.2 바이어스와 분산

■ 훈련집합을 여러 번 수집하여 1차~12차에 적용하는 실험

- 2차는 매번 큰 오차 → 바이어스가 큼. 하지만 비슷한 모델을 얻음 → 낮은 분산
- 12차는 매번 작은 오차 → 바이어스가 작음. 하지만 크게 다른 모델을 얻음 → 높은 분산
- 일반적으로 용량이 단순한 모델은 바이어스는 크고 분산은 작음. 복잡한 모델은 바이어스는 작고 분산은 큼
- 바이어스와 분산은 트레이드오프 관계



(a) 2차 모델(바이어스는 큼, 분산은 작음)

(b) 12차 모델(바이어스는 작고, 분산은 큼)

그림 1-15 모델의 바이어스와 분산 특성

1.5.2 바이어스와 분산

■ 기계 학습의 목표

- 낮은 바이어스와 낮은 분산을 가진 예측기 제작이 목표. 즉 왼쪽 아래 상황

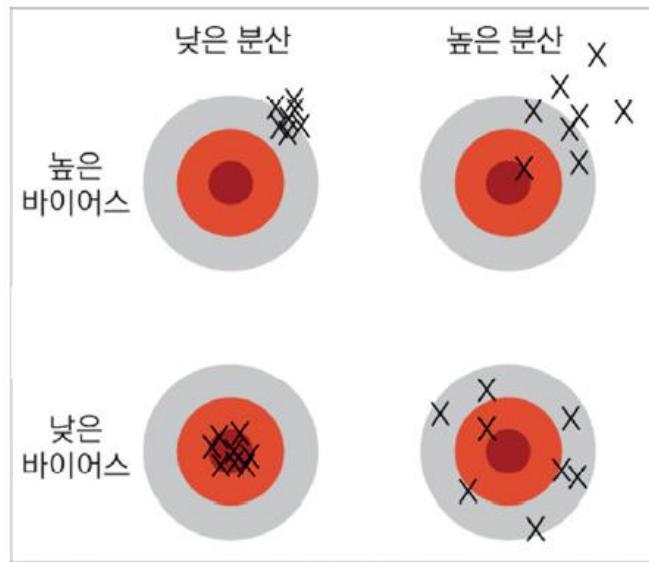


그림 1-16 바이어스와 분산

- 하지만 바이어스와 분산은 트레이드오프 관계
- 따라서 바이어스 희생을 최소로 유지하며 분산을 최대로 낮추는 전략 필요

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

■ 검증집합을 이용한 모델 선택

- 훈련집합과 테스트집합과 다른 별도의 **검증집합**을 가진 상황

알고리즘 1-2 검증집합을 이용한 모델 선택

입력 : 모델집합 Ω , 훈련집합, 검증집합, 테스트집합

출력 : 최적 모델과 성능

- 1 for (Ω 에 있는 각각의 모델)
- 2 모델을 훈련집합으로 학습시킨다.
- 3 검증집합으로 학습된 모델의 성능을 측정한다. // 검증 성능 측정
- 4 가장 높은 성능을 보인 모델을 선택한다.
- 5 테스트집합으로 선택된 모델의 성능을 측정한다.

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

■ 교차검증 cross validation

- 비용 문제로 별도의 검증집합이 없는 상황에 유용한 모델 선택 기법
- 훈련집합을 등분하여, 학습과 평가 과정을 여러 번 반복한 후 평균 사용

알고리즘 1-3 교차검증에 의한 모델 선택

입력 : 모델집합 Ω , 훈련집합, 테스트집합, 그룹 개수 k

출력 : 최적 모델과 성능

- 1 훈련집합을 k 개의 그룹으로 등분한다.
- 2 for (Ω 에 있는 각각의 모델)
 3 for ($i=1$ to k)
 4 i 번째 그룹을 제외한 $k-1$ 개 그룹으로 모델을 학습시킨다.
 5 학습된 모델의 성능을 i 번째 그룹으로 측정한다.
 6 k 개 성능을 평균하여 해당 모델의 성능으로 취한다.
 7 가장 높은 성능을 보인 모델을 선택한다.
 8 테스트집합으로 선택된 모델의 성능을 측정한다.

1.5.3 검증집합과 교차검증을 이용한 모델 선택 알고리즘

■ 부트스트랩 boot strap

- 난수를 이용한 샘플링 반복

알고리즘 1-4 부트스트랩을 이용한 모델 선택

입력 : 모델집합 Ω , 훈련집합, 테스트집합, 샘플링 비율 $p(0 < p \leq 1)$, 반복횟수 T

출력 : 최적 모델과 성능

- 1 for (Ω 에 있는 각각의 모델)
- 2 for ($i=1$ to T)
3 훈련집합 \mathbb{X} 에서 pn 개 샘플을 뽑아 새로운 훈련집합 \mathbb{X}' 를 구성한다.
4 \mathbb{X}' 로 모델을 학습시킨다.
5 $\mathbb{X} - \mathbb{X}'$ 를 이용하여 학습된 모델의 성능을 측정한다.
6 T 개 성능을 평균하여 해당 모델의 성능으로 취한다.
7 가장 높은 성능을 보인 모델을 선택한다.
8 테스트집합으로 선택된 모델의 성능을 측정한다.

1.5.4 모델 선택의 한계와 현실적인 해결책

- [알고리즘 1-2, 1-3, 1-4]에서 모델 집합 Ω
 - [그림 1-13]에서는 서로 다른 차수의 다항식이 Ω 인 셈
 - 현실에서는 아주 다양
 - 신경망, 강화 학습, 확률 그래피컬 모델, SVM, 트리 분류기 등이 선택 대상
 - 신경망을 채택하더라도 MLP, Deep MLP, CNN 등 아주 많음
- 현실적 해결책
 - 경험적으로 큰 틀(model 혹은 hypothesis) 선택
 - 모델 선택 알고리즘으로 세부 모델 선택하는 전략 사용
 - 예) CNN을 사용하기로 정한 후, 은닉층 개수, 활성함수, 모멘텀 계수 등을 정하는데 모델 선택 알고리즘을 적용함

1.5.4 모델 선택의 한계와 현실적인 해결책

■ 이런 경험적인 접근방법에 대한 『Deep Learning』 책의 비유

“To some extent, we are always trying to fit a square peg(the data generating process) into a round hole(our model family). 어느 정도 우리가 하는 일은 항상 둥근 홈(우리가 선택한 모델)에 네모 막대기(데이터 생성 과정)를 끼워 넣는 것이라고 말할 수 있다[Goodfellow2016(222쪽)].”

■ 현대 기계 학습의 전략

- 용량이 충분히 큰 모델을 선택 한 후, 선택한 모델이 정상을 벗어나지 않도록 여러 가지 규제|regularization 기법을 적용함
 - Overfitting 방지를 위한 기법
- 예) [그림 1-13]의 경우 12차 다항식을 선택한 후 적절히 규제를 적용

1.6 규제 (Regularization)

- 1.6.1 데이터 확대
- 1.6.2 가중치 감쇠

1.6.1 데이터 확대

- 데이터를 더 많이 수집하면 일반화 능력이 향상됨

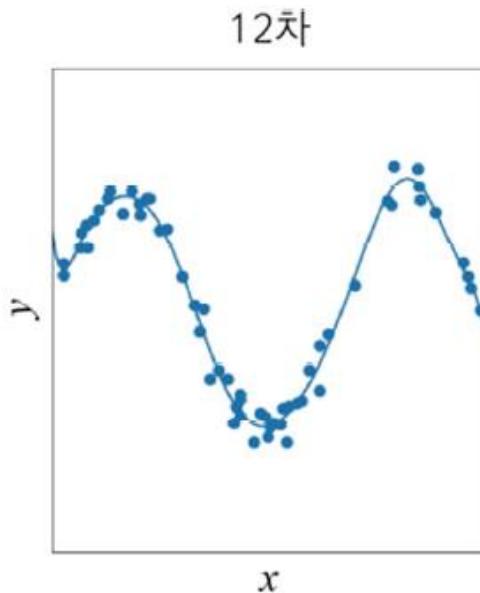
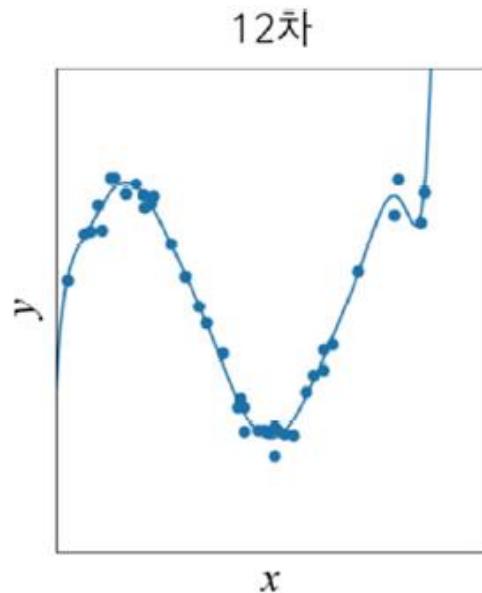
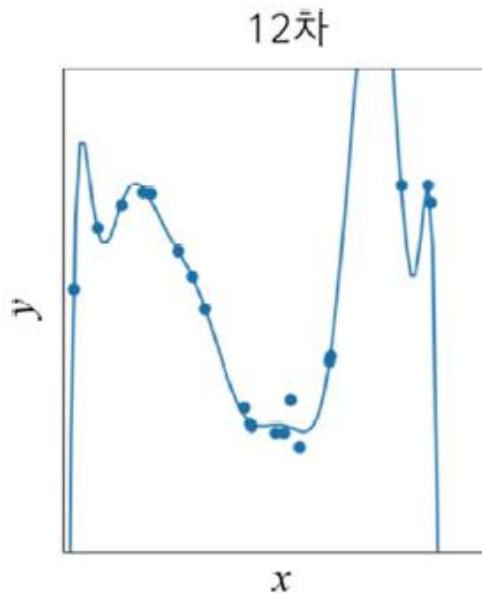


그림 1-17 데이터를 확대하여 일반화 능력을 향상함

1.6.1 데이터 확대

■ 데이터 수집은 많은 비용이 들

- 그라운드 트루스를 사람이 일일이 레이블링해야 함

■ 인위적으로 데이터 확대

- 훈련집합에 있는 샘플을 변형함
- 약간 회전 또는 와핑 (부류 소속이 변하지 않게 주의)

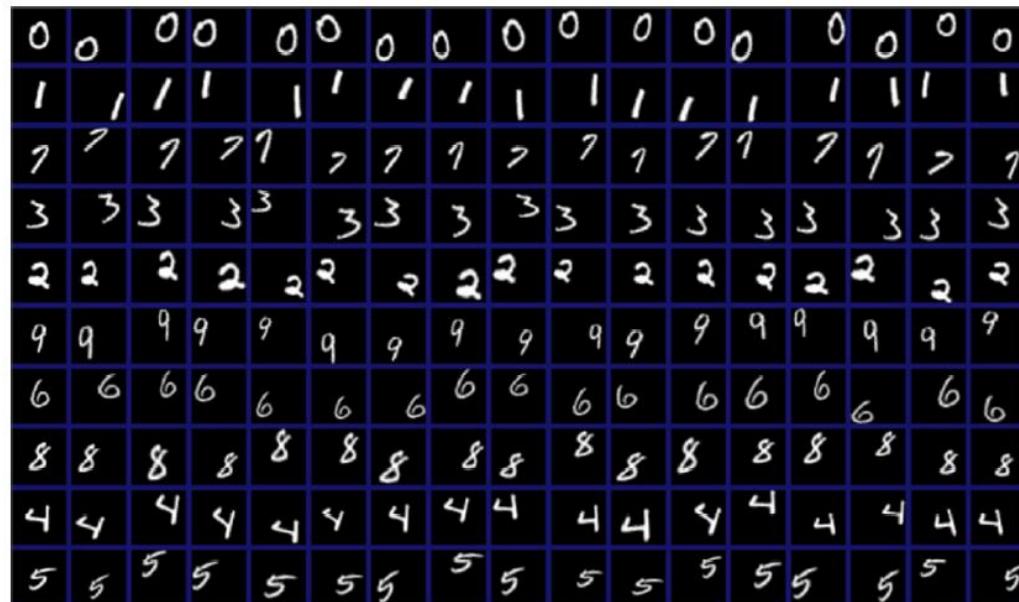


그림 5-24 필기 숫자 데이터의 다양한 변형⁸

1.6.2 가중치 감쇠

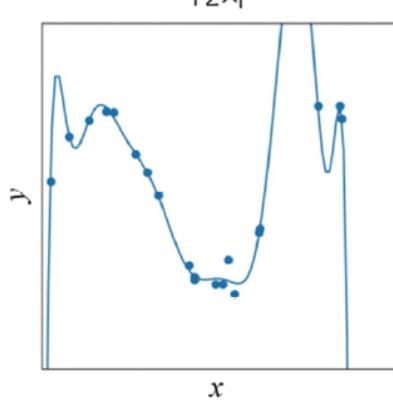
■ 가중치를 작게 조절하는 기법

- [그림 1-18(a)]의 12차 곡선은 가중치가 매우 큼

$$y = 1005.7x^{12} - 27774.4x^{11} + \cdots - 22852612.5x^1 - 12.8$$

- 가중치 감쇠는 개선된 목적함수를 이용하여 가중치를 작게 조절하는 규제 기법
 - 식 (1.11)의 두 번째 항은 규제 항으로서 가중치 크기를 작게 유지해줌

$$J(\Theta) = \frac{1}{n} \sum_{i=1}^n (f_\Theta(\mathbf{x}_i) - y_i)^2 + \lambda \|\Theta\|_2^2 \quad (1.11)$$



(a) 가중치 감쇠 적용 안 함[식 (1.8)의 목적함수]

(b) 가중치 감쇠 적용함[식 (1.11)의 목적함수]

그림 1-18 가중치 감쇠에 의한 규제 효과

1.7 기계 학습의 유형

■ 지도 학습

- 특징 벡터 \mathbb{x} 와 목푯값 \mathbb{y} 가 모두 주어진 상황
- 회귀와 분류 문제로 구분

■ 비지도 학습

- 특징 벡터 \mathbb{x} 는 주어지는데 목푯값 \mathbb{y} 가 주어지지 않는 상황
- 군집화(clustering)
- 밀도 추정, 특징 공간 변환 과업

1.7 기계 학습의 유형

■ 강화 학습

- 목푯값이 주어지는데, 지도 학습과 다른 형태임
- 예) 바둑
 - 수를 두는 행위가 샘플인데, 게임이 끝나면 목푯값 하나가 부여됨
 - 이기면 1, 패하면 -1을 부여
 - 게임을 구성한 샘플들 각각에 목푯값을 나누어 주어야 함

■ 준지도 학습

- 일부는 \mathbb{X} 와 \mathbb{Y} 를 모두 가지지만, 나머지는 \mathbb{X} 만 가진 상황
- 인터넷 덕분으로 \mathbb{X} 의 수집은 쉽지만, \mathbb{Y} 는 수작업이 필요하여 최근 중요성 부각

1.8 기계 학습의 과거와 현재, 미래

- 1.8.1 인공지능과 기계 학습의 간략한 역사
- 1.8.2 기술 추세
- 1.8.3 사회적 전망

1.8.1 인공지능과 기계 학습의 간략한 역사

■ 베비지의 제자인 에이더 여사의 통찰력 (19세기 중반)

- "... 해석엔진은 숫자 이외의 것도 처리할 수 있을 것이다. ... 예를 들어 화음과 음조를 해석 엔진의 표기에 맞출 수 있다면, 해석엔진은 꽤 복잡한 곡을 작곡할 수도 있다." [Ada1843]
- 200여 년이 지난 지금,
 - 흘려 쓴 필기 숫자를 0.23% 오류로 인식
 - 알파고는 이세돌을 이김
 - 자연영상에 대해 다섯 단어 가량의 문장으로 묘사함

1.8.1 인공지능과 기계 학습의 간략한 역사

- 1843 에이더 “… 해석엔진은 꽤 복잡한 곡을 작곡할 수도 있다.”라는 논문 발표[Ada1843]
- 1950 인공지능 여부를 판별하는 튜링 테스트[Turing1950]
- 1956 최초의 인공지능 학술대회인 다트머스 콘퍼런스 개최. ‘인공지능’ 용어 탄생[McCarthy1955]
- 1958 로젠블렛이 퍼셉트론 제안[Rosenblatt1958]
인공지능 언어 Lisp 탄생
- 1959 사무엘이 기계 학습을 이용한 체커 게임 프로그램 개발[Samuel1959]
- 1969 민스키가 퍼셉트론의 과대포장 지적. 신경망 내리막길 시작[Minsky1969]
제1회 IJCAI International Joint Conference on Artificial Intelligence 개최
- 1972 인공지능 언어 Prolog 탄생
- 1973 Lighthill 보고서로 인해 인공지능 내리막길, 인공지능 겨울AI winter 시작
- 1974 웨어보스가 오류 역전파 알고리즘을 기계 학습에 도입[Werbos1974]
- 1975경 의료진단 전문가 시스템 Mycin – 인공지능에 대한 관심 부활
- 1979 「IEEE Transactions on Pattern Analysis and Machine Intelligence」 저널 발간
- 1980 제1회 ICML International Conference on Machine Learning 개최
후쿠시마가 NeoCognitron 제안[Fukushima1980]
- 1986 「Machine Learning」 저널 발간
『Parallel Distributed Processing』 출간
다층 퍼셉트론으로 신경망 부활

1.8.1 인공지능과 기계 학습의 간략한 역사

- 1987 Lisp 머신의 시장 붕괴로 제2의 인공지능 겨울
UCI 리포지토리 서비스 시작
NIPS Neural Information Processing Systems 콘퍼런스 시작
- 1989 「Neural Computation」 저널 발간
- 1993 R 언어 탄생
- 1997 IBM 딥블루가 세계 체스 챔피언인 카스파로프 이김
LSTM Long short-term memory 개발됨
- 1998경 SVM이 MNIST 인식 성능에서 신경망 추월
- 1998 르쿤이 CNN의 실용적인 학습 알고리즘 제안[LeCun1998]
『Neural Networks: Tricks of the Trade』 출간
- 1999 NVIDIA 사에서 GPU 공개
- 2000 「Journal of Machine Learning Research」 저널 발간
OpenCV 최초 공개
- 2004 제1회 그랜드 챌린지(자율 주행)
- 2006 층별학습 탄생[Hinton2006a]
- 2007경 딥러닝이 MNIST 인식 성능에서 SVM 추월
- 2007 GPU 프로그래밍 라이브러리인 CUDA 공개

1.8.1 인공지능과 기계 학습의 간략한 역사

	어번 챌린지(도심 자율 주행)
	Scikit-learn 라이브러리 최초 공개
2009	Theano 서비스 시작
2010	ImageNet 탄생 제1회 ILSVRC 대회
2011	IBM 왓슨이 제퍼디 우승자 꺾음
2012	MNIST에 대해 0.23% 오류율 달성 AlexNet 발표 (3회 ILSVRC 우승)
2013	제1회 ICLR International Conference on Learning Representations 개최
2014	Caffe 서비스 시작
2015	TensorFlow 서비스 시작 OpenAI 창립
2016	알파고와 이세돌의 바둑 대회에서 알파고 승리[Silver2016] 『Deep Learning』 출간
2017	알파고 제로[Silver2017]

1.8.2 기술 추세

- 리뷰 논문
 - [LeCun2015, Jordan2015, Jones2014]

- 기계 학습은 인공지능 실현에 핵심 기술

- 기계 학습 알고리즘과 응용의 다양화

- 서로 다른 알고리즘과 응용의 융합

- 딥러닝이 기계 학습의 주류

- 표현 학습이 중요해짐

1.8.3 사회적 전망

■ 미래의 직업 변화

- 시의적절하고 심사숙고 해야 할 객관적 담론
- 프레이는 702개 직업의 사라질 위기를 확률로 계산 [Frey2017]
- 텔레마케터 99% 오락 치료사 0.28%

■ 기계가 사람을 지배할지 모른다는 두려움

- 알파고 이후 매스컴을 통해 여과 없이 전파. 쓸데없는 과장에 불과
- 냇가에 다리 놓는 일과 목포-제주에 대교를 놓는 일은 규모만 다를 뿐 본질적으로 같은 일
- 오목 프로그램이나 바둑 프로그램은 규모만 다를 뿐 본질적으로 같은 일. 오목은 간단한 규칙으로 구현 가능하나 바둑은 미분을 사용한 복잡한 기계 학습 알고리즘 사용
- 현재 기계 학습은 온통 수학과 컴퓨터 알고리즘일 뿐

머신러닝 소개 및 실습



왜 머신러닝인가?

- 초창기 지능형 애플리케이션
 - 조건문 사용
 - 예) 스팸 메일 필터링
 - 결정 규칙을 사람이 직접 모델링
- 결정 규칙을 직접 만들 때의 단점
 - 결정에 필요한 로직은 한 분야나 작업에 국한됨
 - 작업이 조금만 변경 되더라도 전체 시스템을 다시 개발해야 함
 - 숫자 8인식 → 2인식?
 - 규칙을 설계하려면 그 분야 전문가들이 내리는 결정 방식에 대해 잘 알아야 함
 - 얼굴 인식 문제



왜 머신러닝인가?

- 결정 규칙을 직접 만들 때의 단점
 - 얼굴 인식 문제
 - 컴퓨터의 인식 방식 vs. 사람의 인식하는 방식
 - 얼굴의 다양성. 해결책은?



왜 머신러닝인가?

- 머신러닝으로 풀 수 있는 문제
 - 지도 학습
 - 주어진 입력에 대한 출력 예측
 - 학습에 필요한 입, 출력 데이터는
 - 지도 학습의 예
 - 편지 봉투에 손으로 쓴 우편번호 숫자 판별
 - 입력, 출력?
 - 학습에 필요한 데이터?
 - 의료 영상 이미지에 기반한 종양 판단
 - 입력, 출력?
 - 학습에 필요한 데이터?
 - 의심되는 신용 카드 거래 감지
 - 입력, 출력?
 - 학습에 필요한 데이터?



왜 머신러닝인가?

- 머신러닝으로 풀 수 있는 문제
 - 비지도 학습
 - 주어진 입력에 대한 출력이 제공되지 않음
 - 학습을 이해하거나 평가하는 일이 쉽지 않음
 - 비지도 학습의 예
 - 블로그 글의 주제 구분
 - 텍스트 데이터 요약 및 핵심주제 찾기
 - 고객들을 취향이 비슷한 그룹으로 묶기
 - 어떤 고객들의 취향이 비슷한지 파악
 - 비슷한 취향의 고객 그룹화
 - 비 정상적인 웹사이트 접근 탐지
 - 웹 트래픽을 이용한 탐지



왜 머신러닝인가?

