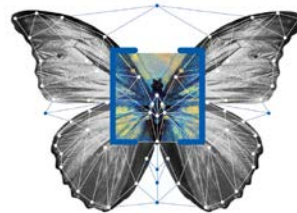


1장. 기계학습 소개

실전코딩

한빛아카데미
HANBIT ACADEMY INC.



MACHINE 기계 학습
LEARNING
오일석 지음

본 강의자료는 한빛아카데미에서 제공하는 강의자료를 바탕으로 작성되었음



PREVIEW

■ 사람의 학습

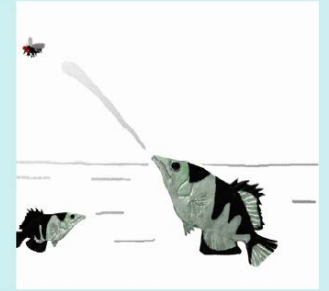
- 수학, 과학, 역사뿐 아니라 수영, 자전거 타기 등

■ 동물의 학습

- 예) 물총물고기의 목표물 맞추기 능력 향상



(a) 자전거 타기 학습



(b) 물총물고기의 사냥

그림 1-1 사람과 동물의 학습

■ 기계 학습

- 그렇다면 기계도 학습할 수 있을까?
- 경험을 통해 점점 성능이 좋아지는 기계를 만들 수 있을까?
- 이 책은 이 질문에 대한 답을 찾아가는 길



각 절에서 다루는 내용

- 1.1절: 기계 학습의 정의와 개념, 인공지능을 구현하는 도구로서의 역할을 설명한다.
- 1.2절: 특징 공간과 공간 변환을 소개한다.
- 1.3절: 데이터의 중요성과 희소성을 강조한다.
- 1.4절: 선형 회귀를 이용하여 기계 학습을 직관적으로 설명한다.
- 1.5절: 모델 선택의 중요성과 방법, 과소적합과 과잉적합을 설명한다.
- 1.6절: 현대 기계 학습에서 매우 중요한 규제 기법으로 데이터 확대와 가중치 감소를 간략히 기술한다.
- 1.7절: 기계 학습의 유형으로 지도 학습, 비지도 학습, 강화 학습, 준지도 학습을 소개한다.
- 1.8절: 기계 학습의 간략한 역사와 인공지능의 사회적 의미를 살펴본다.



1.1 기계 학습이란

- 1.1.1 기계 학습의 정의
- 1.1.2 지식기반 방식에서 기계 학습으로의 대전환
- 1.1.3 기계 학습 개념
- 1.1.4 사람의 학습과 기계 학습



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
 - $80932.46789076 \times 0.39001324$ 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 홀쭉하며, 맨 위와 아래가 둥근 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

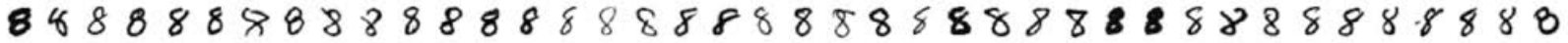


그림 1-2 인식 시스템이 대처해야 하는 심한 변화 양상(8과 단추라는 패턴을 어떻게 기술할 것인가?)

- 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

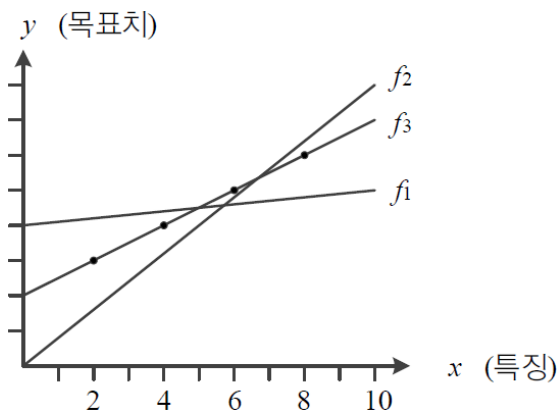


그림 1-4 간단한 기계 학습 예제

■ 예측prediction 문제

- 임의의 시간이 주어지면 이때 이동체의 위치는?
- 회귀regression 문제와 분류classification 문제로 나뉨
 - 회귀는 목표치가 실수, 분류는 부류값(class label) ([그림 1-4]는 회귀 문제)