- 컴퓨터에 머신러닝 적용하려면
 - 컴퓨터가 인식할 수 있는 데이터 제공
 - 샘플 (데이터 포인트), 특성
 - 좋은 특성 추출 필요
 - 예) 성씨로 성별 구분?



왜 머신러닝인가?

- 문제와 데이터 이해하기
 - 머신러닝 프로세스에서 가장 중요한 과정은 사용할 데이터를 이해하고 그 데이터가 해결해야 할 문제와 어떤 관련이 있는지를 이해하는 것
 - 어떤 질문에 대한 답을 원하는가? 데이터가 답을 줄 수 있나?
 - 내 질문을 잘 기술할 수 있는 머신러닝 방법은?
 - 문제를 풀기 위한 충분한 데이터를 모았는가?
 - 내가 추출한 데이터의 특성으로 좋은 결과를 만들 수 있나?
 - 머신러닝 성능 측정 방법?
 - 다른 연구나 제품과의 협력은?



왜 파이썬인가?

- 파이썬은 데이터 과학 분야를 위한 표준 프로그래밍 언어
 - 파이썬은 범용 프로그래밍 언어의 장점은 물론 매트랩MATLAB과 R 같은 특정 분야를 위한 스크립팅 언어의 편리함을 함께 갖춤
 - 다양한 도구: 데이터 적재, 시각화, 통계, 자연어 처리, 이미지 처리 등에 필요한 라이브러리 존재
 - 터미널이나 주피터 노트북(Jupyter Notebook) 같은 도구로 대화하듯 프로그래밍할 수 있음
 - 머신러닝과 데이터 분석은 데이터 주도 분석이라는 점에서 근본적으로 반복 작업, 따라서 반복 작업을 빠르게 처리하고 손쉽게 조작할 수 있는 도구가 필수
 - 범용 프로그래밍 언어로서 파이썬은 복잡한 그래픽 사용자 인터페이스 (GUI)나 웹 서비스도 만들 수 있으며 기존 시스템과 통합하기도 좋음



Scikit-Learn

- 오픈소스로 자유롭게 사용하거나 배포 가능
 - 잘 알려진 머신러닝 알고리즘들은 물론 알고리즘을 설명한 풍부한 문서도 제공
 - <http://scikit-learn.org/stable/documentation>
 - 사이킷런은 매우 인기가 높고 독보적인 파이썬 머신러닝 라이브러리임
 - 산업 현장이나 학계에도 널리 사용되고 많은 튜토리얼과 예제 코드를 온라인에서 쉽게 찾을 수 있음
 - 사이킷런은 다른 파이썬의 과학 패키지들과도 잘 연동됨



Scikit-Learn

■ 설치

- Scikit-learn은 두 개의 다른 파이썬 패키지인 넘파이(NumPy)와 사이파이(SciPy)를 사용
- 그래프를 그리려면 맷플롯립(matplotlib)을, 대화식으로 개발하려면 아이피아이썬(IPython)과 주피터 노트북도 설치해야 함
- 필요한 패키지들을 모아 놓은 파이썬 배포판을 설치하는 방법을 권장
 - **Anaconda: 대용량 데이터 처리, 예측 분석, 과학 계산용 파이썬 배포판**
 - Enthought Canopy: 과학 계산용 파이썬 배포판
 - Python(x,y): 윈도우 환경을 위한 과학 계산용 무료 파이썬 배포판



Scikit-Learn

■ 설치

- 주피터 노트북
 - 주피터 노트북은 프로그램 코드를 브라우저에서 실행해주는 대화식 환경을 제공
- NumPy
 - 파이썬으로 과학 계산을 하려면 꼭 필요한 패키지임. 다차원 배열을 위한 기능과 선형 대수 연산과 푸리에 변환 같은 고수준 수학 함수와 유사(pseudo) 난수 생성기를 포함
- SciPy
 - 과학 계산용 함수를 모아놓은 파이썬 패키지임. SciPy는 고성능 선형 대수, 함수 최적화, 신호 처리, 특수한 수학 함수와 통계 분포 등을 포함한 많은 기능을 제공



Scikit-Learn

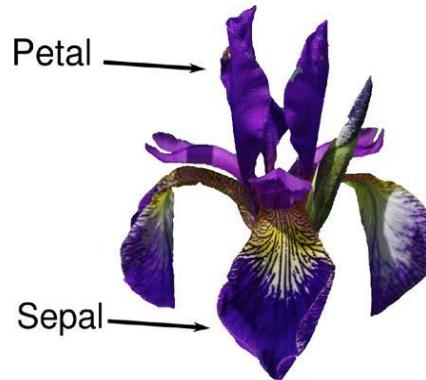
■ 설치

- matplotlib
 - 파이썬의 대표적인 과학 계산용 그래프 라이브러리임. 선 그래프, 히스토그램, 산점도 등을 지원하며 출판에 쓸 수 있을 만큼의 고품질 그래프를 그려줌
- pandas
 - 데이터 처리와 분석을 위한 파이썬 라이브러리임
- mglearn
 - Helper functions for the book ‘Introduction to machine learning with Python’
 - pip install mglearn

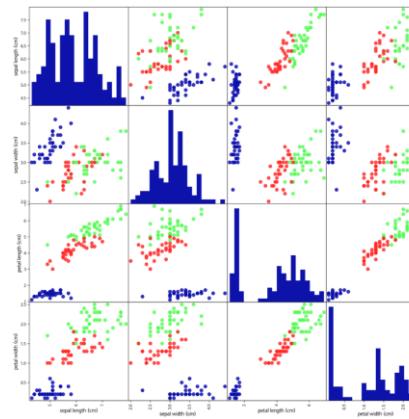


실습 – 붓꽃의 품종 분류

- 어떤 품종인지 구분해 놓은 측정 데이터를 이용, 새로 채집한 붓꽃의 품종을 예측하는 머신러닝 모델 구현
 - 데이터 적재
 - 성과 측정: 훈련 데이터와 테스트 데이터
 - 가장 먼저 할일: 데이터 살펴보기
 - 첫 번째 머신러닝 모델: K-최근접 이웃 알고리즘
 - 예측 및 모델 평가하기



▲그림 1-2 붓꽃의 부위



▲그림 1-3 클래스 레이블을 색으로 구분한 Iris 데이터셋의 산점도 행렬



선형모델



분류와 회귀

- 분류
 - 미리 정의된, 가능성 있는 여러 클래스 레이블 중 하나를 예측하는 것
 - 딱 두 개의 클래스로 분류하는 이진 분류(binary classification)와 셋 이상의 클래스로 분류하는 다중 분류(multiclass)로 나뉨
 - 예) 스팸 메일 분류?
 - 예) 봇꽃 분류?
- 회귀
 - 연속적인 숫자, 또는 프로그래밍 용어로 말하면 부동소수점수(수학 용어로는 실수)를 예측하는 것
 - 예) 어떤 사람의 교육 수준, 나이, 주거지를 바탕으로 연간 소득을 예측
 - 예) 옥수수 농장의 수확량 예측



일반화, 과대적합, 과소적합

- 예) 요트 구매 고객 예측

- 요트를 구매한 고객과 구매하지 않은 고객 데이터를 이용하여 예측
- 구매 할 확률이 높은 고객에게 홍보 메일을 보내는 것이 목표

Table 2-1. Example data about customers

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no



일반화, 과대적합, 과소적합

- 예) 요트 구매 고객 예측

- 가정1: 45세 이상 and (자녀가 셋 미만 or 이혼 하지 않은 고객)이 요트를 구매한다.

- 정확도: 100%

- 우리의 목적에 잘 맞을까?

- 평가를 하려면?

- 가정2: 50세 이상이 요트를 구매한다

- 일반화 관점에서 가정1 vs. 가정2 중 더 좋은 모델은?

- 가정 3: 이혼한 사람이 요트를 구매한다

Table 2-1. Example data about customers

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no



일반화, 과대적합, 과소적합

- 일반화 성능이 최대가 되는 모델이 최적임
 - 일반화
 - 모델이 처음 보는 데이터에 대해 정확하게 예측할 수 있으면 이를 훈련 세트에서 테스트 세트로 “일반화(generalization)”되었다고 함
 - 과대적합
 - 가진 정보를 모두 사용해서 너무 복잡한 모델을 만드는 것을 “과대적합 (overfitting)”이라 함
 - 과소적합
 - 모델이 너무 간단하여 데이터의 면면과 다양성을 잡아내지 못하고 훈련 세트에도 잘 맞지 않는 경우처럼, 너무 간단한 모델이 선택되는 것을 “과소적합(underfitting)”이라 함



일반화, 과대적합, 과소적합

- 모델 복잡도에 따른 훈련과 테스트 정확도

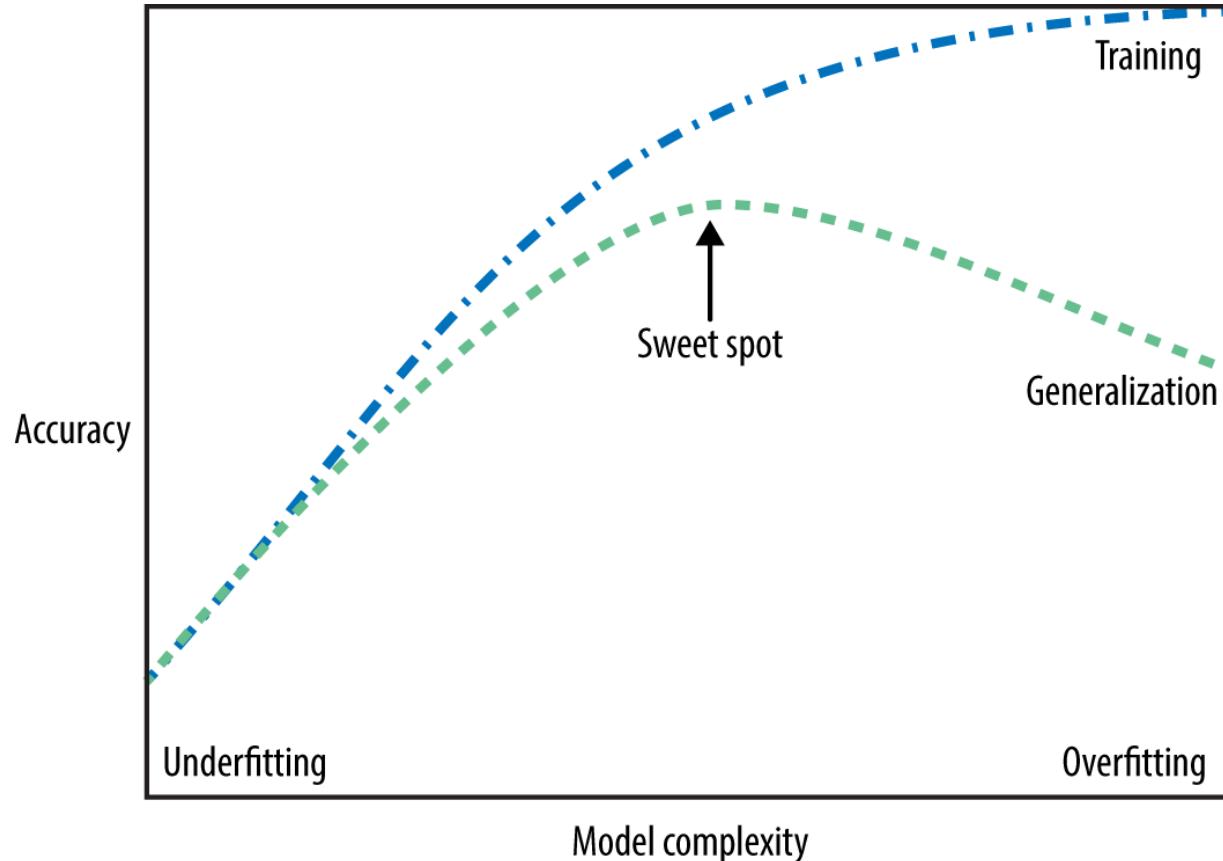


Figure 2-1. Trade-off of model complexity against training and test accuracy

일반화, 과대적합, 과소적합

- 모델 복잡도와 데이터 셋 크기의 관계
 - 모델의 복잡도는 훈련 데이터 셋에 담긴 입력 데이터의 다양성과 관련이 깊음
 - 데이터 포인트(샘플)이 많을수록 과대 적합 없이 복잡한 모델 생성
 - 요트 판매에서 고객 데이터 10,000개를 모았을 때 가정 1을 만족했다면 12개를 사용 할 때 보다 좋은 규칙이라고 생각 할 수 있음



지도 학습 알고리즘

- 머신러닝 알고리즘의 작동 방식 학습
 - 데이터로부터 어떻게 학습하고 예측하는가?
 - 모델의 복잡도가 어떤 역할을 하는가?
 - 알고리즘이 모델을 어떻게 만드는가?
 - 모델들의 장단점을 평가하고 어떤 데이터가 잘 들어맞을지 살펴보기
 - 매개변수와 옵션의 의미 학습
- 알고리즘
 - K-최근접 이웃
 - 선형모델
 - 나이브 베이즈 분류기
 - 결정 트리 및 앙상블
 - 커널 서포트 벡터 머신
 - 신경망 (딥러닝)

지도 학습 알고리즘

- 데이터셋 생성

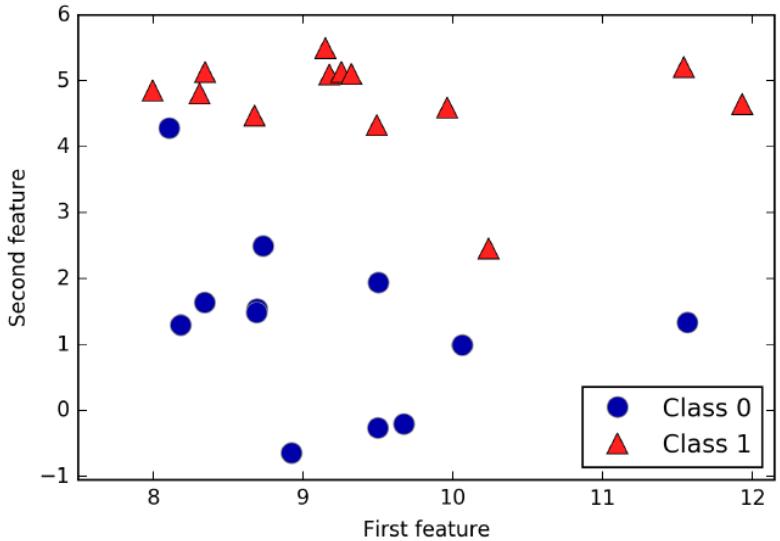


Figure 2-2. Scatter plot of the forge dataset

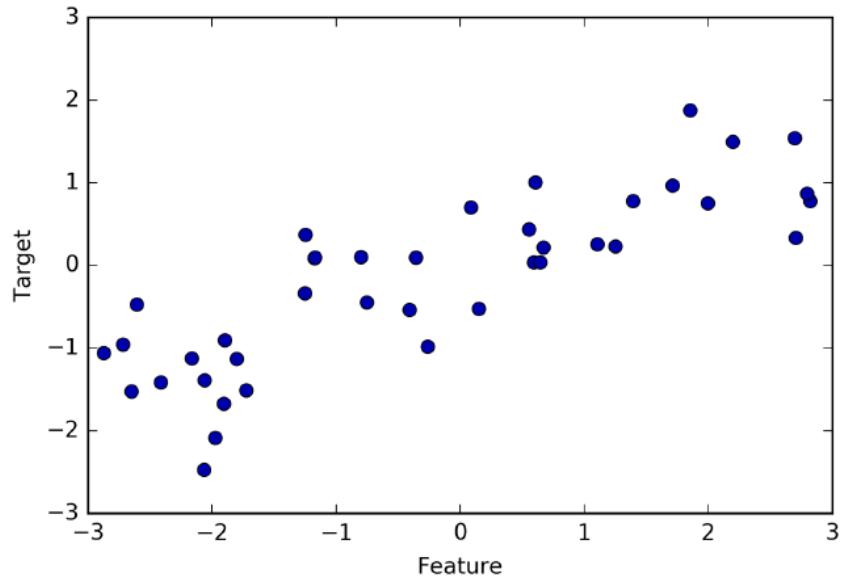


Figure 2-3. Plot of the wave dataset, with the x-axis showing the feature and the y-axis showing the regression target

지도 학습 알고리즘

- K-NN 알고리즘
 - 중요 매개변수
 - Metric
 - # of neighbor
 - 이해하기 매우 쉬운 알고리즘
 - 조정을 많이 하지 않아도 성능이 괜찮음
 - 복잡한 알고리즘을 사용하기 전에 시도해보기 좋음
 - 훈련 세트가 매우 크면 예측이 느려짐
 - 전처리 중요 (3장 참고)
 - 특성이 많은 경우 잘 작동하지 않음
 - 특성 값 대부분이 0인 경우 잘 작동하지 않음
 - 요약: 이해하기 쉬우나 예측이 느리고 많은 특성을 처리하는 능력 부족으로 현업에서는 잘 사용하지 않음



지도 학습 알고리즘

- 1차원 선형 모델
 - 학습데이터
 - 총 N개의 (나이, 키) 데이터
 - 입력변수 $x = \{x_0, x_1, x_2, \dots, x_{N-1}\}$
 - 목표변수 $t = \{t_0, t_1, t_2, \dots, t_{N-1}\}$
 - 데이터 랜덤 생성



지도 학습 알고리즘

■ 1차원 선형 모델

▪ Hypothesis (모델링)

- 분포하는 데이터를 보고 모델링
- 가장 간단한 모델인 직선 방정식 사용
 - $y(x) = w_0x + w_1$

▪ Loss function (비용함수)

- 학습 데이터 분포와 가장 잘 맞는 직선의 식을 찾는 것이 목적
- 직선 방정식을 이루는 적당한 파라미터 (w_0, w_1)을 찾아야 함
 - 학습 데이터와 직선의 방정식이 가장 잘 맞는다의 의미?
 - 맞고 안 맞음의 정도를 측정할 수 있는 측정값 필요
- 가장 많이 사용되는 평균제곱오차(MSE)
 - $J(w_0, w_1) = \frac{1}{N} \sum_{n=0}^{N-1} (y_n - t_n)^2$
 - $y_n = y(x_n) = w_0x_n + w_1$



지도 학습 알고리즘

- 1차원 선형 모델

- Optimization (최적화)

- Loss function을 최소화 하는 작업

- J 함수가 최소가 되는 w_0, w_1 값 계산

- Gradient Descent (경사하강법)

- $w(k+1) = w(k) - \alpha \nabla J$

- $\nabla J = ?$

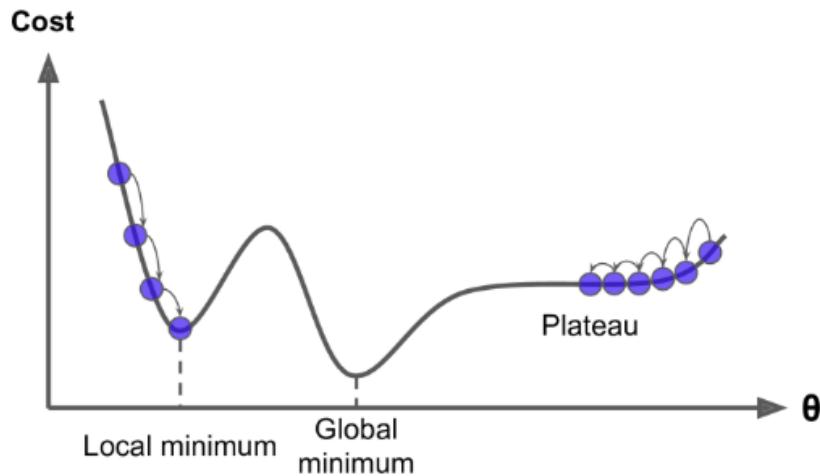
- $w(k+1) = ?$

```
def calcGrad(x, t, w0, w1):  
    y = w0 * x + w1  
    d_w0 = 2 * np.mean((y - t) * x)  
    d_w1 = 2 * np.mean(y - t)  
    return d_w0, d_w1  
  
temp = calcGrad(X, T, 10, 165)  
print(np.round(temp, 1))
```



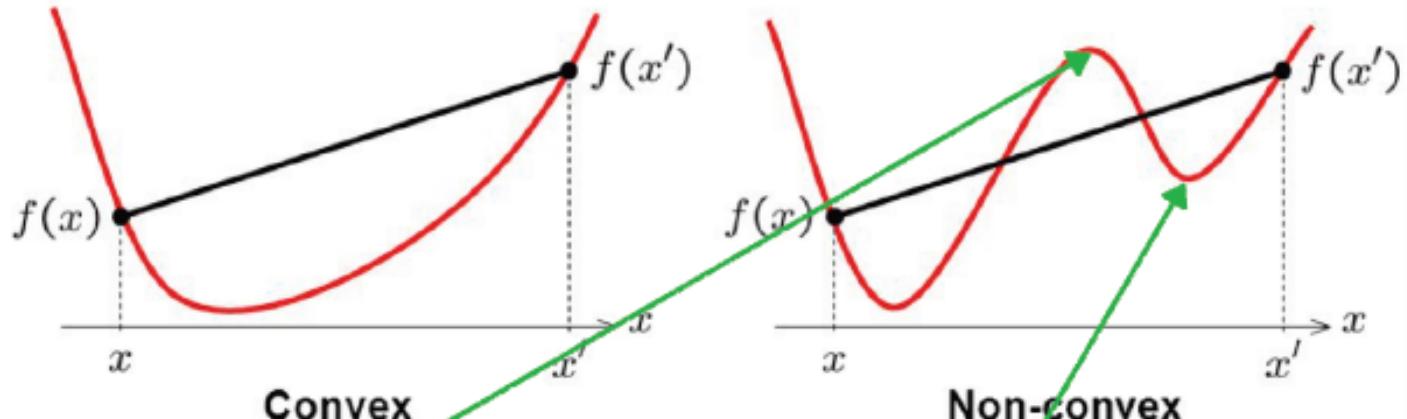
(참고) 경사하강법 (Gradient Descent)

- 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



(참고) 경사하강법

■ Convex Function



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



(참고) 경사하강법

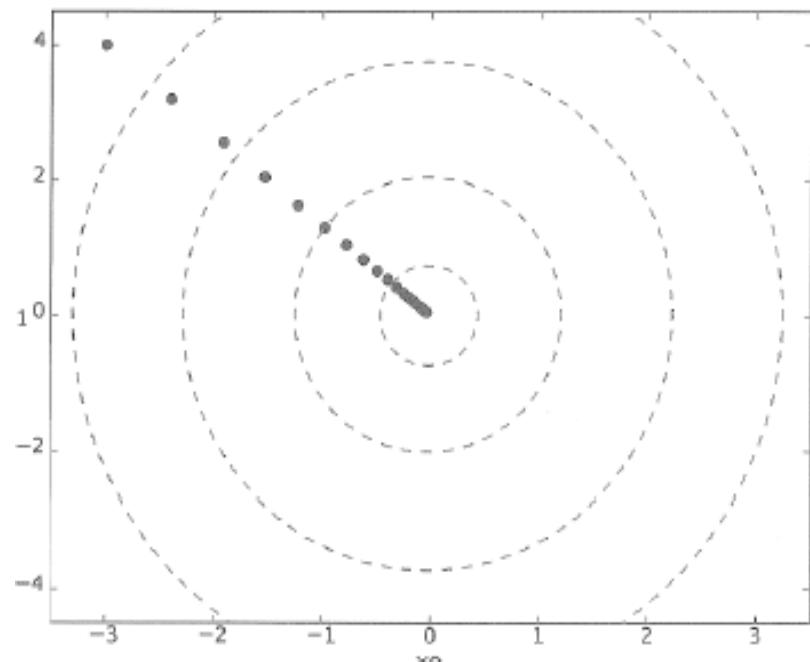
■ 개념

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

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

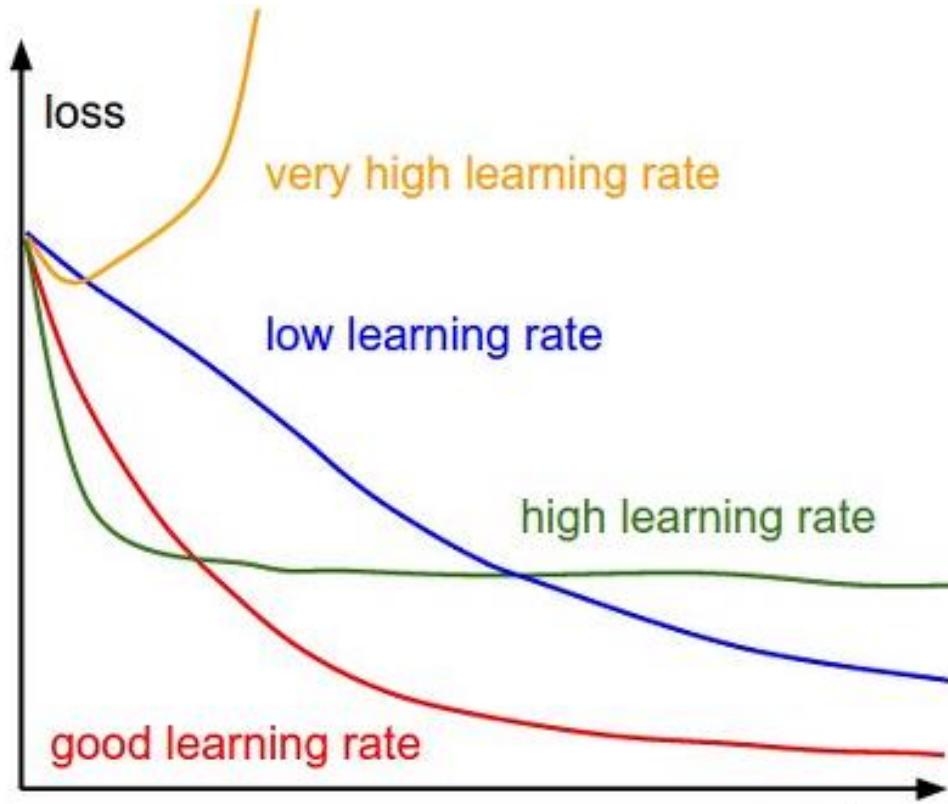
$$x_1 = x_1 - \eta \frac{\partial f}{\partial x_1}$$

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



(참고) 경사하강법

- 개념
 - 학습률(learning rate)



출처: <https://seamless.tistory.com/38>

epoch

지도 학습 알고리즘

■ 1차원 선형 모델

▪ Optimization (최적화)

- Loss function을 최소화 하는 작업

▪ grad_loss 함수 이용 경사하강법

```
def GradDsctOpt(x, t):
    w_init = [10.0, 165.0] # 초기 매개 변수
    alpha = 0.001 # 학습률
    i_max = 100000 # 반복의 최대 수
    eps = 0.1 # 반복을 종료 기울기의 절대 값의 한계
    w0 = w_init[0]
    w1 = w_init[1]
    for i in range(1, i_max):
        dmse = calcGrad(x, t, w0, w1)
        w0 = w0 - alpha * dmse[0]
        w1 = w1 - alpha * dmse[1]
        if max(np.absolute(dmse)) < eps: # 종료판정, np.absolute는 절대치
            break
    return w0, w1
```

```
# 구매법 호출
_w0, _w1 = GradDsctOpt(X, T)
# 결과보기
print('W=[{0:.6f}, {1:.6f}]'.format(_w0, _w1))
```



지도 학습 알고리즘

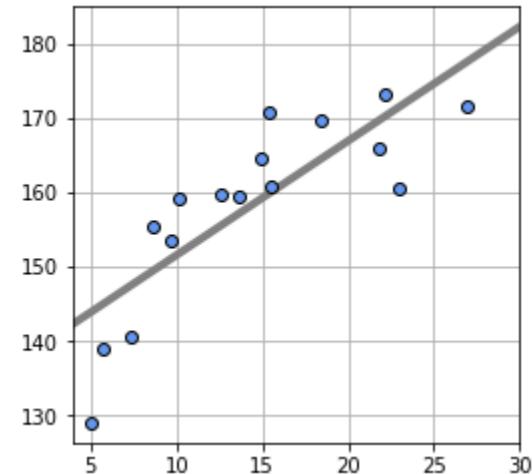
- 1차원 선형 모델
 - 최종 결과
 - 그래프 가시화

```
# 리스트 5-1-(10)
# 선 표시 -----
def show_line(w):
    xb = np.linspace(X_min, X_max, 100)
    y = w[0] * xb + w[1]
    plt.plot(xb, y, color=(.5, .5, .5), linewidth=4)

# 예제 -----
plt.figure(figsize=(4, 4))
W=np.array([W0, W1])
mse = mse_line(X, T, W)
print("w0={0:.3f}, w1={1:.3f}".format(W0, W1))
# mse = mse_line(X, T, W)
print("SD={0:.3f} cm".format(np.sqrt(mse)))
show_line(W)
plt.plot(X, T, marker='o', linestyle='None',
         color='cornflowerblue', markeredgecolor='black')
plt.xlim(X_min, X_max)
plt.grid(True)
plt.show()
```

w0=1.540, w1=136.176
SD=7.002 cm

```
def mse_line(x, t, w):
    y = w[0] * x + w[1]
    mse = np.mean((y - t)**2)
    return mse
```



지도 학습 알고리즘

■ 선형 모델 실습

- 1차원 직선 방정식을 정하고 이 식을 이용하여 데이터를 50개 생성
 - $w_0 = 3, w_1 = 5$
 - x의 범위는 -20~50까지 50개 생성
 - 랜덤 값을 이용하여 생성 (scale = 50)
- loss function, gradient function, gradient_descent function 구현
- 최적화 및 w_0, w_1 값 구하기
 - 초기값 $w_0 = ?, w_1 = ?$
 - learning rate = ?
 - iteration count = ?



지도 학습 알고리즘

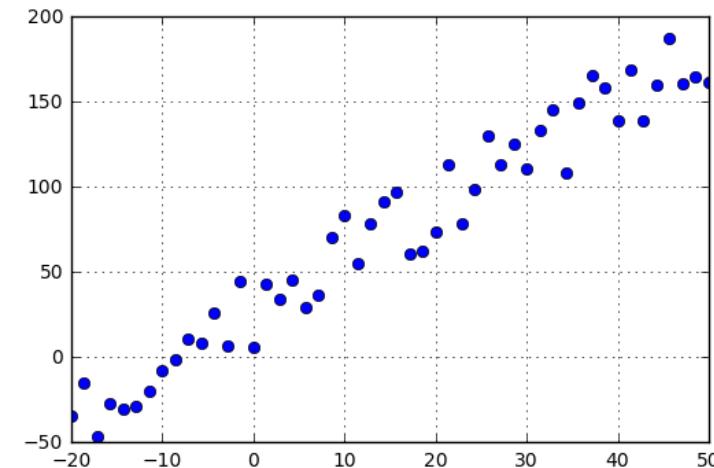
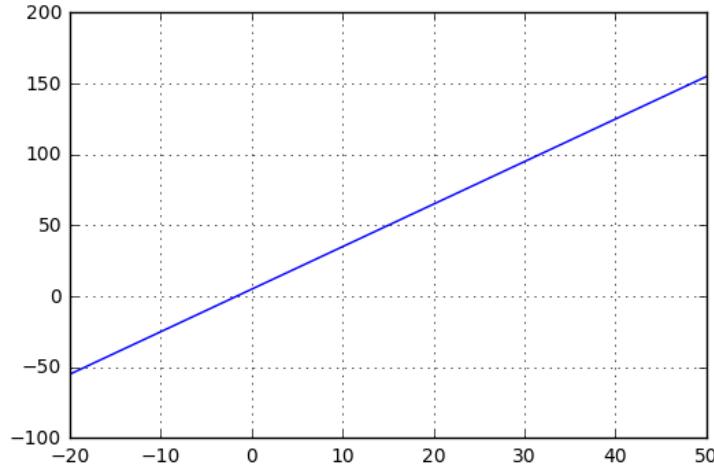
- 선형 모델 실습
 - 데이터 생성

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
def targetfunc(x):
    return 3*x + 5
```

```
x = np.linspace(-20, 50, 50)
fx = targetfunc(x)
plt.plot(x, fx)
plt.grid()
plt.show()
```

```
np.random.seed(1)
t = fx + 50 * np.random.rand(len(x))
plt.plot(x, t, 'o')
plt.grid()
plt.show()
```



지도 학습 알고리즘

■ 선형 모델 실습

- gradient loss, gradient descent, show line function 정의

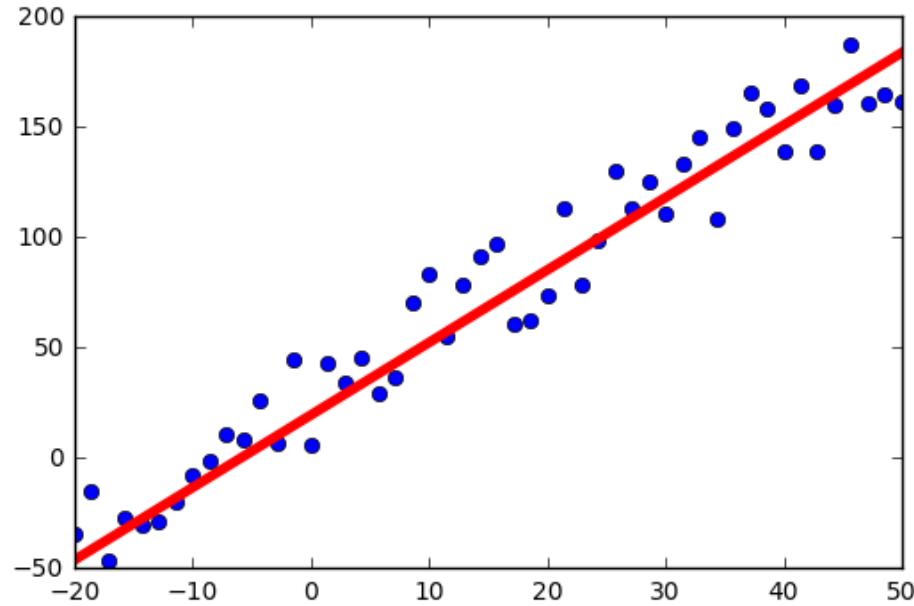
```
def grad_loss(x, t, w0, w1):  
    y = w0 * x + w1  
    grad_w0 = 2*np.mean((y-t) * x)  
    grad_w1 = 2*np.mean(y - t)  
    return grad_w0, grad_w1  
  
def grad_descent(x, t, w0, w1, lr, itr):  
    _w0 = w0  
    _w1 = w1  
    eps = 0.1  
    for i in range(1, itr):  
        grad_w = grad_loss(x, t, _w0, _w1)  
        _w0 = _w0 - lr*grad_w[0]  
        _w1 = _w1 - lr*grad_w[1]  
        if (max(np.absolute(grad_w)) < eps):  
            break  
    return _w0, _w1  
  
def show_line(x, t, w0, w1):  
    # true - dot  
    plt.plot(x, t, 'o')  
    # model - line  
    y = w0*x + w1  
    plt.plot(x, y, color='red', linewidth = 4)
```

지도 학습 알고리즘

- 선형 모델 실습
 - 함수 실행