1.1.3 기계 학습 개념

■ 훈련집합

- 가로축은 **특징**, 세로축은 **목표치**
- 관측한 4개의 점이 **훈련집합**을 구성함

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

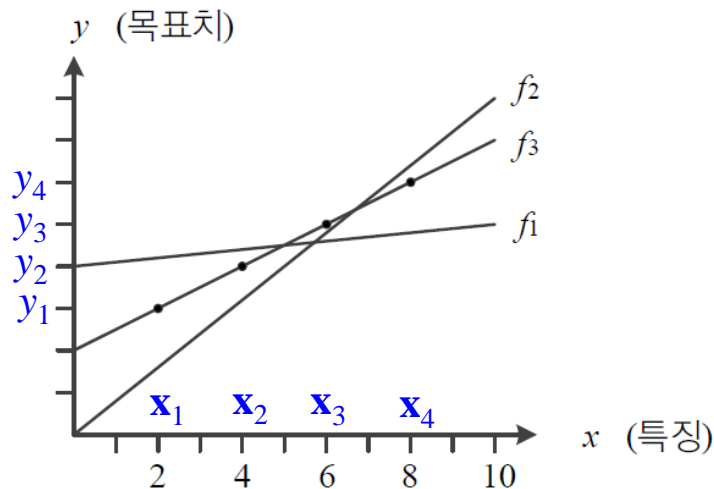


그림 1-4 간단한 기계 학습 예제

[그림 1-4] 예제의 훈련집합

$$\mathbb{X} = \{\mathbf{x}_1 = (2.0), \mathbf{x}_2 = (4.0), \mathbf{x}_3 = (6.0), \mathbf{x}_4 = (8.0)\}$$

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



1.1.3 기계 학습 개념

■ 데이터를 어떻게 모델링 할 것인가

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

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

■ 기계 학습은

- 가장 정확하게 예측할 수 있는, 즉 최적의 매개변수(=파라미터)를 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 $b=2.0$

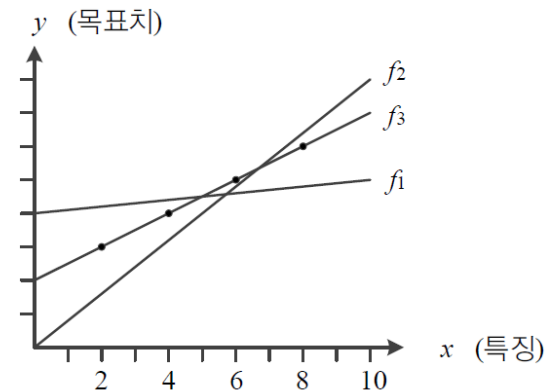


그림 1-4 간단한 기계 학습 예제



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

- 딥러닝(신경망 기반)
 - 동시에 학습 가능한 과업 수: 딥러닝에서는 서로 다른 여러 neural network를 동시에 학습시킬 수 있음. (ex. GAN(Generative Adversarial Network))
 - 학습원리에 대한 지식: black-box 모델. 모든 과정이 밝혀지지 않는. 다만 주어진 objective function에 대해서 learning을 통해 목적을 달성하는 것은 확인 가능
- 역사: less than 20 years



1.2 특징 공간에 대한 이해

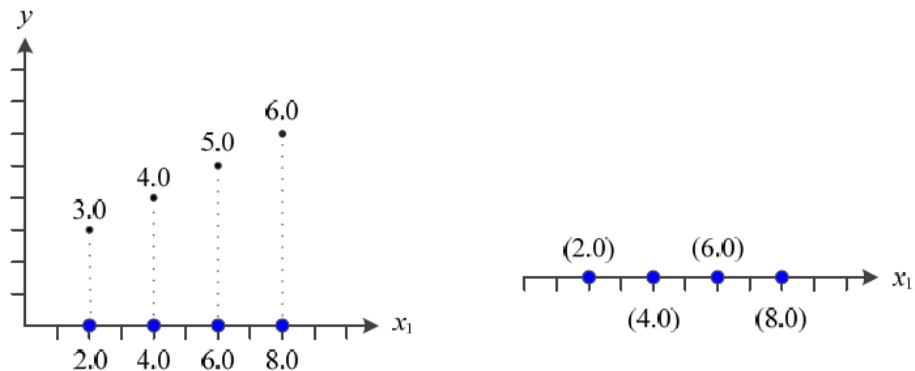
- 1.2.1 1차원과 2차원 특징 공간
- 1.2.2 다차원 특징 공간
- 1.2.3 특징 공간 변환과 표현 학습