```
w0 = 1.0
w1 = 1.0
lr = 0.001
itr = 1000
w0_opt, w1_opt = grad_descent(x, t, w0, w1, lr, itr)
print('W = {0:.3f}, {1:.3f}'.format(w0_opt, w1_opt))
show_line(x, t, w0_opt, w1_opt)
```

W = 3.285, 19.581



지도 학습 알고리즘

■ 선형 모델 – 회귀

- 입력 특성에 대한 선형 함수를 만들어 예측 수행
- 회귀의 선형 모델

$$\hat{y} = w[0] * x[0] + b$$

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b$$

- 특성이 한 개인 경우: 직선
- 특성이 두 개인 경우: 평면
- 특성이 N 개인 경우: 초평면
- 최소 제곱법

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2$$

Cost function for simple linear model



지도 학습 알고리즘

- 선형 모델 – 회귀

- 릿지 회귀

- 회귀를 위한 선형 모델의 일종
 - 가중치의 절대값을 가능한 한 작게 만듦
 - 특성이 출력에 주는 영향 최소화
 - L²norm 규제 (regularization)

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p w_j^2$$

Cost function for ridge regression



지도 학습 알고리즘

- 선형 모델 – 회귀

- 라쏘

- 릿지와 마찬가지로 가중치의 절대값을 가능한 한 작게 만듦
 - L¹norm 규제 사용

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p |w_j|$$

Cost function for Lasso regression

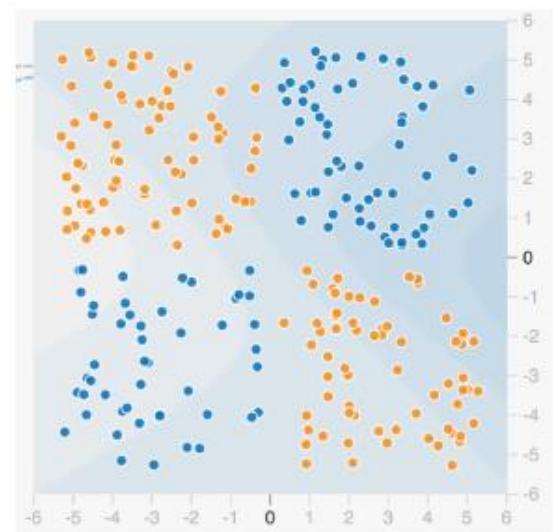
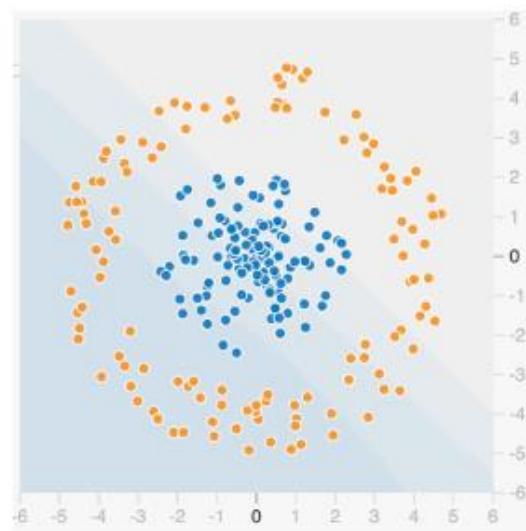
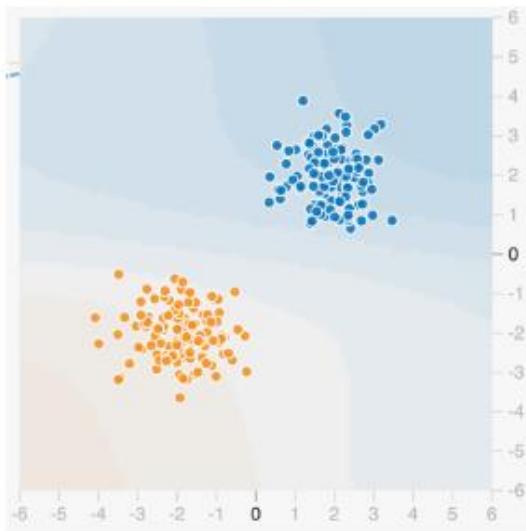


지도 학습 알고리즘

- 회귀 vs. 분류
 - 회귀문제
 - 목표 데이터: 연속된 수치
 - 분류문제
 - 목표 데이터: 클래스 (카테고리, 라벨)
 - 확률 개념 도입



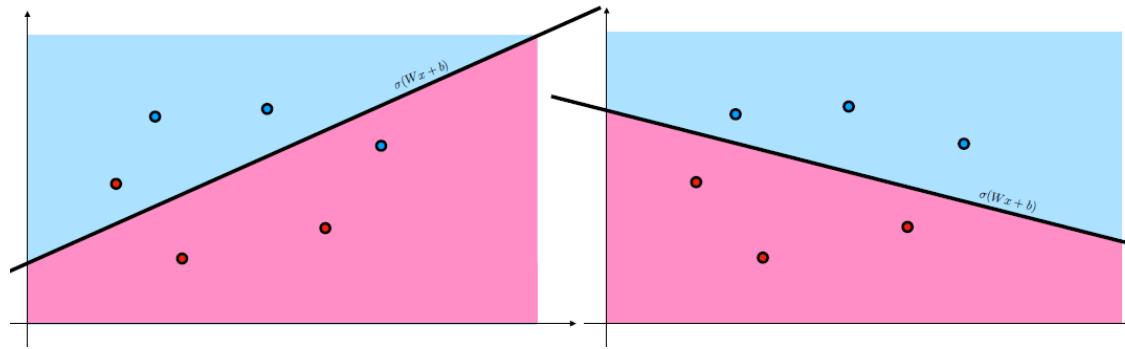
2클래스 분류(Binary Classification)



2클래스 분류 – 개념

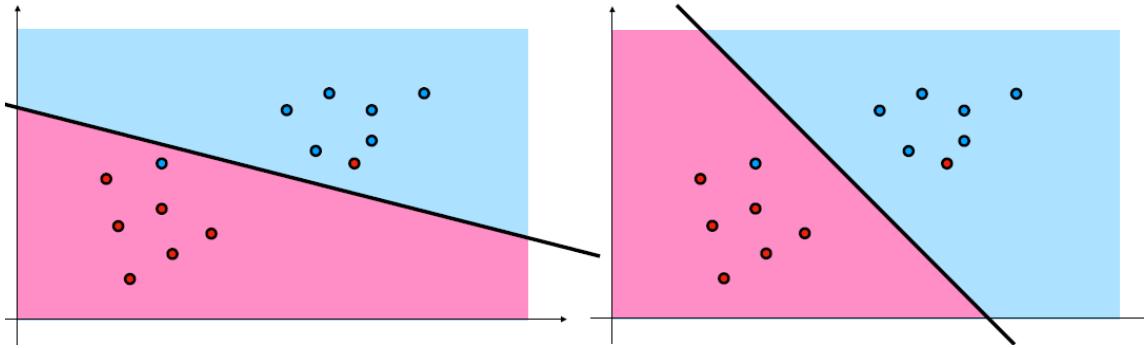
- 좋은 모델이란?

- 예측이 틀린 개수가 적을수록 좋은 모델



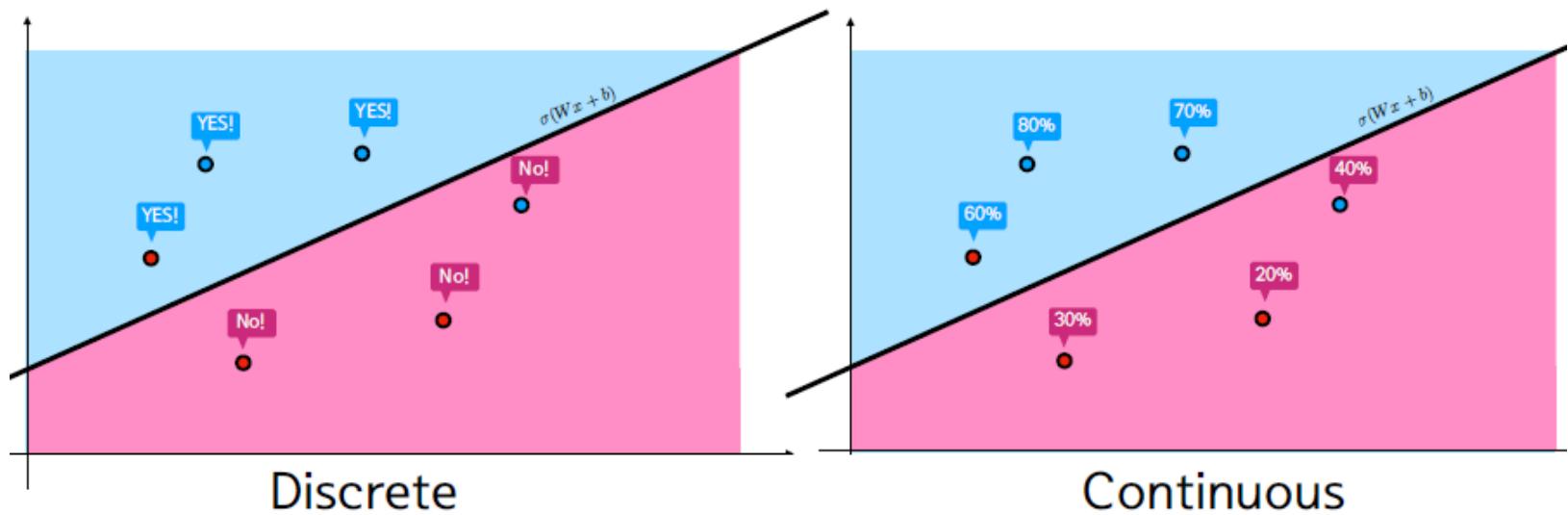
- 한계점

- Continuous 평가 방식 필요



2클래스 분류 – 개념

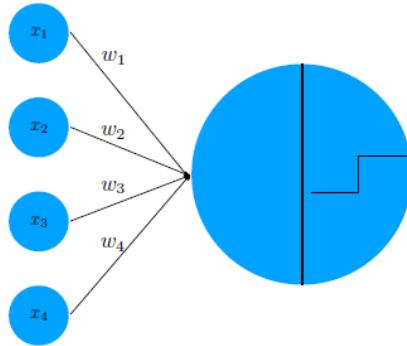
- 확률 개념 도입
 - 예측을 맞출 확률이 최대가 되도록: 최대 가능성 추정
 - Activation function 등장
- 2클래스 분류 접근법



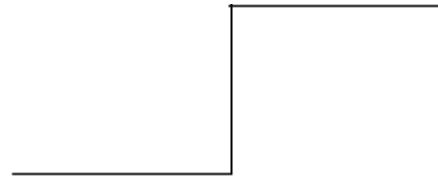
2클래스 분류 – 개념

■ Activation function

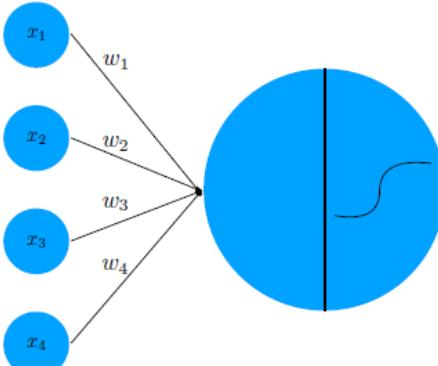
- Discrete: Step function



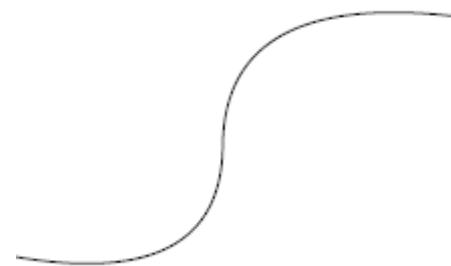
$$f(w, x) = \text{sign}(XW)$$



- Continuous: Sigmoid function



$$f(w, x) = \frac{1}{1 + e^{-XW}}$$



2클래스 분류 – 개념

- Loss function

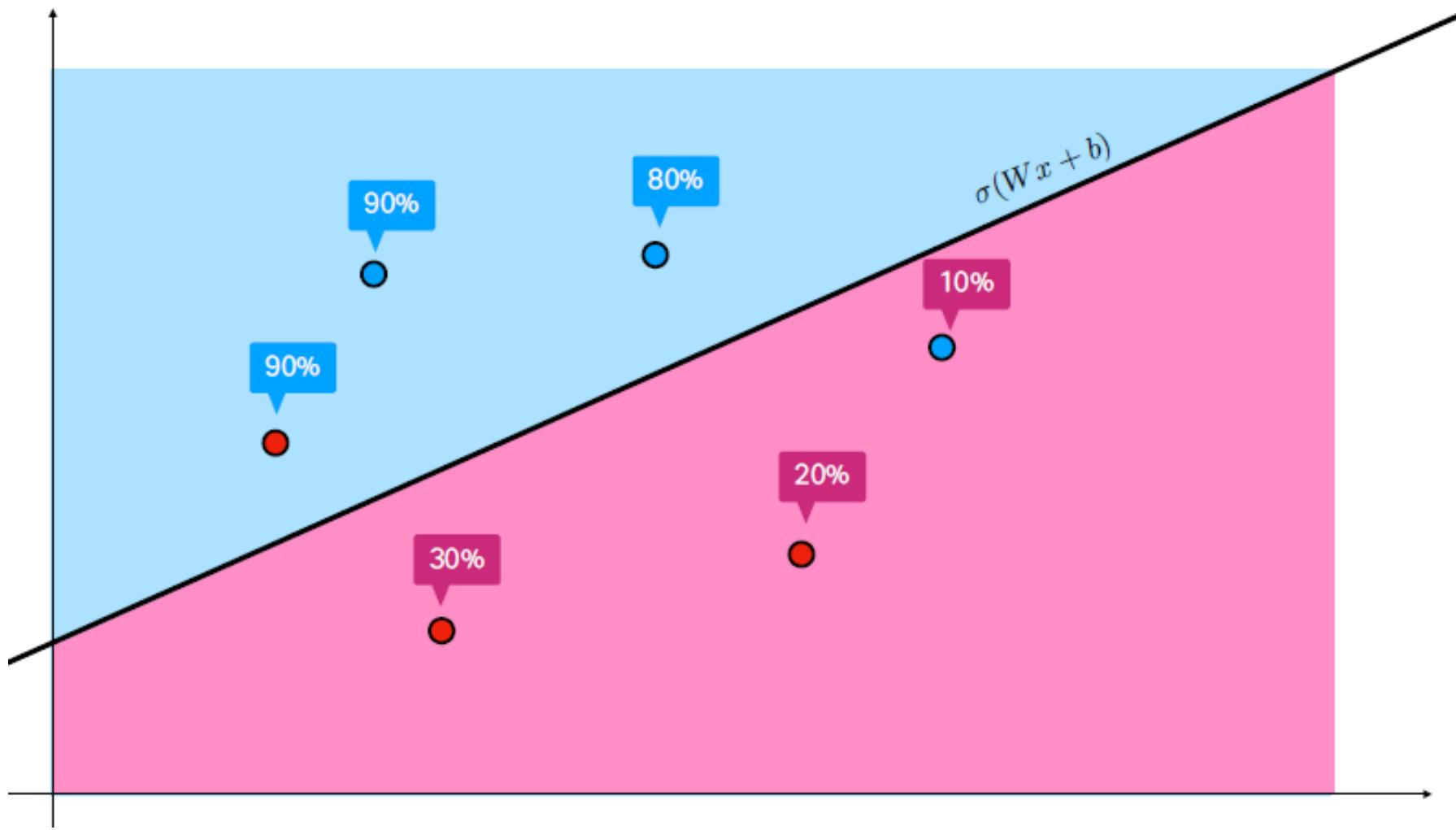
- Discrete
 - 틀린 예측값 개수
- Continuous
 - Cross-Entropy

$$E = \sum_i -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

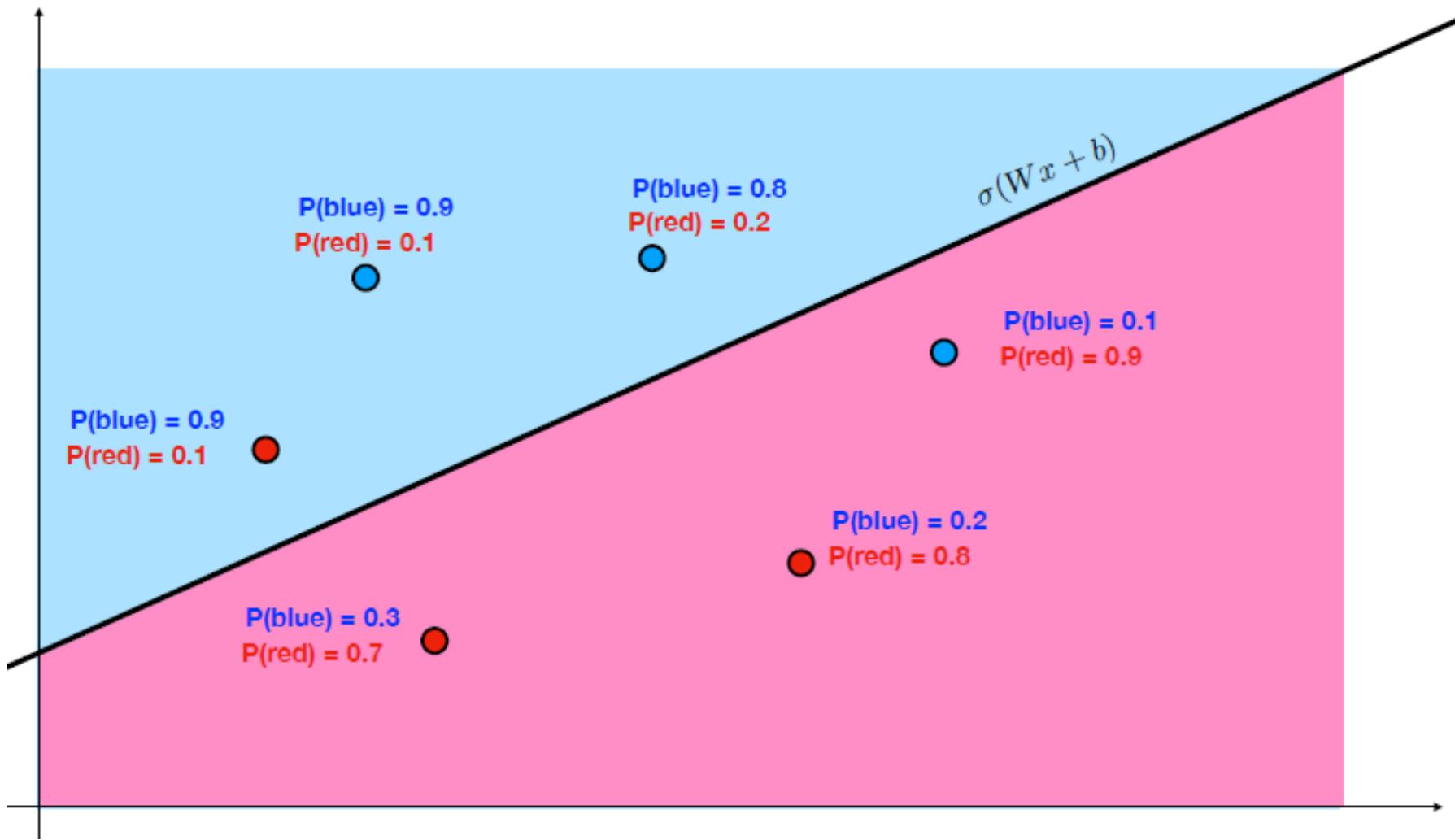
- 의미?



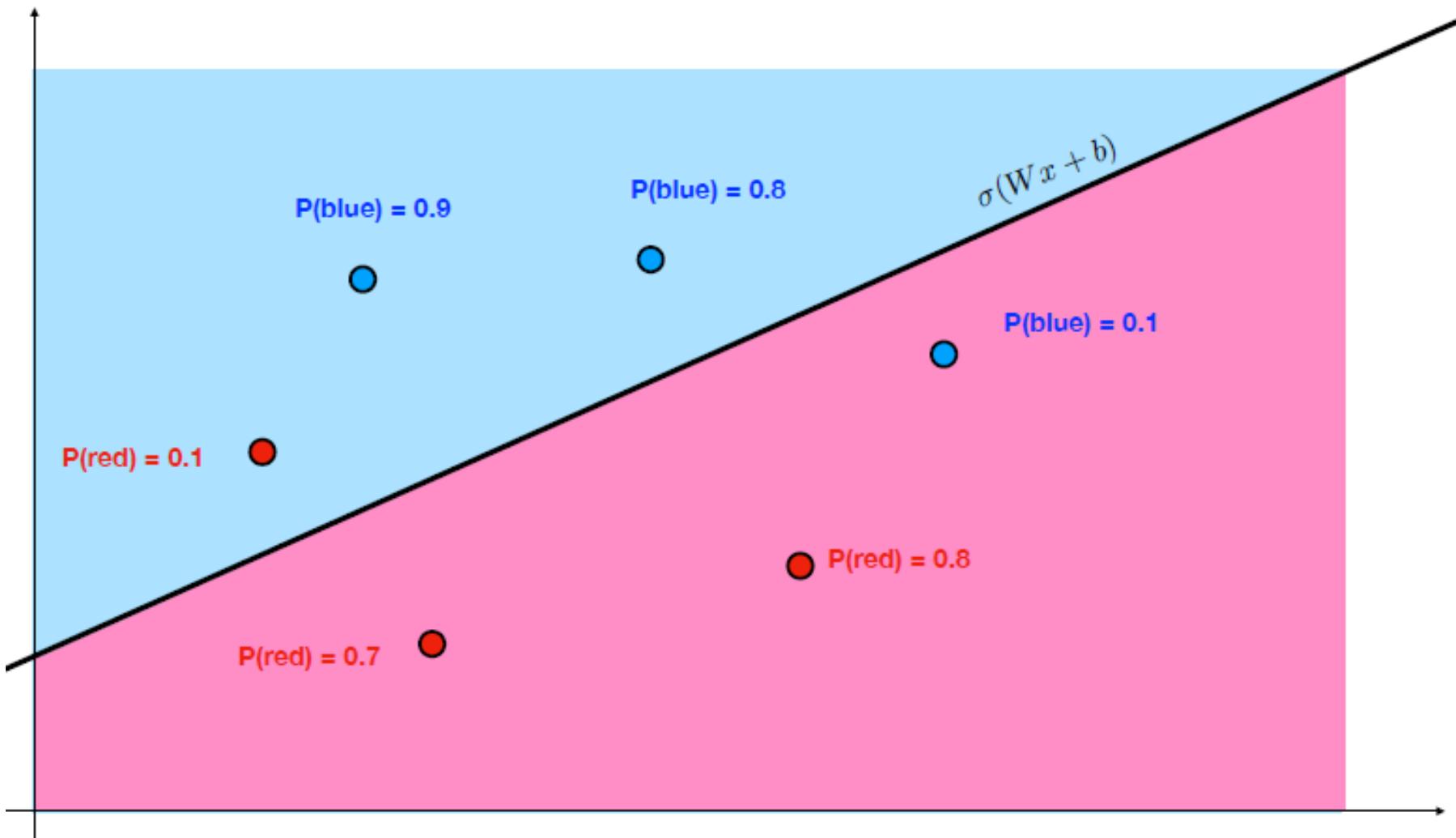
2클래스 분류 – 개념



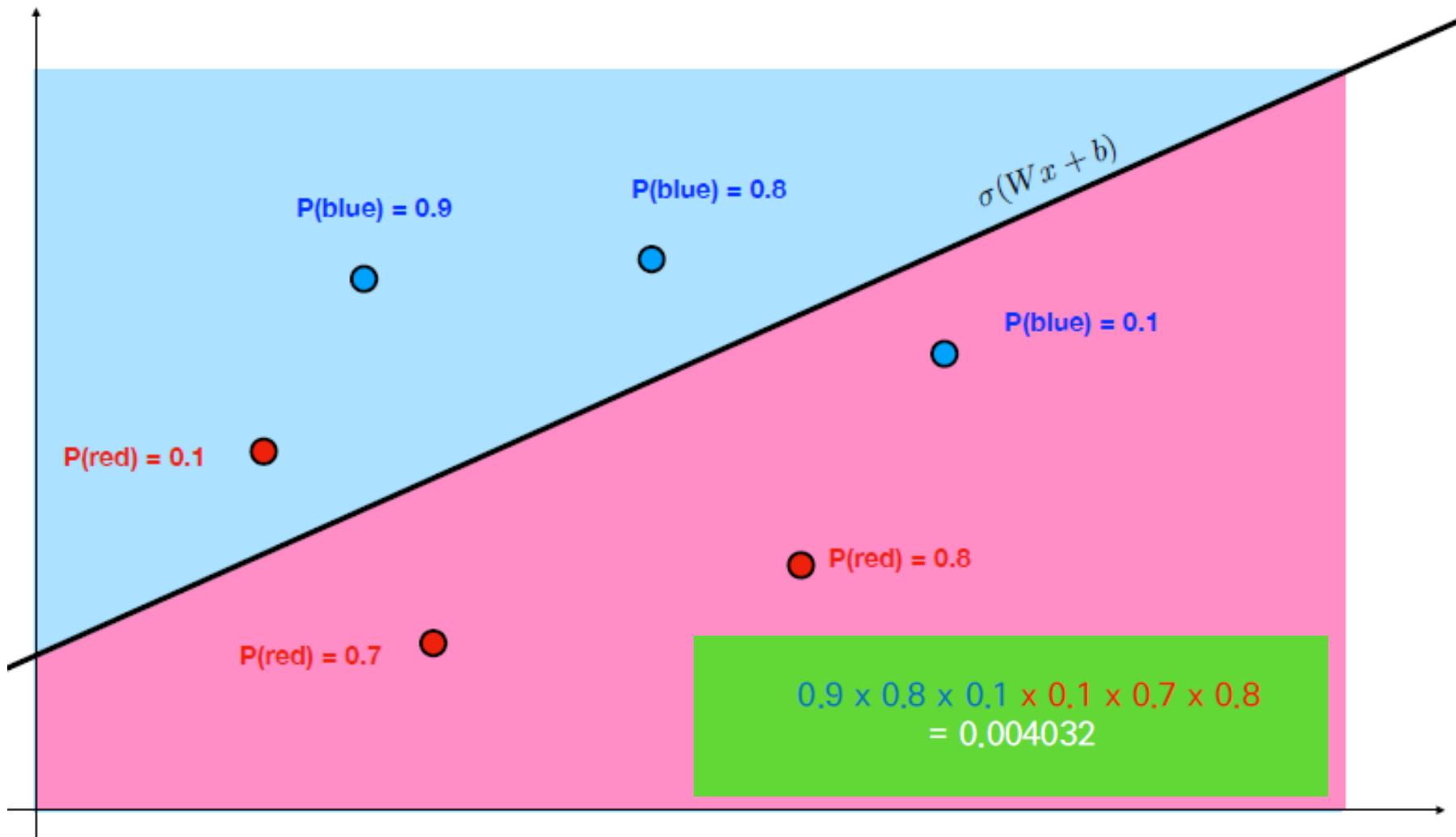
2클래스 분류 – 개념



2클래스 분류 – 개념



2클래스 분류 – 개념



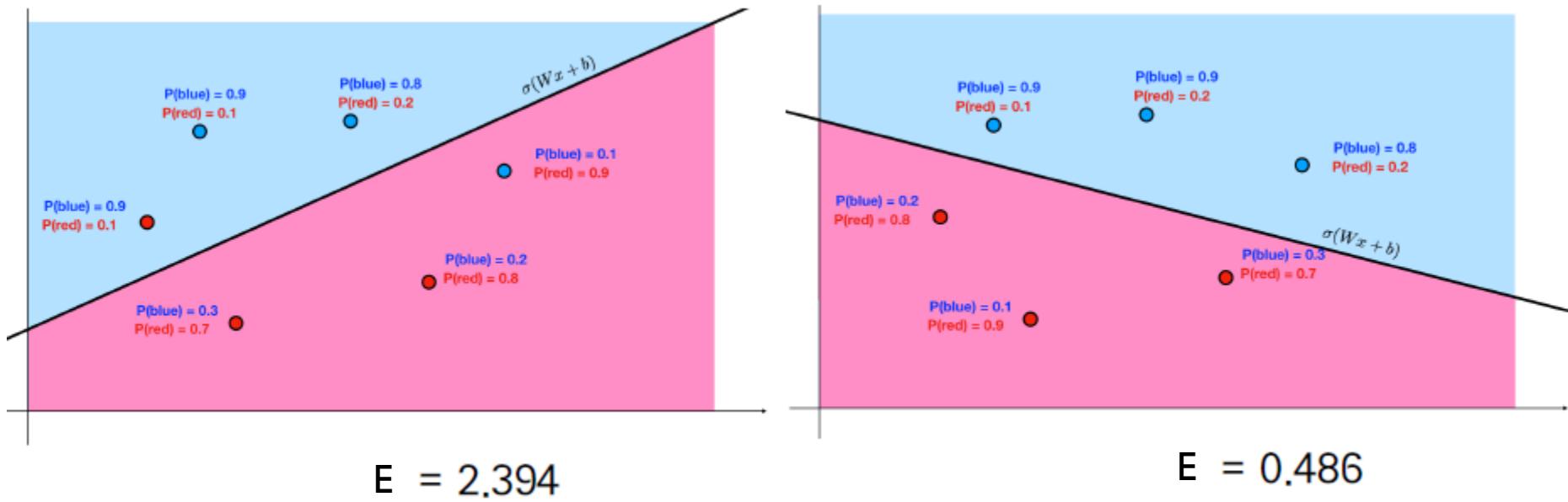
2클래스 분류 – 개념

- 최대 가능도 추정
 - 확률을 최대로 만드는 모델을 찾는 방법
 - 확률의 곱으로 값을 표현
 - 값이 너무 작아짐
 - 최적화 시 효율적인 계산
 - 이러한 이유로 log를 붙이고 덧셈으로 변경
 - 마이너스를 붙여 최소값 문제로 변경

$$E = \sum_i -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$



2클래스 분류 – 개념



어떤 모델이 좋은 모델인가?



지도학습 알고리즘

- 1차원 입력 2클래스 분류
 - 로지스틱 회귀 모델
 - 시그모이드 함수의 파라미터로 직선의 식 대입

$$y = \frac{1}{1+e^{-x}} \quad \rightarrow \quad y = \frac{1}{1+e^{-(w_0x + w_1)}}$$

- 직선 식의 값이 큰 양의 값인 경우: y 는 1에 가까운 값
- 직선 식의 값이 절대값이 큰 음의 값인 경우: y 는 0에 가까운 값
- 결국 y 는 0 ~ 1 사이가 됨



지도학습 알고리즘

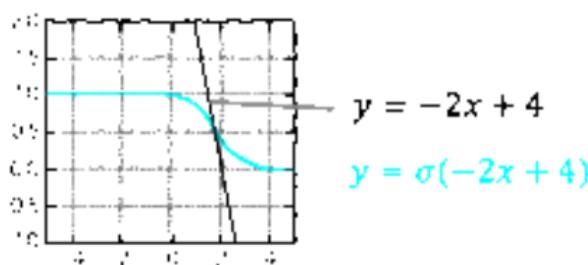
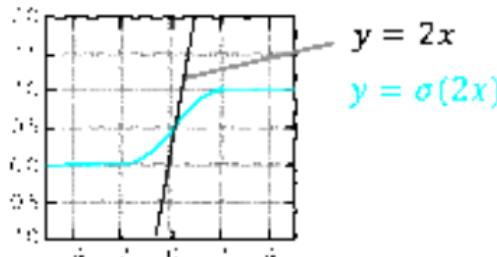
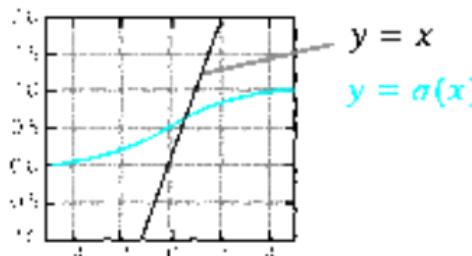
- 1차원 입력 2클래스 분류
 - 로지스틱 회귀 모델

$$y = w_0x + w_1$$



$$y = \sigma(w_0x + w_1)$$

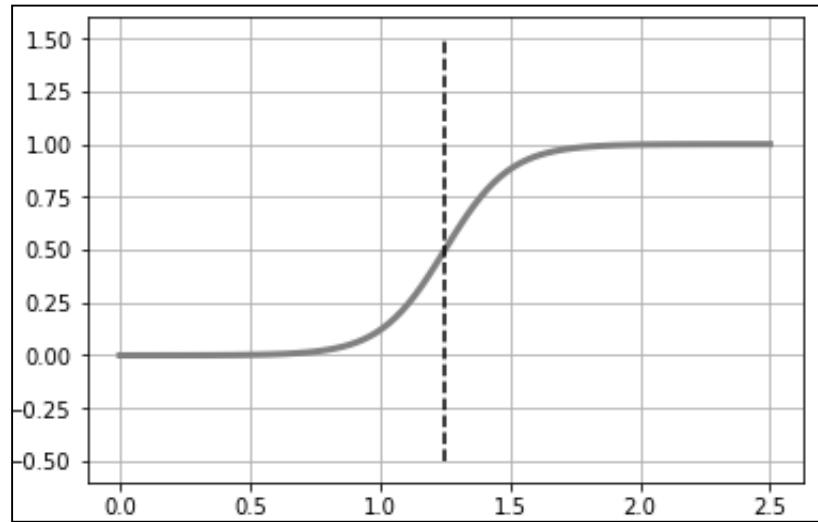
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



1차원 입력 2클래스 분류

- 로지스틱 회귀 모델
 - 구현

```
In [8]: def logistic(x, w):  
    y = 1/(1+np.exp(-(w[0]*x+w[1])))  
    return y
```



지도 학습 알고리즘

- 2차원 2클래스 분류

- 데이터 만들기

- 선형 식을 정의하고 클래스 설정

- $$\text{식} = w_0 * x_0 + w_1 * x_1 + w_2$$

- 직선을 기준으로 아래에 있는 데이터는 클래스 0

- 직선을 기준으로 위에 있는 데이터는 클래스 1

- 데이터 분포 가시화 ($N = 50$)

- $3/4x_0 + 1.0x_1 - 4/5 = 0$ 식을 이용하여 클래스 지정

- 비용 함수 설정

- Cross-Entropy

- $$E(W) = -\frac{1}{N} \sum_{n=0}^{N-1} \{tn \log yn + (1 - tn) \log(1 - yn)\}$$

- 최적화

- 경사 하강법 사용



지도 학습 알고리즘

- 2차원 2클래스 분류
 - 데이터 만들기

```
np.random.seed(seed=1) # 난수를 고정
W = np.array([3./4., 1.0, -4./5.])
N = 50
dim = 2
K = 2
scale = 1;
T = np.zeros((N, K), dtype=np.uint8)
X = scale*np.random.rand(N, dim)
print(X.shape)
```

(50, 2)

```
for n in range(N):
    for k in range(K):
        if W[0]*X[n, 0]+W[1]*X[n, 1]+W[2] > 0:
            T[n, 1] = 1
        else:
            T[n, 0] = 1

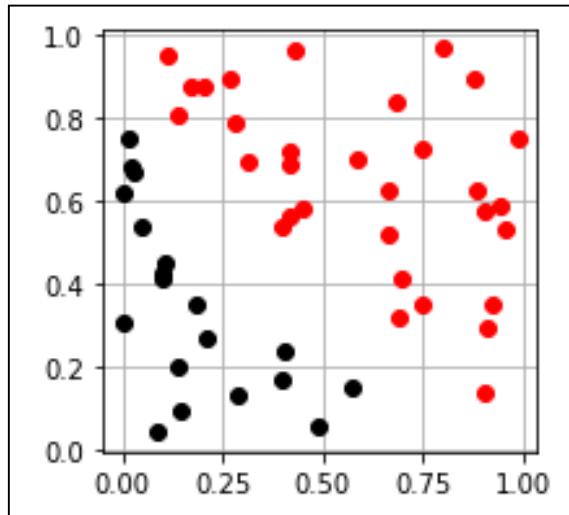
print(X[:5, :])
print(T[:5, :])
```

```
[[4.17022005e-01 7.20324493e-01]
 [1.14374817e-04 3.02332573e-01]
 [1.46755891e-01 9.23385948e-02]
 [1.86260211e-01 3.45560727e-01]
 [3.96767474e-01 5.38816734e-01]]
[[0 1]
 [1 0]
 [1 0]
 [1 0]
 [0 1]]
```

지도 학습 알고리즘

- 2차원 2클래스 분류
 - 데이터 만들기

```
def show_data(x, t):  
    c = [[0, 0, 0], [1, 0, 0]]  
    for k in range(K):  
        plt.plot(x[t[:, k] == 1, 0], x[t[:, k] == 1, 1], linestyle='none', marker='o', color=c[k])  
    plt.grid(True)  
  
plt.figure(figsize=(3, 3))  
show_data(X, T)
```



지도 학습 알고리즘

- 2차원 2클래스 분류
 - 비용 함수 및 최적화 함수

```
def logistic2(x0, x1, w):  
    y = 1 / (1 + np.exp(-(w[0] * x0 + w[1] * x1 + w[2])))  
    return y
```

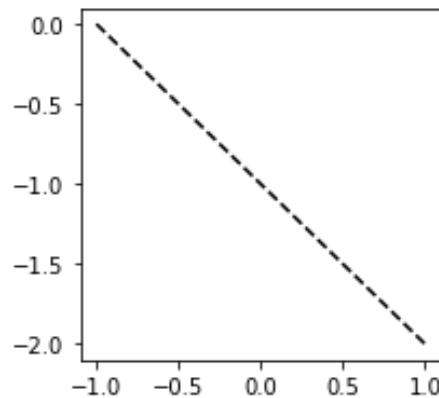
```
def cee_logistic2(w, x, t):  
    X_n = x.shape[0]  
    y = logistic2(x[:, 0], x[:, 1], w)  
    cee = 0  
    for n in range(len(y)):  
        cee = cee - (t[n, 0] * np.log(y[n]) + (1 - t[n, 0]) * np.log(1 - y[n]))  
    cee = cee / X_n  
    return cee  
  
# test ---  
_w=[-1., -1., -1.]  
cee_logistic2(_w, X, T)
```

0.7170005111218646

지도 학습 알고리즘

- 2차원 2클래스 분류
 - 직선 그리기 함수

```
def show_line(W):  
    xn = 50 # 파라미터의 분할 수  
    X_range0 = [-1, 1] # x0 범위 표시 용  
    x0 = np.linspace(X_range0[0], X_range0[1], xn)  
    x1 = -(W[0]/W[1])*x0 - W[2]/W[1]  
    plt.plot(x0, x1, '--k')  
  
# test ---  
plt.figure(figsize=(3,3))  
_W=[-1, -1, -1]  
show_line(_W)
```

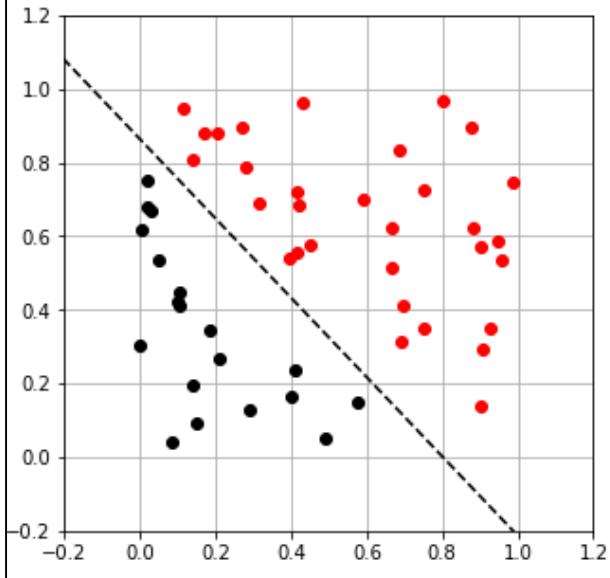


지도 학습 알고리즘

- 2차원 2클래스 분류
 - 최적화 - 경사하강법

```
def fit_logistic(w_init, X, t):  
    #res = minimize(cee_logistic2, w_init, args=(X, t), jac=doee_logistic2, method="CG")  
    res = minimize(cee_logistic2, w_init, args=(X, t), method="CG")  
    return res.x  
  