특징 공간: 관측값들이 있는 공간 (여러 차원으로 구성될 수 있음)



1.2.1 1차원과 2차원 특징 공간

■ 1차원 특징 공간 →



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

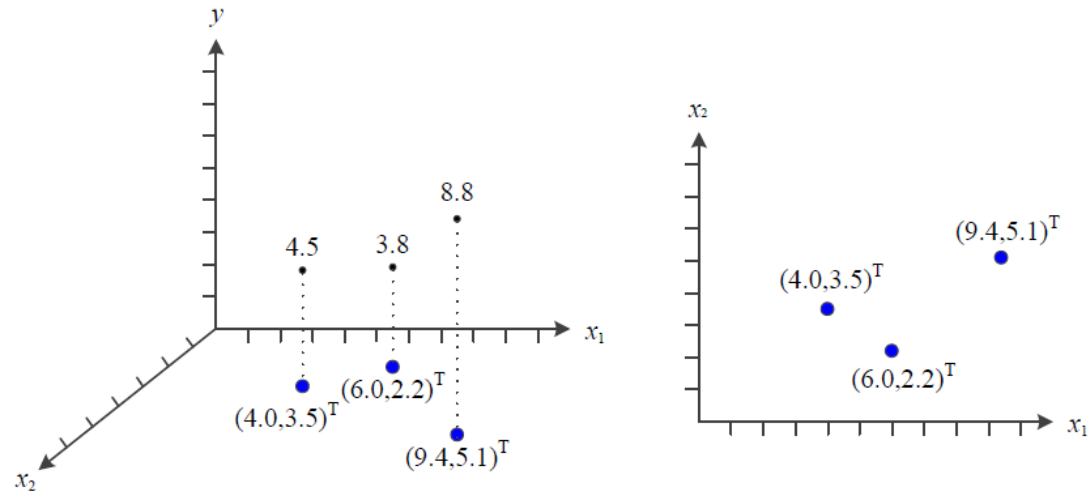
■ 2차원 특징 공간 →

■ 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 목표값을 축으로 표시, 오른쪽: 특징 벡터만 축으로 표시)

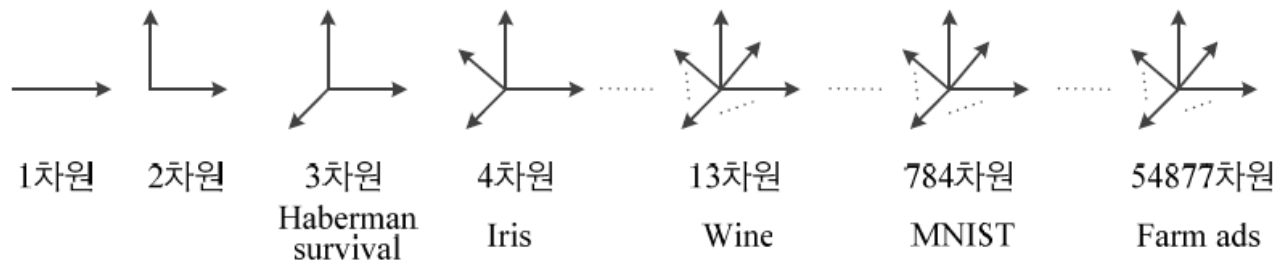
그림 1-5 특징 공간과 데이터의 표현

벡터(vector): 크기와 방향을 동시에 가지는 물리적 양
특징공간(feature space)는 vector로 표현될 수 있음 (각 feature가 vector의 성분이 됨)
특징벡터(feature vector)는 attribute vector 라고도 함



1.2.2 다차원 특징 공간

■ 다차원 특징 공간 예제



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

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

Wine: $\mathbf{x} = (\text{Alcohol}, \text{Malic acid}, \text{Ash}, \text{Alcalinity of ash}, \text{Magnesium}, \text{Total phenols}, \text{Flavanoids}, \text{Nonflavanoid phenols}, \text{Proanthocyanins}, \text{Color intensity}, \text{Hue}, \text{OD280 / OD315 of diluted wines}, \text{Proline})^T$