#W = grad_descent(W_init, X, T, lr, itr)  
_W = fit_logistic(W_init, X, T)
```

w0 = -54.00, w1 = -50.15, w2 = 43.29
CEE = nan



지도 학습 알고리즘

- 선형 모델 – 분류

- 이진 분류

- 두 개의 클래스를 구분하여 예측

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b > 0$$

- 결정 경계

- 대표적인 선형 분류 알고리즘

- 로지스틱 회귀

- 선형 서포트 벡터 머신



지도 학습 알고리즘

- 선형 모델 – 분류

- 다중 분류

- 세 개 이상의 클래스를 구분하여 예측

- 클래스 별로 이진 분류기에 사용되는 계수 벡터와 절편을 가짐

$$w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b$$



나이브 베이즈 모델 및 결정 트리



(Review) 확률과 통계

■ 확률변수 random variable

- 예) 윷

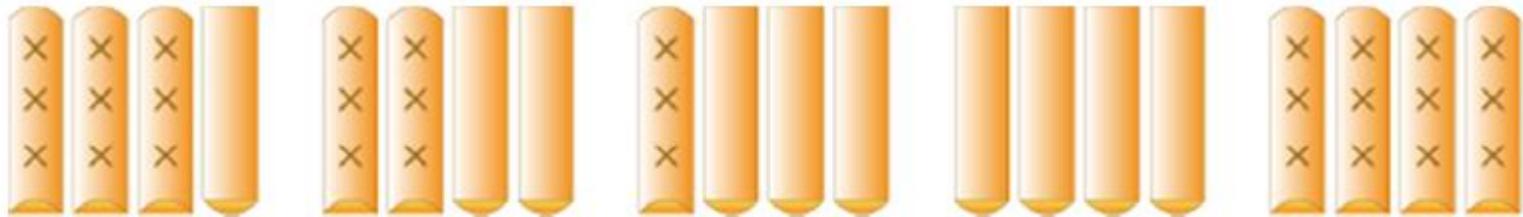


그림 2-13 윷을 던졌을 때 나올 수 있는 다섯 가지 경우(왼쪽부터 도, 개, 걸, 윷, 모)

- 다섯 가지 경우 중 한 값을 갖는 확률변수 x
- x 의 정의역은 {도, 개, 걸, 윷, 모}

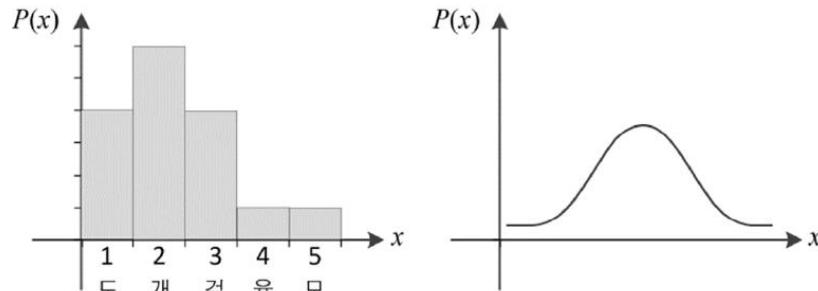


(Review) 확률과 통계

■ 확률분포

- 정의역 전체에 걸쳐 확률을 표현한 것

$$P(x = \text{도}) = \frac{4}{16}, P(x = \text{개}) = \frac{6}{16}, P(x = \text{걸}) = \frac{4}{16}, P(x = \text{윷}) = \frac{1}{16}, P(x = \text{모}) = \frac{1}{16}$$



(a) 이산인 경우의 확률질량함수

(b) 연속인 경우의 확률밀도함수

그림 2-14 확률분포

■ 확률벡터 random vector

- 확률변수가 벡터인 경우

- 예) Iris에서 확률벡터 \mathbf{x} 는 4차원 $\mathbf{x} = (x_1, x_2, x_3, x_4)^T = (\text{꽃받침 길이}, \text{꽃받침 너비}_1, \text{꽃잎 길이}, \text{꽃잎 너비})^T$

(Review) 확률과 통계

■ 간단한 확률실험 장치

- 주머니에서 번호를 뽑은 다음, 번호에 따라 해당 병에서 공을 뽑고 색을 관찰함
- 번호를 y , 공의 색을 x 라는 확률변수로 표현하면 정의역은 $y \in \{1, 2, 3\}$, $x \in \{\text{파랑}, \text{하양}\}$

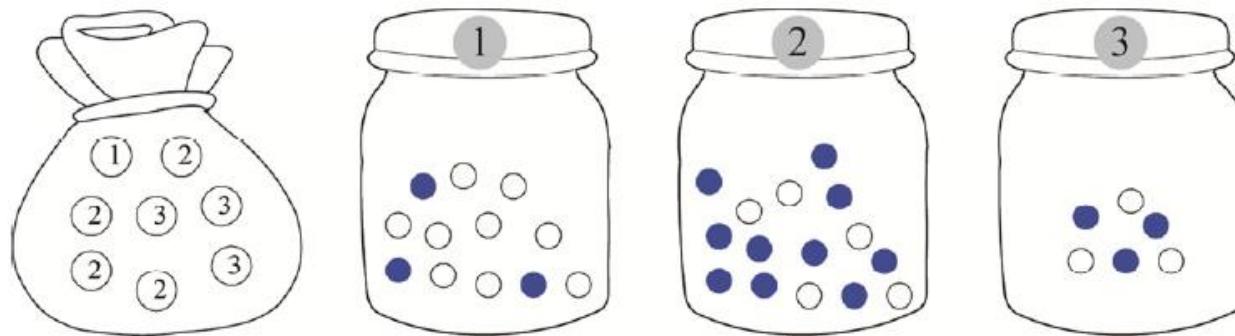


그림 2-15 확률 실험

(Review) 확률과 통계

■ 곱 규칙과 합 규칙

- ①번 카드를 뽑을 확률은 $P(y=①)=P(①)=1/8$
- 카드는 ①번, 공은 하양일 확률은 $P(y=①, x=\text{하양})=P(①, \text{하양}) \leftarrow \text{결합확률}$

$$P(y = ①, x = \text{하양}) = P(x = \text{하양}|y = ①)P(y = ①) = \frac{9}{12} \cdot \frac{1}{8} = \frac{3}{32}$$

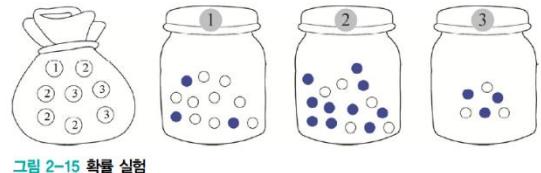


그림 2-15 확률 실험

■ 곱 규칙

곱 규칙: $P(y, x) = P(x|y)P(y)$ (2.23)

■ 하얀 공이 뽑힐 확률

$$\begin{aligned} P(\text{하양}) &= P(\text{하양}|①)P(①) + P(\text{하양}|②)P(②) + P(\text{하양}|③)P(③) \\ &= \frac{9}{12} \cdot \frac{1}{8} + \frac{5}{15} \cdot \frac{4}{8} + \frac{3}{6} \cdot \frac{3}{8} = \frac{43}{96} \end{aligned}$$

■ 합 규칙

합 규칙: $P(x) = \sum_y P(y, x) = \sum_y P(x|y)P(y)$ (2.24)



(Review) 확률과 통계

■ 베이즈 정리 (식 (2.26))

$$P(y, x) = P(x|y)P(y) = P(x, y) = P(y|x)P(x)$$

$$\longrightarrow P(y|x) = \frac{P(x|y)P(y)}{P(x)} \quad (2.26)$$

- 다음 질문을 식 (2.27)로 쓸 수 있음

“하얀 공이 나왔다는 사실만 알고 어느 병에서 나왔는지 모르는데, 어느 병인지 추정하라.”

$$\hat{y} = \operatorname{argmax}_y P(y|x) \quad (2.27)$$



(Review) 확률과 통계

■ 베이즈 정리 (식 (2.26))

- 베이즈 정리를 적용하면, $\hat{y} = \operatorname{argmax}_y P(y|x = \text{하양}) = \operatorname{argmax}_y \frac{P(x = \text{하양}|y)P(y)}{P(x = \text{하양})}$
 - 세 가지 경우에 대해 확률을 계산하면,

$$P(\textcircled{1}|\text{하양}) = \frac{P(\text{하양}\textcircled{1})P(\textcircled{1})}{P(\text{하양})} = \frac{\frac{9}{12} \cdot \frac{1}{8}}{\frac{43}{96}} = \frac{9}{43}$$

$$P(\textcircled{2}|\text{하양}) = \frac{P(\text{하양}\textcircled{2})P(\textcircled{2})}{P(\text{하양})} = \frac{\frac{5}{15} \cdot \frac{4}{8}}{\frac{43}{96}} = \frac{16}{43}$$

$$P(\text{③}|\text{복|복}) = \frac{P(\text{복|복})P(\text{③})}{P(\text{복|복})} = \frac{\frac{3}{6} \cdot \frac{3}{8}}{\frac{43}{96}} = \frac{18}{43}$$

③ 번 병일 확률이 가장 높음
(병의 확률과 공의 확률 모두 고려)

■ 베이즈 정리의 해석

$$\overbrace{P(y|x)}^{\text{사후회률}} = \frac{\overbrace{P(x|y)}^{\text{우도}} \overbrace{P(y)}^{\text{사전회률}}}{\overbrace{P(x)}^{\text{우도}}}$$

(Review) 확률과 통계

■ 기계 학습에 적용

- 예) Iris 데이터 분류 문제

- 특징 벡터 \mathbf{x} , 부류 $y \in \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- 분류 문제를 argmax 로 표현하면 식 (2.29)

$$\hat{y} = \underset{y}{\text{argmax}} P(y|\mathbf{x}) \quad (2.29)$$

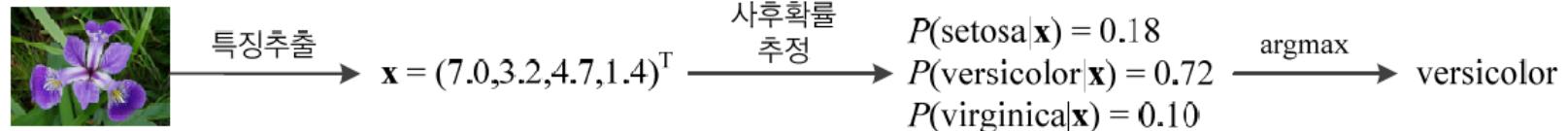


그림 2-16 붓꽃의 부류 예측 과정

- 사후확률 $P(y|\mathbf{x})$ 를 직접 추정하는 일은 아주 단순한 경우를 빼고 불가능
- 따라서 베이즈 정리를 이용하여 추정함

$$\widehat{P}(y|x) = \frac{\widehat{P}(x|y) \widehat{P}(y)}{\widehat{P}(x)}$$

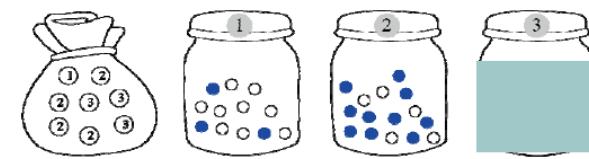
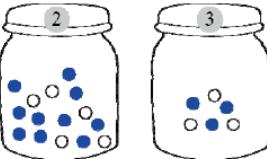


(Review) 확률과 통계

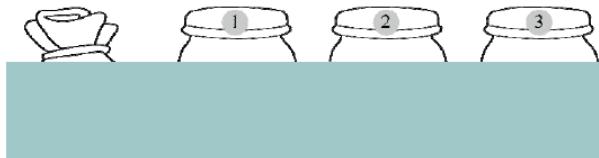
■ 매개변수 Θ 를 모르는 상황에서 매개변수를 추정하는 문제



(a) $\Theta = \{p_1, p_2\}$



(b) $\Theta = \{q_3\}$



(c) $\Theta = \{p_1, p_2, q_1, q_2, q_3\}$

그림 2-17 매개변수가 감추어진 여러 가지 상황

- 예) [그림 2-17(b)] 상황

데이터집합 $X = \{\bullet \circ \circ \bullet \circ \bullet \circ \circ \bullet \bullet \bullet \circ \circ\}$

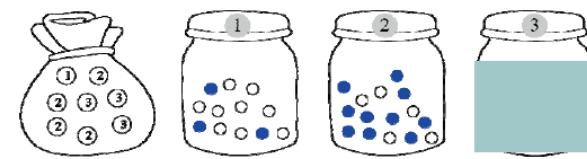
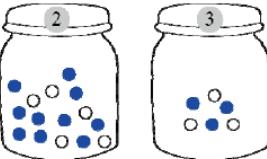
“데이터 X 가 주어졌을 때, X 를 발생시켰을 가능성을 최대로 하는 매개변수 $\Theta = \{q_3\}$ 의 값을 찾아라.”

(Review) 확률과 통계

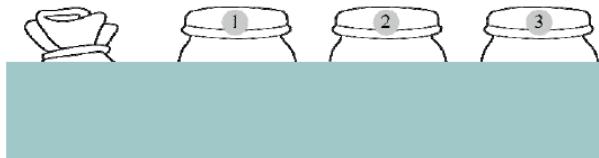
■ 매개변수 Θ 를 모르는 상황에서 매개변수를 추정하는 문제



(a) $\Theta = \{p_1, p_2\}$



(b) $\Theta = \{q_3\}$



(c) $\Theta = \{p_1, p_2, q_1, q_2, q_3\}$

그림 2-17 매개변수가 감추어진 여러 가지 상황

- 예) [그림 2-17(b)] 상황

데이터집합 $X = \{\bullet \circ \circ \bullet \circ \bullet \circ \circ \bullet \bullet \bullet \circ \circ\}$

“데이터 X 가 주어졌을 때, X 를 발생시켰을 가능성을 최대로 하는 매개변수 $\Theta = \{q_3\}$ 의 값을 찾아라.”

(Review) 확률과 통계

■ 최대 우도법

- [그림 2-17(b)] 문제를 수식으로 쓰면,

$$\hat{q}_3 = \operatorname{argmax}_{q_3} P(\mathbb{X}|q_3) \quad (2.31)$$

- 일반화 하면,

$$\text{최대 우도 추정: } \hat{\Theta} = \operatorname{argmax}_{\Theta} P(\mathbb{X}|\Theta) \quad (2.32)$$

- 수치 문제를 피하기 위해 로그 표현으로 바꾸면,

$$\text{최대 로그우도 추정: } \hat{\Theta} = \operatorname{argmax}_{\Theta} \log P(\mathbb{X}|\Theta) = \operatorname{argmax}_{\Theta} \sum_{i=1}^n \log P(\mathbf{x}_i|\Theta) \quad (2.34)$$



지도 학습 알고리즘

- 나이브 베이즈 분류
 - 베이즈 정리

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- 조건부 확률 $P(A|B)$: 사건 B가 발생한 경우 A의 확률
- 결국 $P(A|B)$ 는 $P(A \cap B)$ 와 $P(B)$ 에 기반을 두어야 한다는 정리

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$



지도 학습 알고리즘

■ 베이즈 정리 예제

■ 날씨와 비

- $P(\text{비}) = ?$ $P(\text{비안옴}) = ?$
- $P(\text{비} | \text{맑은날}) = ?$
- $P(\text{맑은날} | \text{비}) = ?$
- $P(\text{맑은날}) = ?$

날씨와 비의 관계표			
	비왔는가?		
	옴	안옴	
맑은날	2	8	10
흐린날	5	5	10
	7	13	20

날씨와 비의 관계표			
	비왔는가?		
	옴	안옴	
맑은날	$P(\text{맑은날} \text{비})$ $= 2/7 = 0.28\dots$	8	$P(\text{맑은날})$ $= 10/20 = 0.5$
흐린날	5	5	10
	$P(\text{비})$ $= 7/20 = 0.35$	13	20

출처: <https://gomguard.tistory.com/69>



지도 학습 알고리즘

■ 나이브 베이즈 분류

- 데이터 셋의 모든 특징은 동등하고 독립적이라 가정

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)} = \frac{P(y)\prod_{i=1}^n P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i \mid y)$$

↓

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i \mid y)$$

- 예) Feature 확장

- 날씨, 바람, 기압, 온도

- 질문) 날씨-좋음, 바람-없음, 기압-높음

온도-낮음 일 때 비가 올까?

- 확률 식으로 표현해보면...

비 와 변수들간 관계표									
	날씨가 좋은가?		바람이 많이 부는가?		기압이 높은가?		온도가 높은가?		
	Yes	No	Yes	No	Yes	No	Yes	No	계
비 온날	2	6	6	2	8	0	5	3	8
안온날	8	4	2	10	2	10	6	6	12
계	10	10	8	12	10	10	11	9	20

출처: <https://gomguard.tistory.com/69>

지도 학습 알고리즘

■ 확률계산

- $A = (\text{비}, \sim\text{비})$
- $B = (\text{날씨 } \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})$
- 결국 $P(\text{비} | B)$ 를 구하는 문제
- $P(\text{비} | B) = ?$

비 와 변수들간 관계표									
	날씨가 좋은가?		바람이 많이 부는가?		기압이 높은가?		온도가 높은가?		
	Yes	No	Yes	No	Yes	No	Yes	No	계
비 온날	2/8	6/8	6/8	2/8	8/8	0/8	5/8	3/8	8
안온날	8/12	4/12	2/12	10/12	2/12	10/12	6/12	6/12	12
계	10	10	8	12	10	10	11	9	20

출처: <https://gomguard.tistory.com/69>



지도 학습 알고리즘

- 나이브 베이즈 분류 적용
 - 스팸 메일 분류
 - 총 E메일: 100개
 - 스팸 E메일 확률: $P(\text{spam}) = 30/100$
 - money라는 단어가 포함된 E메일 확률: $P(\text{money}) = 50/100$
 - 스팸 E메일 중 money를 포함한 E메일 확률: $P(\text{money} | \text{spam}) = 2/5$
 - 그렇다면 money를 포함한 E메일 중 스팸 메일 일 확률,
 - 즉 $P(\text{spam} | \text{money}) = ?$



지도 학습 알고리즘

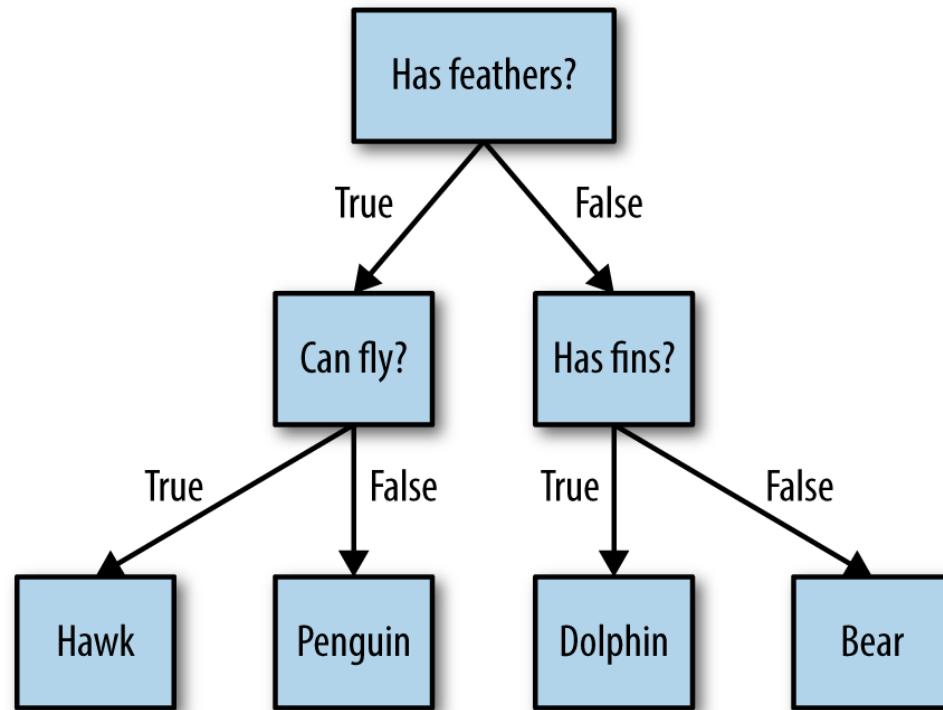
- 나이브 베이즈 분류
 - 특징
 - 분류만 가능
 - 선형 모델보다 훨씬 빠름
 - 대용량 데이터 셋과 고차원 데이터에 가능
 - 선형 모델보다 정확성은 떨어지는 경향이 있음(?)
 - 실습
 - GaussianNB
 - 연속 데이터에 적용
 - BernoulliNB
 - 이진 데이터에 적용
 - MultinomialNB
 - 이산 데이터에 적용



지도 학습 알고리즘

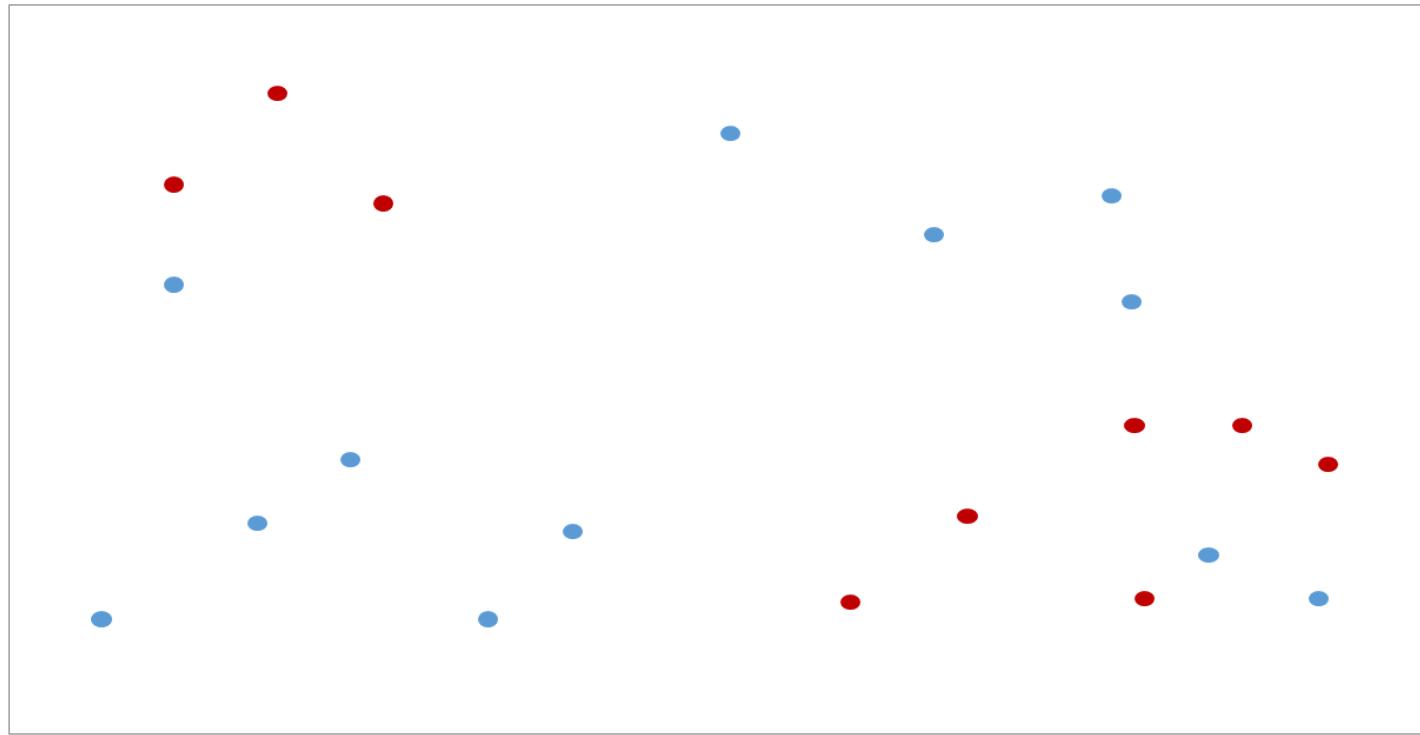
- 결정 트리

- 분류와 회귀 문제에 사용되는 모델
- 예/아니오 질문을 이어가며 학습 (스무고개)
- CART (Classification And Regression Tree)



지도 학습 알고리즘

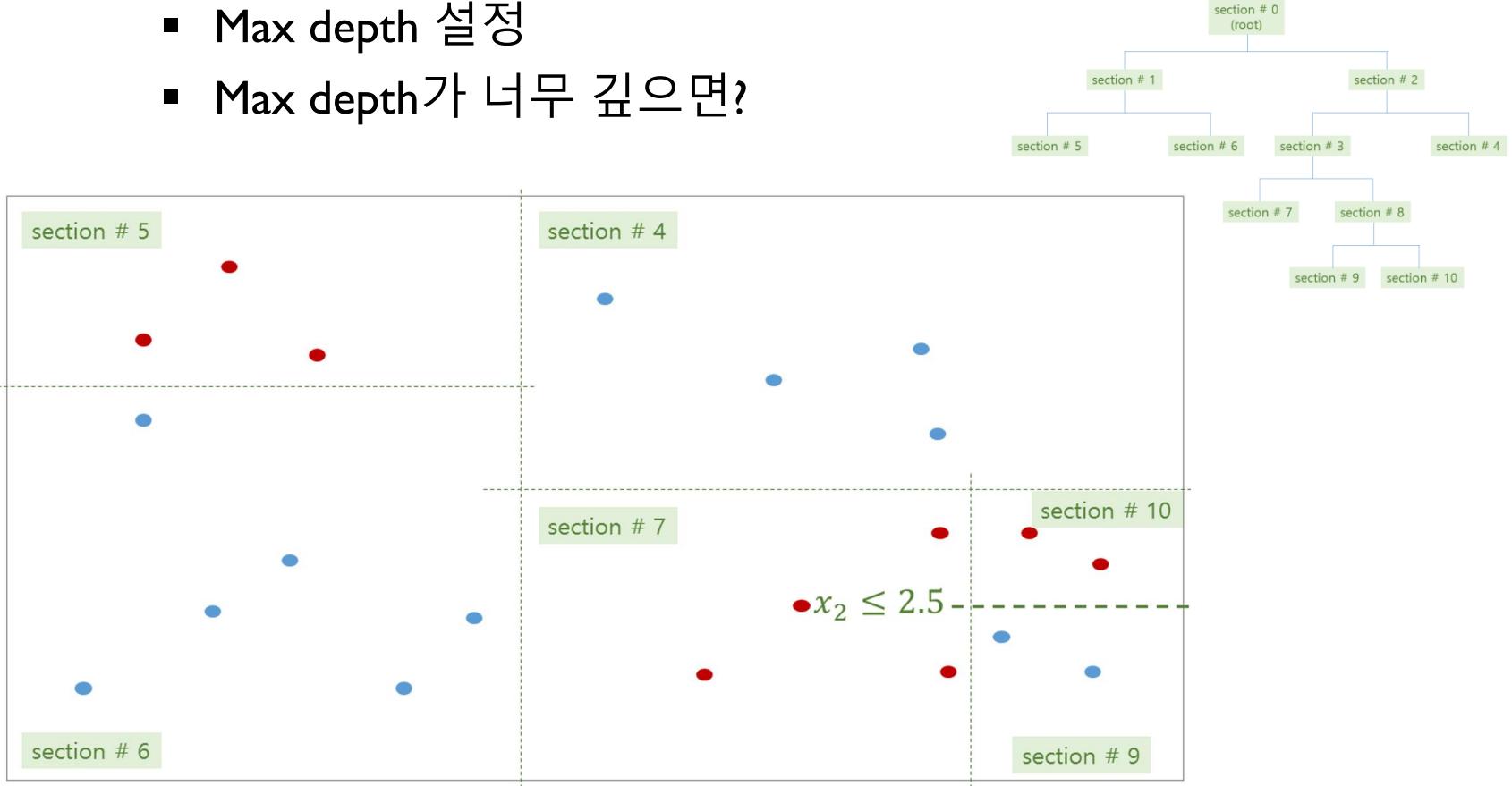
- 결정 트리
 - Clustering처럼 비슷한 공간을 하나의 leaf node로 나눔



지도 학습 알고리즘

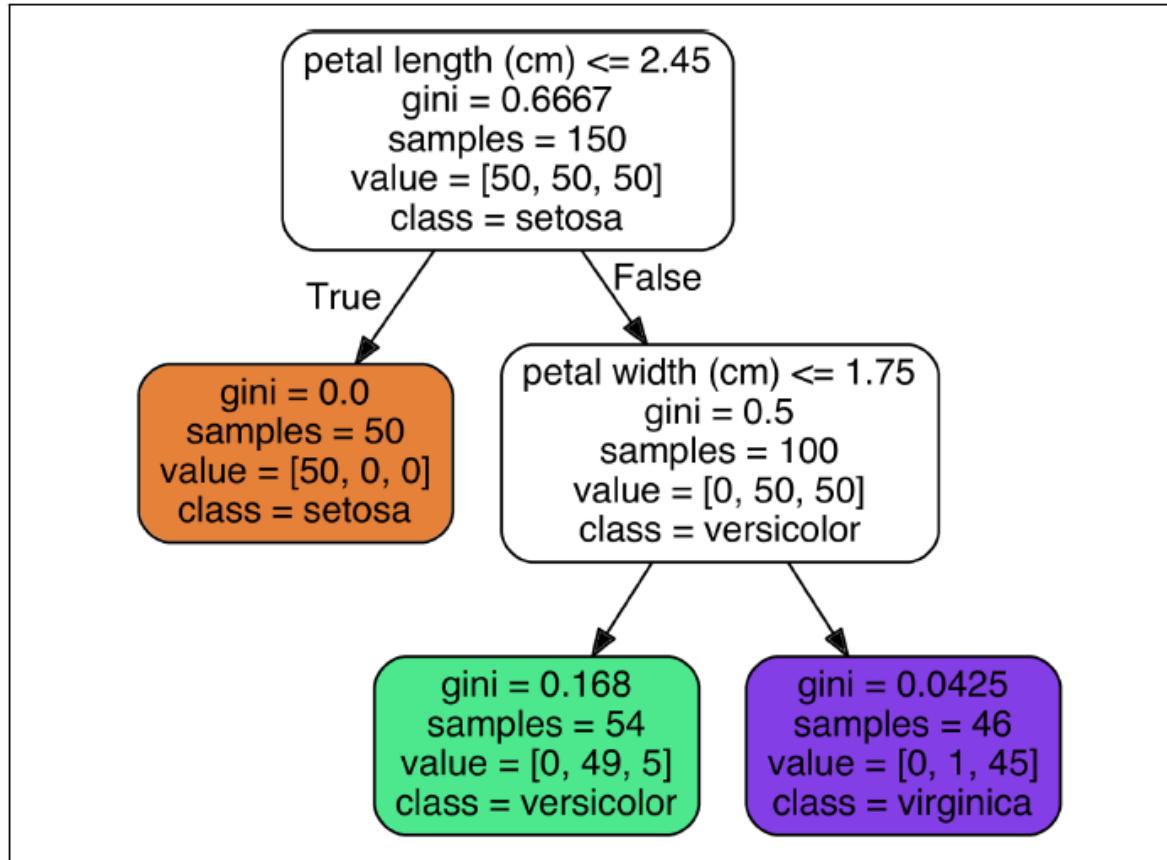
- 결정 트리

- Clustering처럼 비슷한 공간을 하나의 leaf node로 나눔
 - Max depth 설정
 - Max depth가 너무 깊으면?



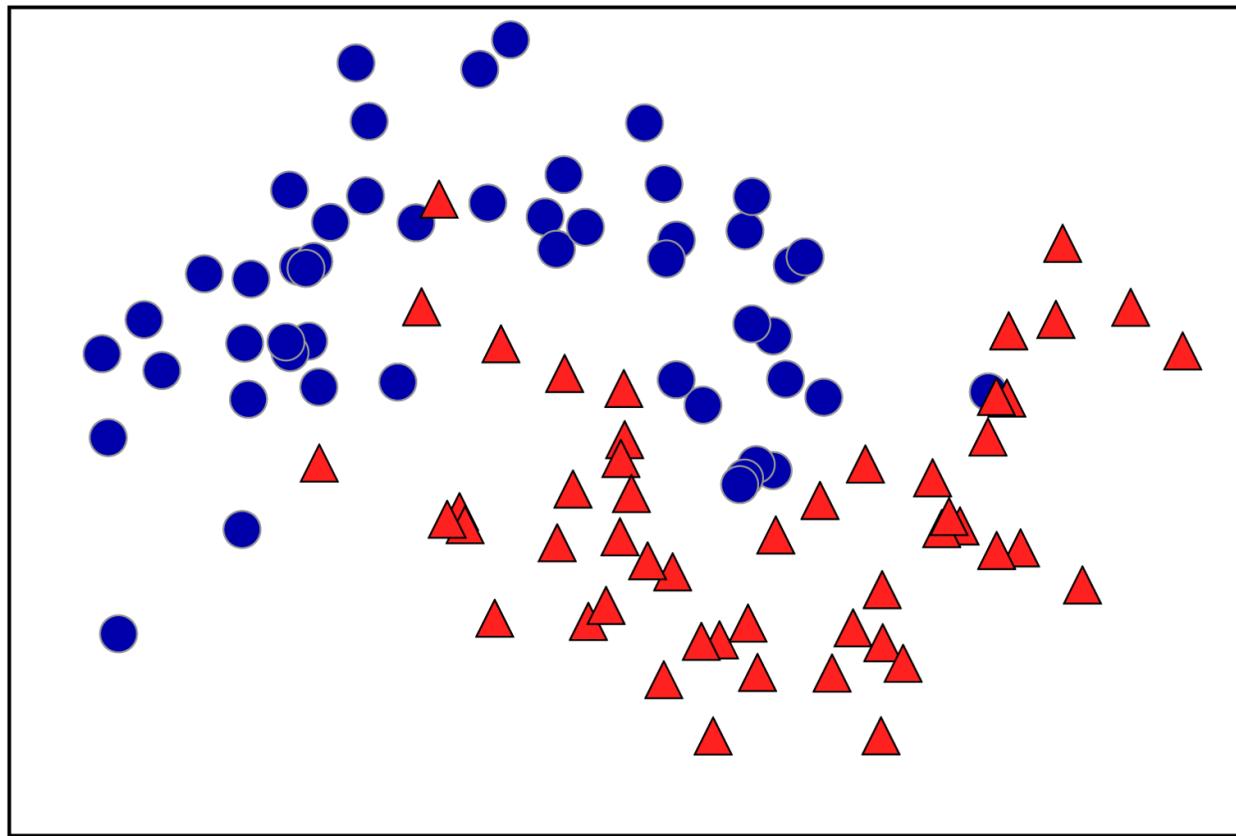
지도 학습 알고리즘

- 결정 트리
 - Iris 데이터에 적용



지도 학습 알고리즘

- 결정 트리 만들기
 - Two-moons dataset



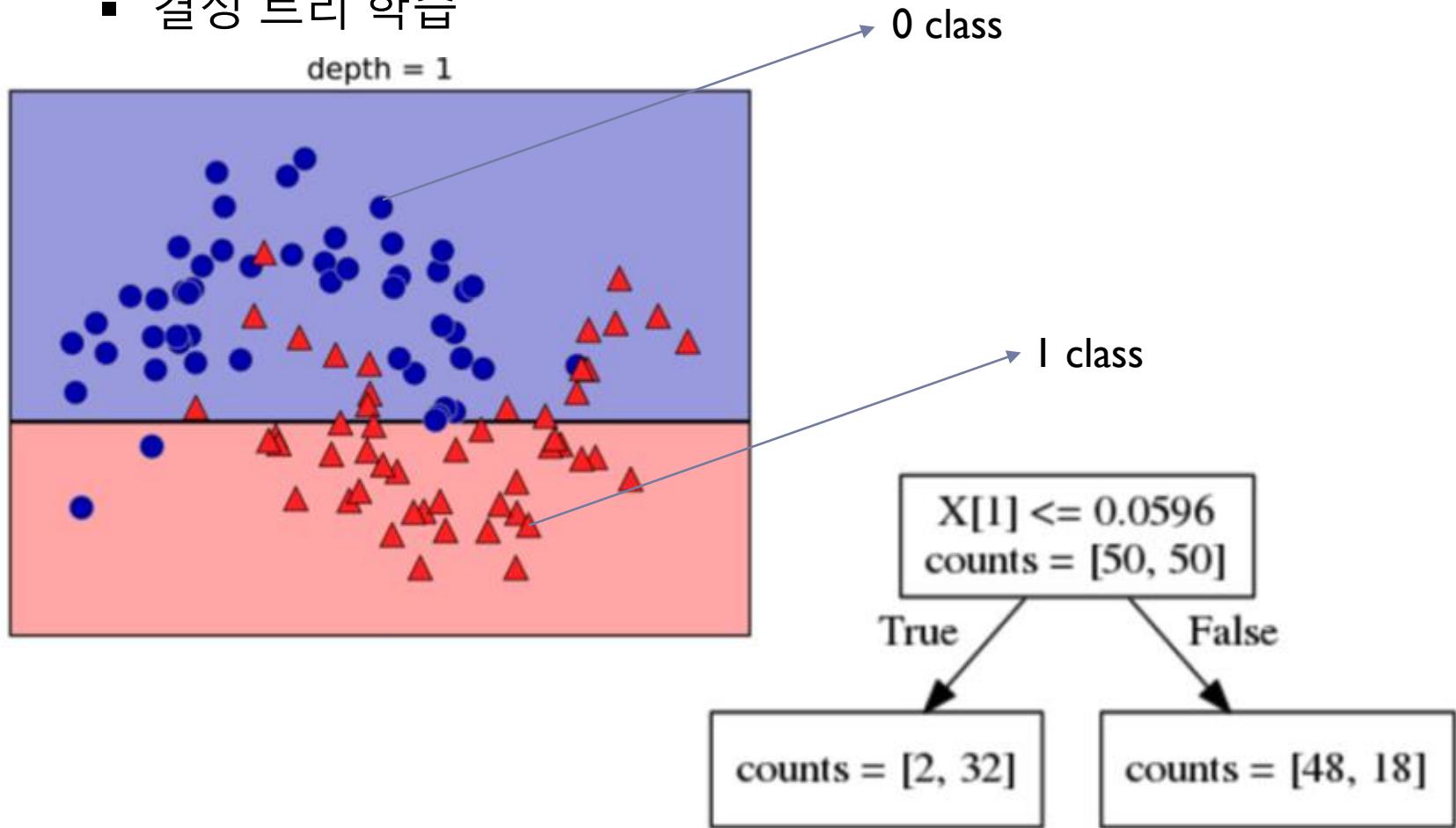
지도 학습 알고리즘

- 결정 트리 만들기
 - 결정 트리 학습
 - 정답에 가장 빨리 도달하는 Yes/No 질문 목록(tests)을 얻는 것
 - 일반적으로 연속적인 데이터에 적용할 tests
 - “Is feature i larger than value a ?”
 - 가능한 모든 tests에서 타깃 값에 대해 가장 많은 정보를 가진 것을 고름
 - 클래스를 가장 잘 나누는 결정 경계 선택



지도 학습 알고리즘

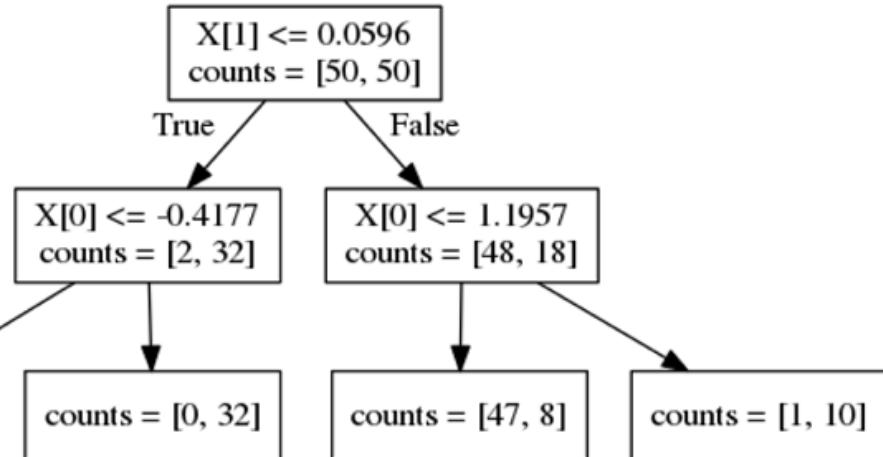
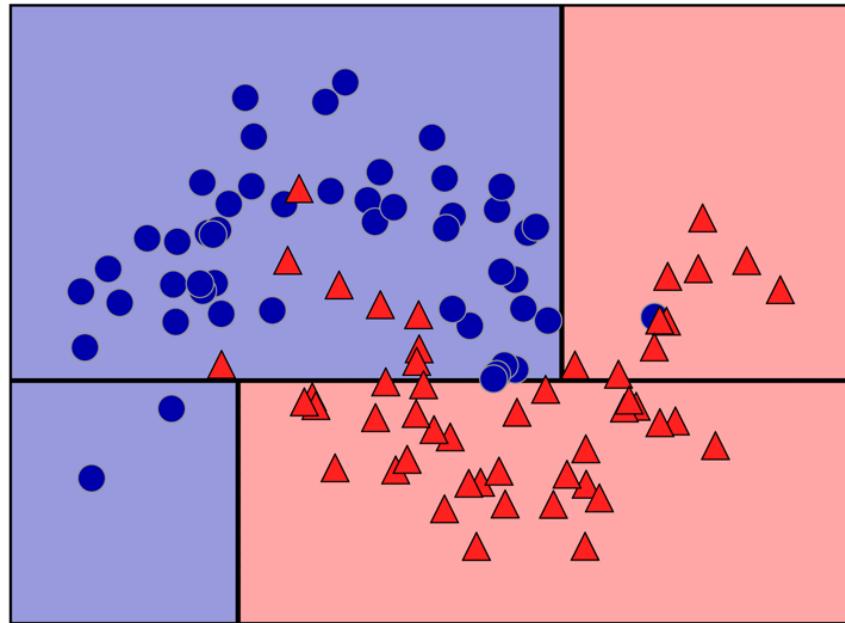
- 결정 트리 만들기
 - 결정 트리 학습



지도 학습 알고리즘

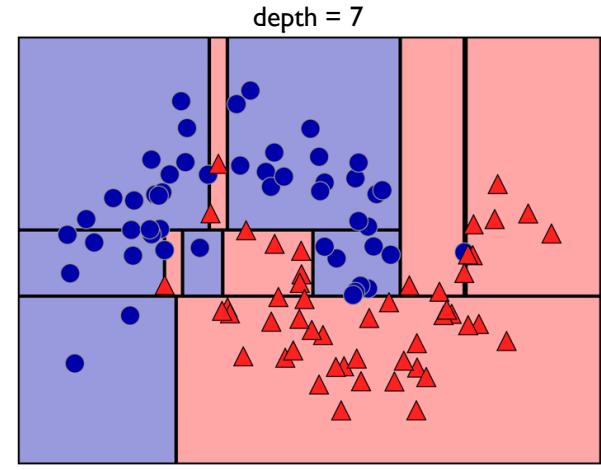
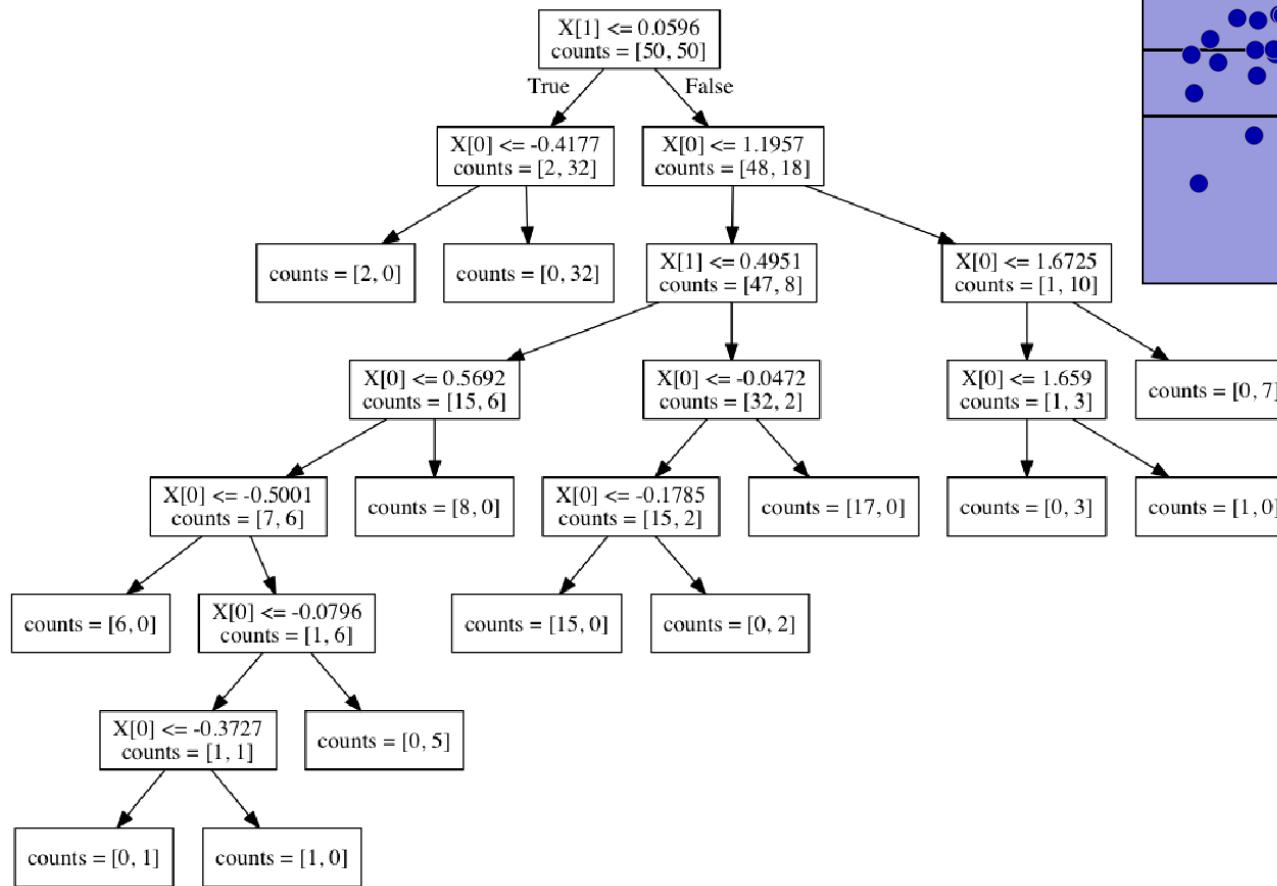
- 결정 트리 만들기
 - 결정 트리 학습

depth = 2



지도 학습 알고리즘

- 결정 트리 만들기
 - 결정 트리 학습



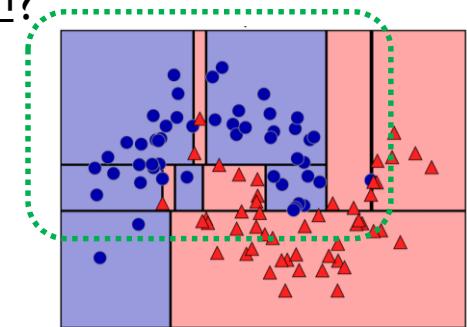
지도 학습 알고리즘

- 결정 트리 복잡도 제어

- 결정 트리 학습

- 모든 leaf node가 pure node라면 학습 데이터의 정확도는?

- 모든 leaf node가 pure node가 될 때까지 학습하면?



- 과대적합(overfitting) 줄이기

- 사전 가지치기 (Pre-pruning)

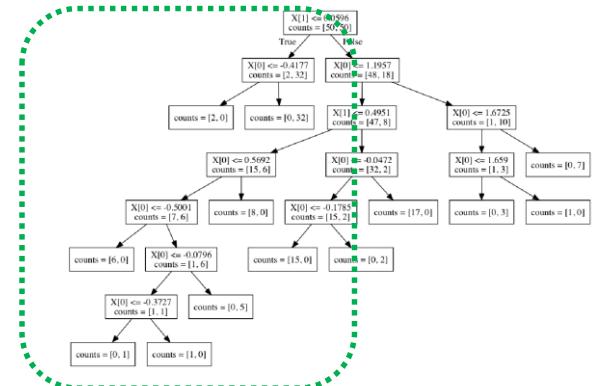
- max depth, leaf 제한

- 노드 분할을 위한 최소 포인트 개수 지정

- 사이킷런 지원

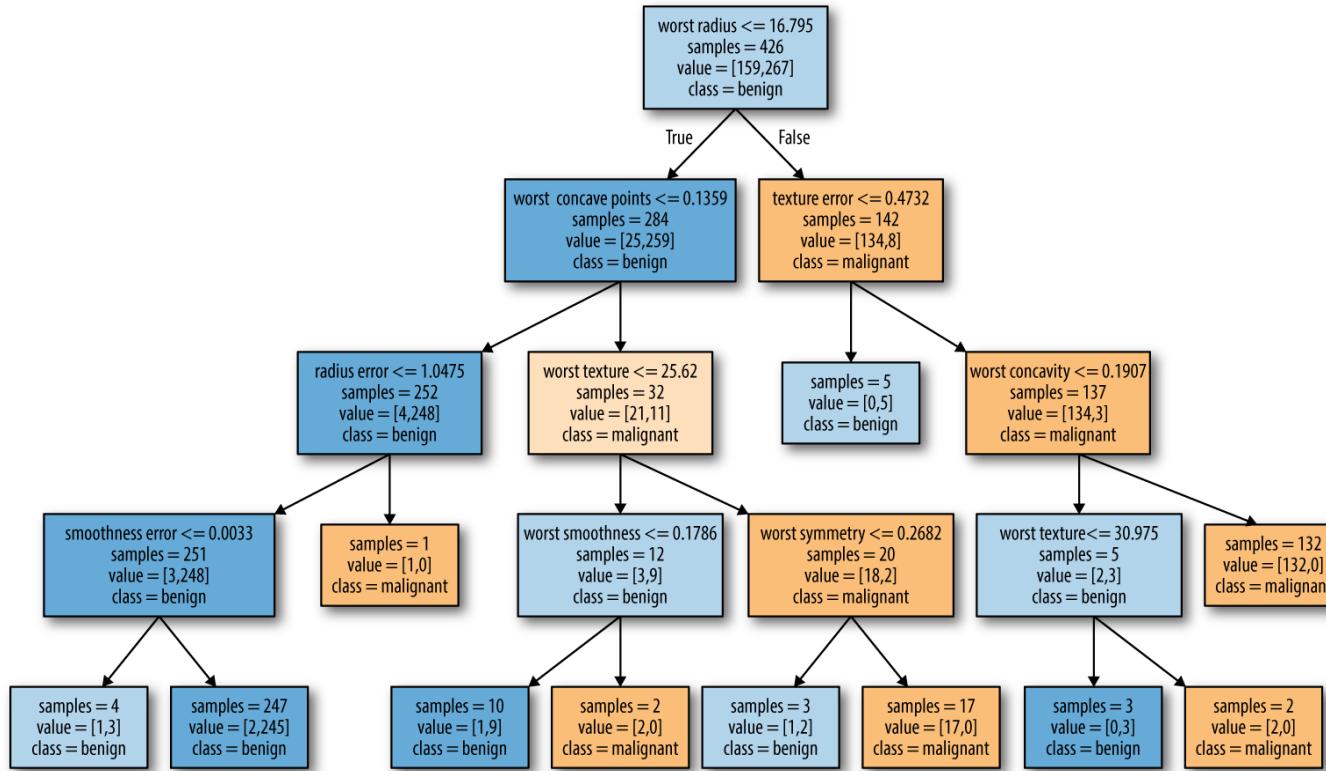
- 사후 가지치기 (Post-pruning)

- 데이터 포인트가 적은 노드 삭제, 병합



지도 학습 알고리즘

- 결정 트리 복잡도 제어
 - 실습
 - 유방암 데이터 셋에 적용



지도 학습 알고리즘

- 결정 트리

- 특성 중요도

- 0 ~ 1 사이의 숫자
 - 각 특성에 대한 중요도를 나타냄
 - 0: 전혀 사용되지 않음
 - 1: 완벽하게 타깃 클래스 예측
 - 전체 합은 1
 - 값이 낮은 특성이 반드시 유용하지 않다는 것은 아님
 - 현재 생성된 트리가 선택을 하지 않았다는 것이고 다른 특성이 동일한 정보를 지니고 있어 낮게 나오는 것일 수도 있음