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

Farm ads: $\mathbf{x} = (\text{단어1}, \text{단어2}, \dots, \text{단어54877})^T$

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= $d+1$

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

- 2차 곡선 모델을 사용하면 매개변수 수가 크게 증가

- 매개변수 수= d^2+d+1

- 1-차원 데이터 \rightarrow 2차 곡선 $\rightarrow y=w_1x_1^2+w_2x_1+b$

- 예) Iris 데이터: $d=4$ 이므로 21개의 매개변수

- 예) MNIST 데이터: $d=784$ 이므로 615,441개의 매개변수

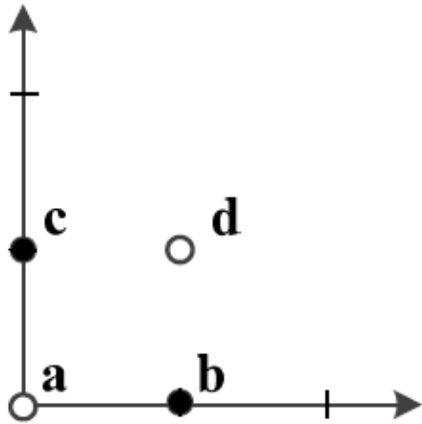
$$y = \underline{w_1}x_1^2 + \underline{w_2}x_2^2 + \dots + \underline{w_d}x_d^2 + \underline{w_{d+1}}x_1x_2 + \dots + \underline{w_{d^2}}x_{d-1}x_d + \underline{w_{d^2+1}}x_1 \\ + \dots + \underline{w_{d^2+d}}x_d + \underline{b} \quad (1.5)$$



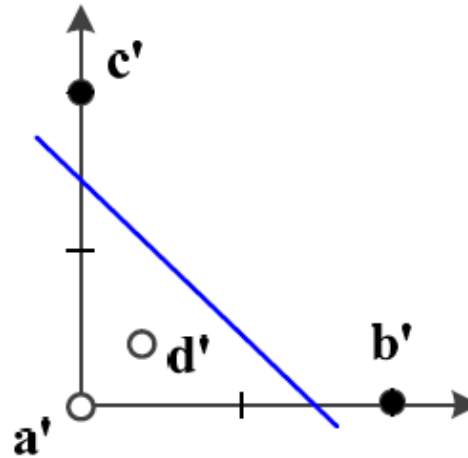
1.2.3 특징 공간 변환과 표현 학습

■ 선형 분리 불가능(Linearly non-separable)한 원래 특징 공간 ([그림 1-7(a)])

- 아래 그림 (a)에서는 어떤 직선 모델을 적용하더라도 75% 정확률이 한계 (4중 1개는 분류 못함)



(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)])

- 특징 공간 변환을 통해 직선 모델로 100% 정확률 달성

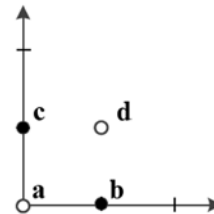
$$\text{원래 특징 벡터 } \mathbf{x} = (x_1, x_2)^T \rightarrow \text{변환된 특징 벡터 } \mathbf{x}' = \left(\frac{x_1}{2x_1x_2 + 0.5}, \frac{x_2}{2x_1x_2 + 0.5} \right)^T \quad (1.6)$$

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

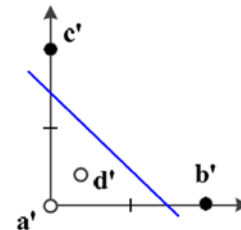
$$\mathbf{b} = (1,0)^T \rightarrow \mathbf{b}' = (2,0)^T$$