```
In[62]:
```

```
print("Feature importances:\n{}".format(tree.feature_importances_))
```

```
Out[62]:
```

```
Feature importances:  
[ 0.        0.        0.        0.        0.        0.        0.        0.        0.        0.01  
 0.048    0.        0.        0.002    0.        0.        0.        0.        0.727    0.046  
 0.        0.        0.014    0.        0.018    0.122    0.012    0.      ]
```

비지도 학습



비지도 학습 알고리즘

- 개념
 - 출력 값에 대한 정보 없이 학습 진행
 - 입력 데이터만으로 데이터에서 지식 추출
- 종류
 - 비지도 변환 (unsupervised transformation)
 - 데이터를 새롭게 표현
 - 차원 축소 (dimensionality reduction)
 - 텍스트 문서 주제 추출
 - 군집 (클러스터링, clustering)
 - 비슷한 데이터 그룹화
 - SNS 업로드 사진 그룹화



비지도 학습 알고리즘

- 도전과제

- 평가

- 알고리즘이 데이터에 유용한 학습인지 아닌지 판단 필요
 - 학습결과 직접 확인하여 평가하는 경우가 많음

- 적용

- 데이터 변환을 통해 데이터를 더 잘 이해하기 위한 분석 단계
 - 지도학습의 전처리 단계
 - 비지도 학습 결과로 새롭게 표현된 데이터를 사용해 학습



비지도 변환

- 데이터 전처리
 - StandardScaler
 - Feature의 평균을 0, 분산을 1로 변경
 - 최대/최소 값을 제한하지 않음
 - RobustScaler
 - 중간값과 사분위 값 사용
 - Outlier에 영향을 덜 받음
 - MinMaxScaler
 - 0 ~ 1 사이 정규화
 - Normalizer
 - Feature vector의 L^2_{norm} = 1로 조정
 - 위의 세가지 방식은 전체 샘플에 대한 통계 값 이용하지만
이 방식은 해당 데이터만 사용



비지도 변환

- 데이터 전처리
 - 데이터 스케일 조정 비교

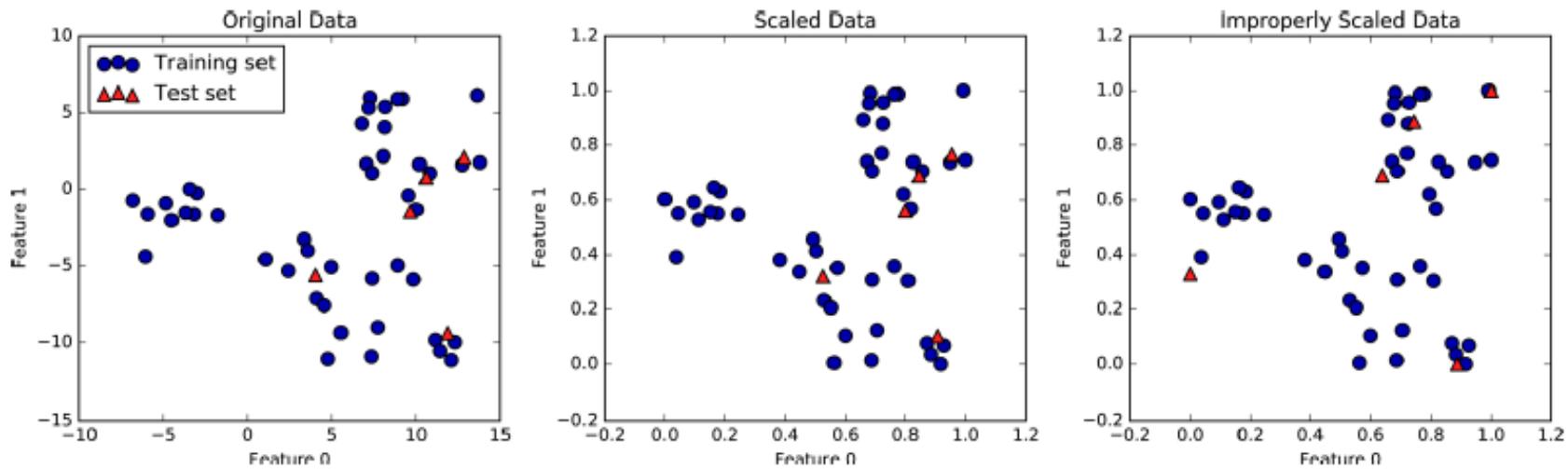


Figure 3-2. Effect of scaling training and test data shown on the left together (center) and separately (right)

비지도 변환

- 데이터 전처리
 - 실습
 - 지도학습에 적용



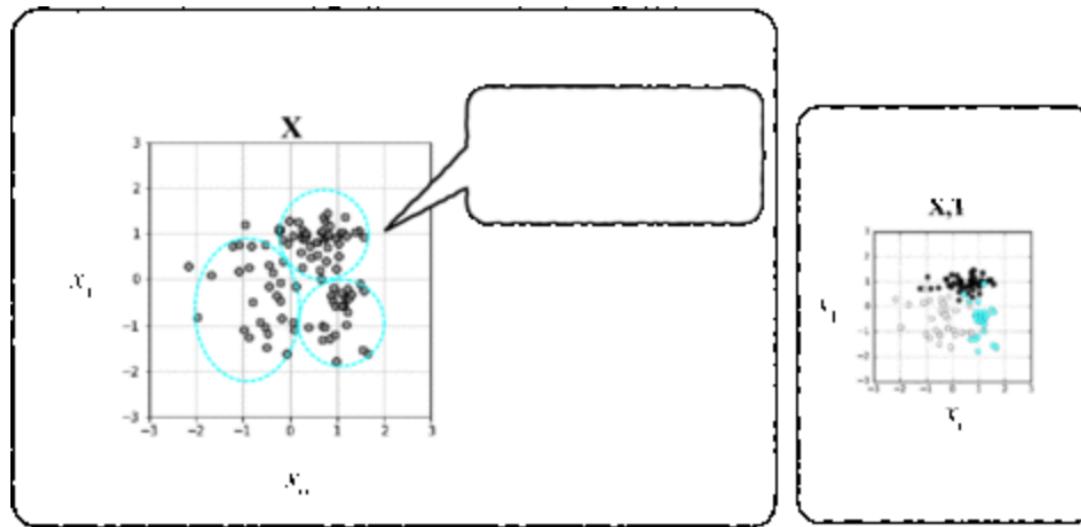
군집 (클러스터링)

- 개념

- 클래스 정보 없이 입력 데이터가 비슷한 것끼리 클래스로 나누는 작업

- 2차원 입력 데이터

- 입력 데이터 X 사용
 - 클래스 데이터 T는 사용하지 않음



군집

- 적용 예

- 고객데이터 클러스터링
 - 주부층, 직장인 층 등 유형별 판매 전략
- 곤충 데이터
 - 질량, 길이, 머리 크기 등 이용 변종 존재 가능성 체크

- 알고리즘

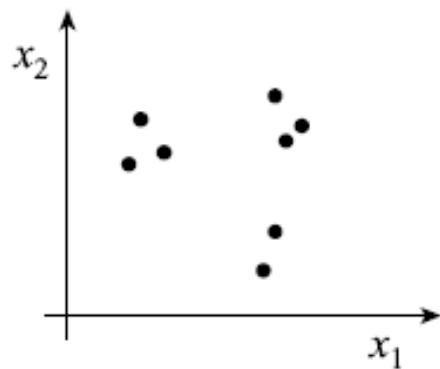
- K-means 기법
- 병합
- DBSCAN
- 가우시안 혼합 모델 (GMM, Gaussian Mixture Model)



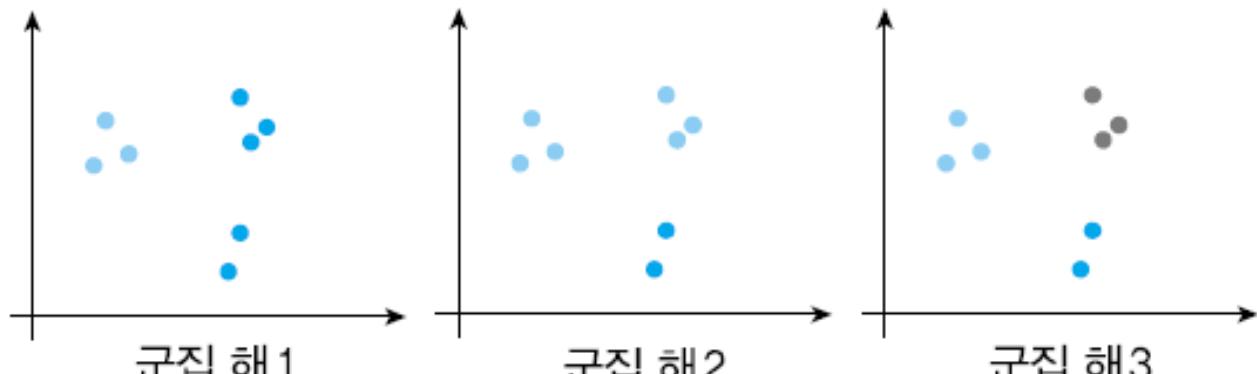
군집

■ 특성

- 주관적인 판단에 따라 결과가 달라짐
 - 클러스터링 결과의 품질은 응용이 처한 상황과 요구사항에 따라 다름



(a) 샘플 집합



(b) 세 가지 군집화 결과를 어떻게 평가할까?

군집

■ 정의

- 군집화란? 샘플 집합 $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ 이 주어졌다고 하자. \mathbf{x}_i 는 i 번째 샘플로 d 차원 특징 벡터 $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$ 로 표기한다. 이 입력에 대해 아래 조건을 만족하는 k 개의 군집으로 구성되는 군집 해 clustering solution $C = \{c_1, c_2, \dots, c_k\}$ 를 찾아라. 보통 군집의 개수 k 는 N 에 비해 매우 작다. 상황에 따라 k 가 주어지는 경우도 있고 그렇지 않고 이 값을 찾아야 하는 경우도 있다.

$$\left. \begin{array}{l} c_i \neq \emptyset, i = 1, \dots, k \\ \cup_{i=1,k} c_i = X \\ c_i \cap c_j = \emptyset, i \neq j \end{array} \right\}$$



군집

■ 거리와 유사도

군집화가 추구하는 본질적인 목표는 같은 군집 내의 샘플은 서로 가깝고 다른 군집에 속한 샘플 사이의 거리는 멀게 하는 것이다. 따라서 군집화에서 거리 개념은 매우 중요하고 여러 가지 계산 방법이 개발되어 있다. 어떤 계산 방법을 사용하느냐에 따라 군집화 결과는 달라지고 상황에 적합할 수도 있고 그렇지 않을 수도 있다. 따라서 주어진 문제에 적합한 거리 측정 방법을 선택하는 것이 매우 중요하다. 거리와 distance 유사도는 similarity 반대 개념이고 하나를 알면 다른 것은 공식을 이용하여 쉽게 계산할 수 있다.



거리와 유사도

■ Minkowski 거리

- 두 점간의 거리 척도
- $p = 1$ 면 도시 블록 거리
- $p = 2$ 면 유클리디언 거리

$$d_{ij} = \left(\sum_{k=1}^d |x_{ik} - x_{jk}|^p \right)^{1/p}$$

유클리디언 거리 ($p = 2$) $d_{ij} = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2}$

맨하탄 거리 ($p = 1$) $d_{ij} = \sum_{k=1}^d |x_{ik} - x_{jk}|$

■ Hamming 거리

- 이진 특징 벡터에 적용 가능 (서로 다른 비트의 개수)
- 예) $(1,0,1,0,0,0,1,1)^T$ 과 $(1,0,0,1,0,0,1,0)^T$ 의 해밍 거리는 ?

■ Cosine 유사도

- 문서 검색 응용에 주로 사용
- 특징: 단어, 특징 값: 출현 빈도

$$s_{ij} = \cos \theta = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$



거리와 유사도

- 점 집합을 위한 거리

- 점과 점 집합과의 거리
- 점 집합 간 거리

점 \mathbf{x}_i 와 군집 (점 집합) c_j 간의 거리를 $D(\mathbf{x}_i, c_j)$ 로 표기
두 군집 c_i 와 c_j 간의 거리는 $D(c_i, c_j)$ 로 표기

- 점과 군집 사이의 거리 (모든 샘플 참여)

$$D_{\max}(\mathbf{x}_i, c_j) = \max_{\mathbf{y}_k \in c_j} d_{ik}$$

$$D_{\min}(\mathbf{x}_i, c_j) = \min_{\mathbf{y}_k \in c_j} d_{ik}$$

$$D_{\text{ave}}(\mathbf{x}_i, c_j) = \frac{1}{|c_j|} \sum_{\mathbf{y}_k \in c_j} d_{ik}$$

- d_{ik} 는 \mathbf{x}_i 와 \mathbf{y}_k 는 간의 거리 (\mathbf{y}_k 는 c_j 에 속한 샘플)
- D_{\max} 와 D_{\min} 은 outlier에 민감



거리와 유사도

- 점 집합을 위한 거리
 - 점과 군집 사이의 거리 (대표 샘플만 참여)
 - 평균을 대표로 삼음
 - y_{mean} : 모든 군집의 평균점
 - $d_{i,mean}$: x_i 와 y_{mean} 간의 거리

$$\left. \begin{array}{l} D_{mean}(x_i, c_j) = d_{i,mean} \\ \text{여기서 } y_{mean} = \frac{1}{|c_j|} \sum_{y_k \in c_j} y_k \end{array} \right\}$$

- 다른 점과의 거리의 합이 가장 작은 점을 대표로 삼음

$$\left. \begin{array}{l} D_{rep}(x_i, c_j) = d_{i,rep} \\ \text{여기서 } \sum_{y_k \in c_j} d_{rep,k} \leq \sum_{y_k \in c_j} d_{lk}, \forall y_l \in c_j \end{array} \right\}$$



거리와 유사도

■ 예) 점과 군집 사이의 거리

$$c_j = \{\mathbf{y}_1 = (1,1)^T, \mathbf{y}_2 = (1,2)^T, \mathbf{y}_3 = (2,1)^T, \mathbf{y}_4 = (3,1)^T\}, \mathbf{x}_i = (4,2)^T$$

$$D_{\max} = \max(3.162, 3.0, 2.236, 1.414) = 3.162$$

$$D_{\min} = \min(3.162, 3.0, 2.236, 1.414) = 1.414$$

$$D_{\text{ave}} = (3.162+3.0+2.236+1.414)/4 = 2.453$$

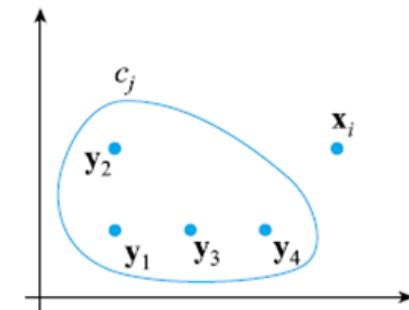


그림 10.4 군집과 점 간의 거리

$$\mathbf{y}_{\text{mean}} = ((1,1)^T + (1,2)^T + (2,1)^T + (3,1)^T)/4 = (1.75, 1.25)^T$$

$$D_{\text{mean}} = d_{i,\text{mean}} = 2.372$$

$$D_{\text{rep}}(\mathbf{x}_i, c_j) = d_{i,\text{rep}} = 2.236$$

$$\sum_{\mathbf{y}_k \in c_j} d_{1k} = 1.0 + 1.0 + 2.0 = 4.0$$

$$\sum_{\mathbf{y}_k \in c_j} d_{2k} = 1.0 + 1.414 + 2.236 = 4.65$$

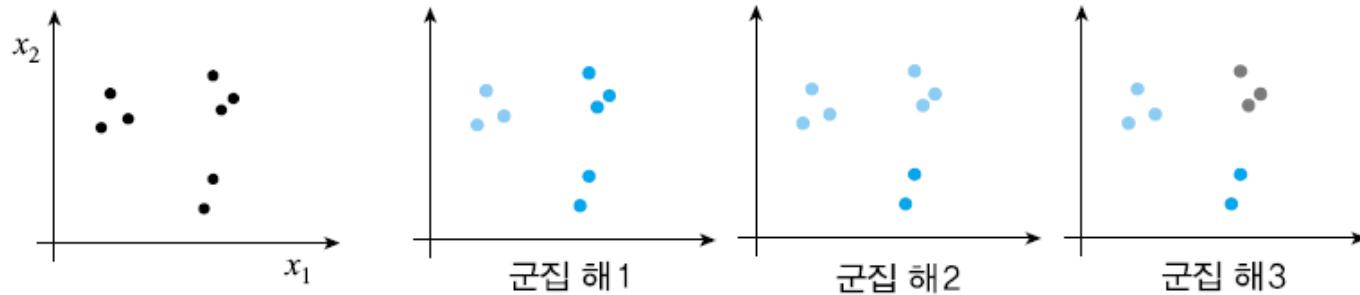
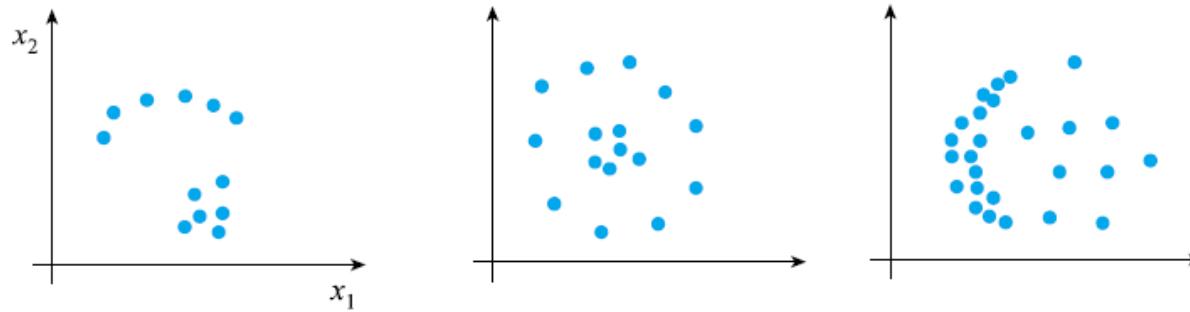
$$\sum_{\mathbf{y}_k \in c_j} d_{3k} = 1.0 + 1.414 + 1.0 = 3.414$$

(rep=3이 됨)

$$\sum_{\mathbf{y}_k \in c_j} d_{4k} = 2.0 + 2.236 + 2.0 = 6.236$$

군집화

- 다양한 알고리즘
 - 군집화 문제의 본질적인 성질에 기인
 - 주관이 많이 개입



(a) 샘플 집합

(b) 세 가지 군집화 결과를 어떻게 평가할까?

K-means 군집화

- 특성
 - 가장 많이 사용
 - 직관적 이해, 구현이 간편
 - 군집 개수를 설정해 주어야 함

- 알고리즘

알고리즘 [10.4] k-means 알고리즘

입력: 샘플 집합 $X = \{x_1, x_2, \dots, x_N\}$, 군집의 개수 k

출력: 군집 해 C

알고리즘:

1. k 개의 군집 중심 $Z = \{z_1, z_2, \dots, z_k\}$ 을 초기화 한다.
2. **while** (TRUE) {
 3. **for** ($i = 1$ **to** N) x_i 를 가장 가까운 군집 중심에 배정한다.
 4. **if** (\circ) 배정이 이전 루프의 배정과 같음) **break**;
 5. **for** ($j = 1$ **to** k) z_j 에 배정된 샘플의 평균으로 z_j 를 대치한다.
6. }



K-means 군집화

■ 예) K-means 알고리즘

7개 샘플을 $k=3$ 개의 군집으로 만드는 상황

$$\mathbf{x}_1 = (18,5)^T, \mathbf{x}_2 = (20,9)^T, \mathbf{x}_3 = (20,14)^T, \mathbf{x}_4 = (20,17)^T, \mathbf{x}_5 = (5,15)^T, \mathbf{x}_6 = (9,15)^T, \\ \mathbf{x}_7 = (6,20)^T$$

초기화에 의해 $\{\mathbf{x}_1\}$ 은 \mathbf{z}_1
(그림 10.12(a)), $\{\mathbf{x}_2\}$ 은 \mathbf{z}_2

$\{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}$ 은 \mathbf{z}_3

라인 5에 의해 $\mathbf{z}_1 = \mathbf{x}_1 = (18,5)^T$
(그림 10.12(b)), $\mathbf{z}_2 = \mathbf{x}_2 = (20,9)^T$

$$\mathbf{z}_3 = (\mathbf{x}_3 + \mathbf{x}_4 + \mathbf{x}_5 + \mathbf{x}_6 + \mathbf{x}_7)/5 = (12,16.2)^T$$

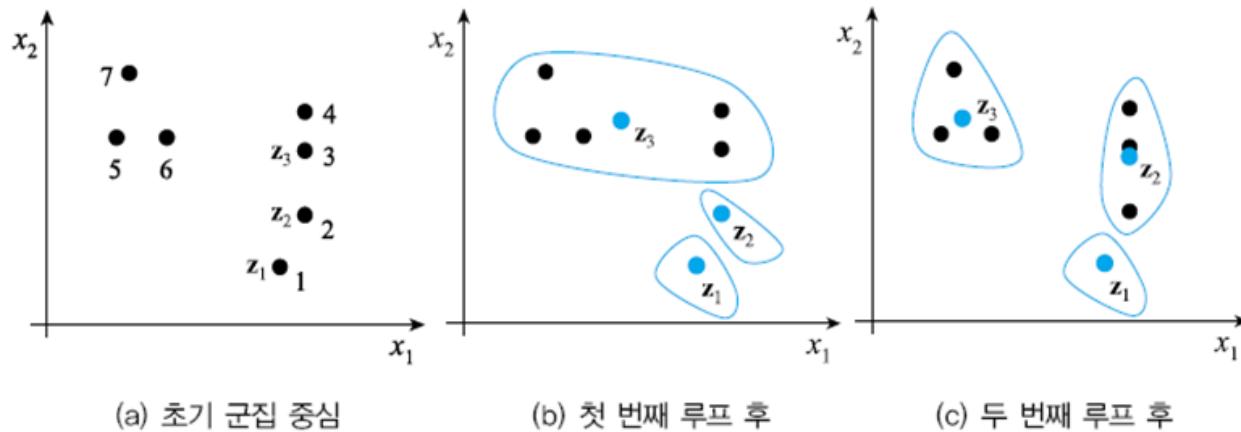


그림 10.12 k-means의 동작 예

K-means 군집화

■ 예) K-means 알고리즘

두 번째 루프를 실행하면 $\{x_1\}$ 은 z_1

(그림 10.12(c)), $\{x_2, x_3, x_4\}$ 은 z_2

$\{x_5, x_6, x_7\}$ 은 z_3

$$z_1 = x_1 = (18, 5)^T$$

$$z_2 = (x_2 + x_3 + x_4)/3 = (20, 13.333)^T$$

$$z_3 = (x_5 + x_6 + x_7)/3 = (6.667, 16.667)^T$$

세 번째 루프는 그 이전과 결과가 같다. 따라서 멈춘다.

결국 출력은

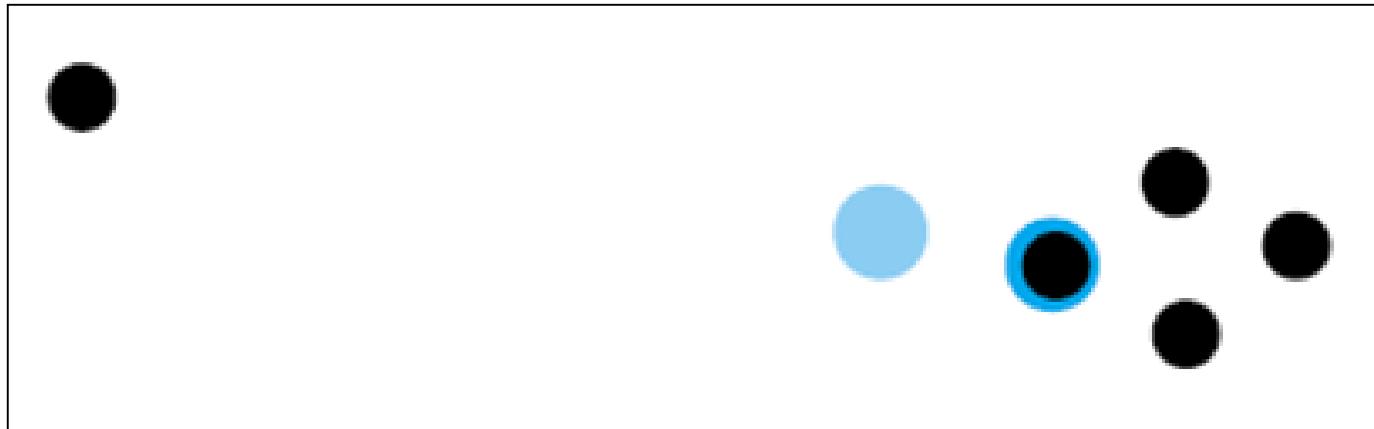
$$C = \{\{x_1\}, \{x_2, x_3, x_4\}, \{x_5, x_6, x_7\}\}$$



K-means 군집화

- 특징

- 항상 지역 최적점으로 수렴 (전역 최적점 보장 못함)
 - 다중 초기점 알고리즘
- 초기 군집 중심에 민감
- 빠르다.
- 외톨이(outlier)에 민감하다. (cf. k-medoids)



(진파랑: k-medoids, 연파랑: k-means)



K-means 군집화

- 실습



병합 군집 (Agglomerative Clustering)

- 개념
 - 시작할 때 각 포인트를 하나의 클러스터로 지정
 - 종료 조건을 만족할 때 까지 두 클러스터 병합
 - 사이킷런의 경우 클러스터 개수로 종료
- 클러스터 측정 방법
 - ward (default)
 - 모든 클러스터 내 분산을 가장 작게 증가시키는 두 클러스터 병합
 - 크기가 비교적 비슷한 클러스터 생성
 - average
 - 평균 거리가 가장 짧은 두 클러스터 병합
 - complete
 - 최대 거리가 가장 짧은 두 클러스터 병합



병합 군집

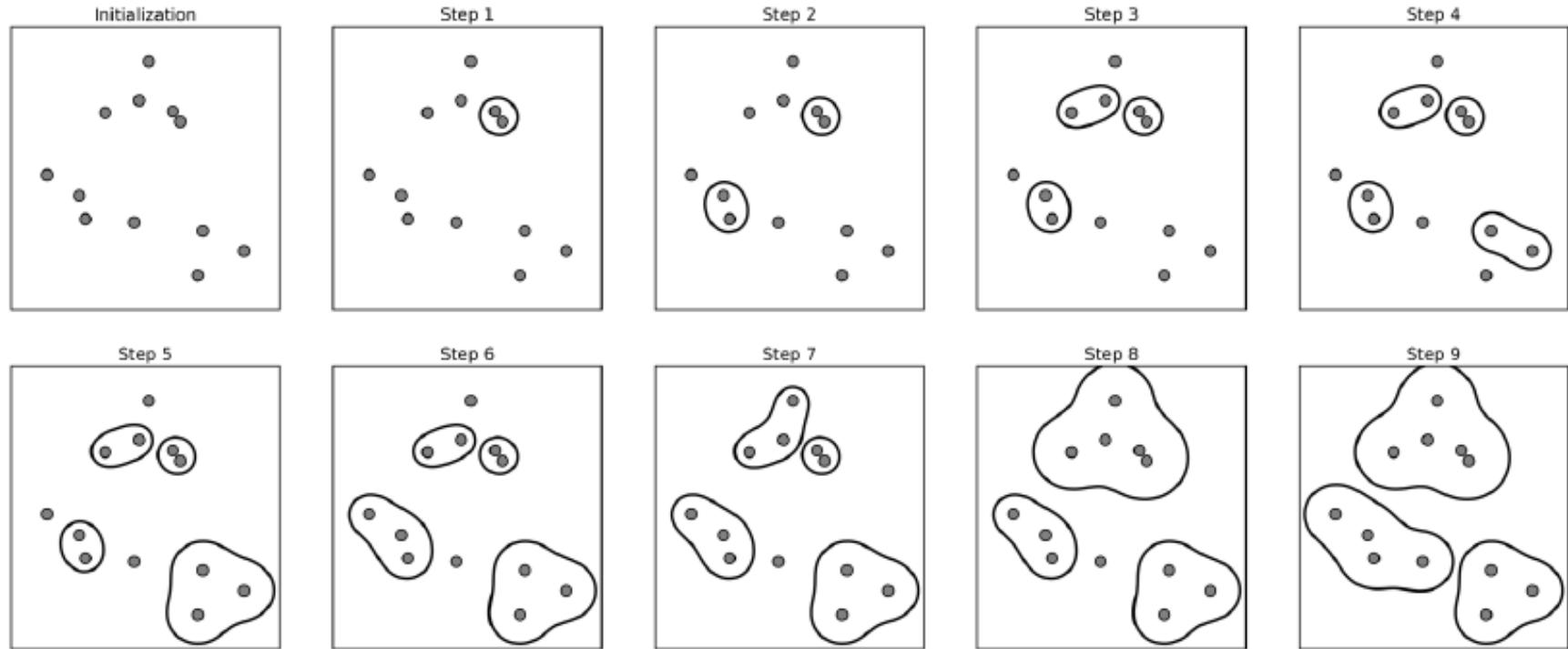


Figure 3-33. Agglomerative clustering iteratively joins the two closest clusters

병합 군집

- 실습



DBSCAN

■ 개념

- Density-Based Spatial Clustering of Applications with Noise
- 클러스터의 개수를 미리 지정할 필요 없음
- 큰 데이터 셋에 적용 가능
- 가까이 있는 데이터가 많아 불비는 지역(dense region) 검색 및 군집화
- 핵심 샘플 (dense region에 있는 포인트) 정의
 - eps 와 min_samples 로 정의
 - eps안에 데이터가 min_samples만큼 들어 있을 때
 - eps보다 가까운 핵심샘플은 동일한 클러스터로 병합



DBSCAN

■ 알고리즘

- 무작위로 포인트 선택
- 포인트에서 eps 거리 안의 모든 포인트 검색
- eps 안에 있는 포인트 수가 min_samples 보다 적으면
 - 잡음(어떤 클래스에도 속하지 않음) 레이블링
- eps 안에 있는 포인트 수가 min_samples 보다 많다면
 - 핵심샘플로 레이블링
 - 새로운 클러스터 레이블 할당
 - 현재 포인트의 eps 거리 안의 모든 이웃 살핌
 - 클러스터에 할당이 안되었다면 바로 전 클러스터 레이블링
 - 핵심샘플이면 포인트의 이웃을 차례로 방문
 - eps 안에 더 이상 핵심 샘플이 없을 때 까지 작업 진행
 - 아직 방문하지 않은 포인트를 선택하고 같은 과정 반복



DBSCAN

- 특징
 - eps가 클수록 클러스터에 속하는 샘플은?
 - 클러스터를 커지게 함
 - 여러 클러스터를 하나로 합치게 만듦
 - eps가 너무 크면
 - 모든 포인트가 단 하나의 클러스터에 속하게 됨
 - eps가 너무 작으면
 - 핵심 샘플수가 줄어들고 대부분이 잡음 포인트가 됨
 - min_samples가 클수록 핵심 샘플 수는 줄어듦
 - 잡음 포인트는 늘어남
- 실습



참고 자료

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