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

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



(a) 원래 특징 공간



(b) 분류에 더 유리하도록 변환된 새로운 특징 공간

■ 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 수행한 셈



1.2.3 특징 공간 변환과 표현 학습

■ 차원에 대한 몇 가지 설명

- 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

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

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

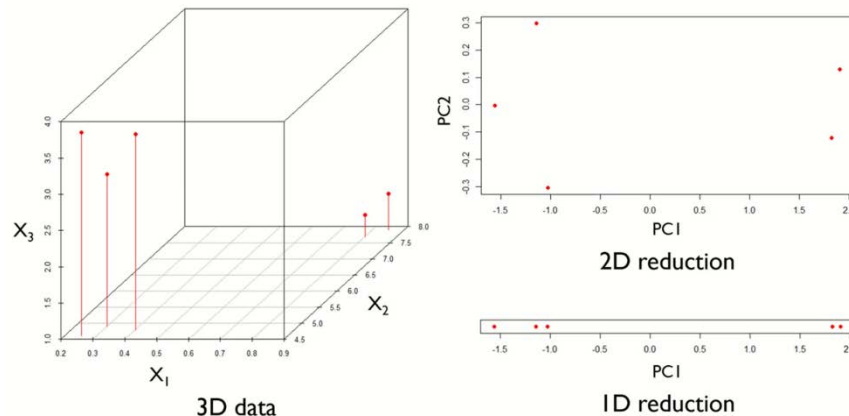


1.2.3 특징 공간 변환과 표현 학습

■ 차원의 저주 (Curse of Dimensionality)

- 차원이 높아짐에 따라 발생하는 현실적인 문제들
- 차원이 증가할 수록 개별 차원 내 학습할 데이터 수가 적어지는(sparse) 현상 발생
 - 예) $d=4$ 인 Iris 데이터에서 축마다 100개 구간으로 나누면 총 $100^4=1$ 억 개의 칸
 - 예) $d=784$ 인 MNIST 샘플의 화소가 0과 1값을 가진다면 2^{784} 개의 칸. 이 거대한 공간에 고작 6만 개의 샘플을 흩뿌린 매우 희소한 분포

Reduce data from 3D to 2D or 1D



- 즉, 가지고 있는 데이터(샘플)가 고차원의 feature space에 고르게 분포할 수 없으며, 이렇게 비어있는 공간은 정보가 없으니 머신러닝 모델을 만들 때 이런 공간을 고려하기 어렵고 결국 성능이 감소될 수 있음



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

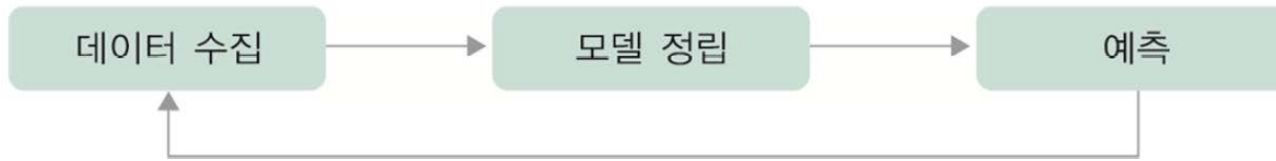


그림 1-8 과학기술의 발전 과정

- 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 붓꽃(*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 목꽃값은 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



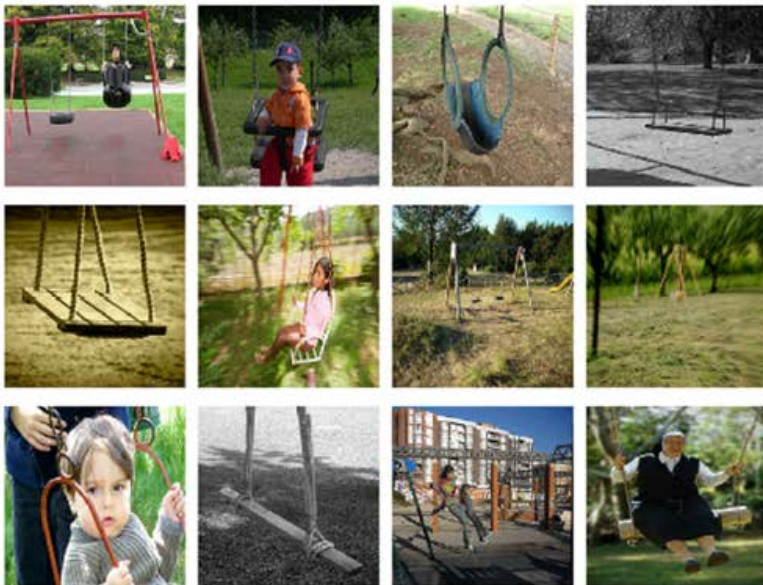
1.3.2 데이터베이스의 중요성

- 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



1.3.3 데이터베이스 크기와 기계 학습 성능

■ 데이터베이스의 왜소한 크기

- 예) MNIST: 28*28 흑백 비트맵이라면 서로 다른 총 샘플 수는 2^{784} 가지이지만, MNIST는 고작 6만 개 샘플

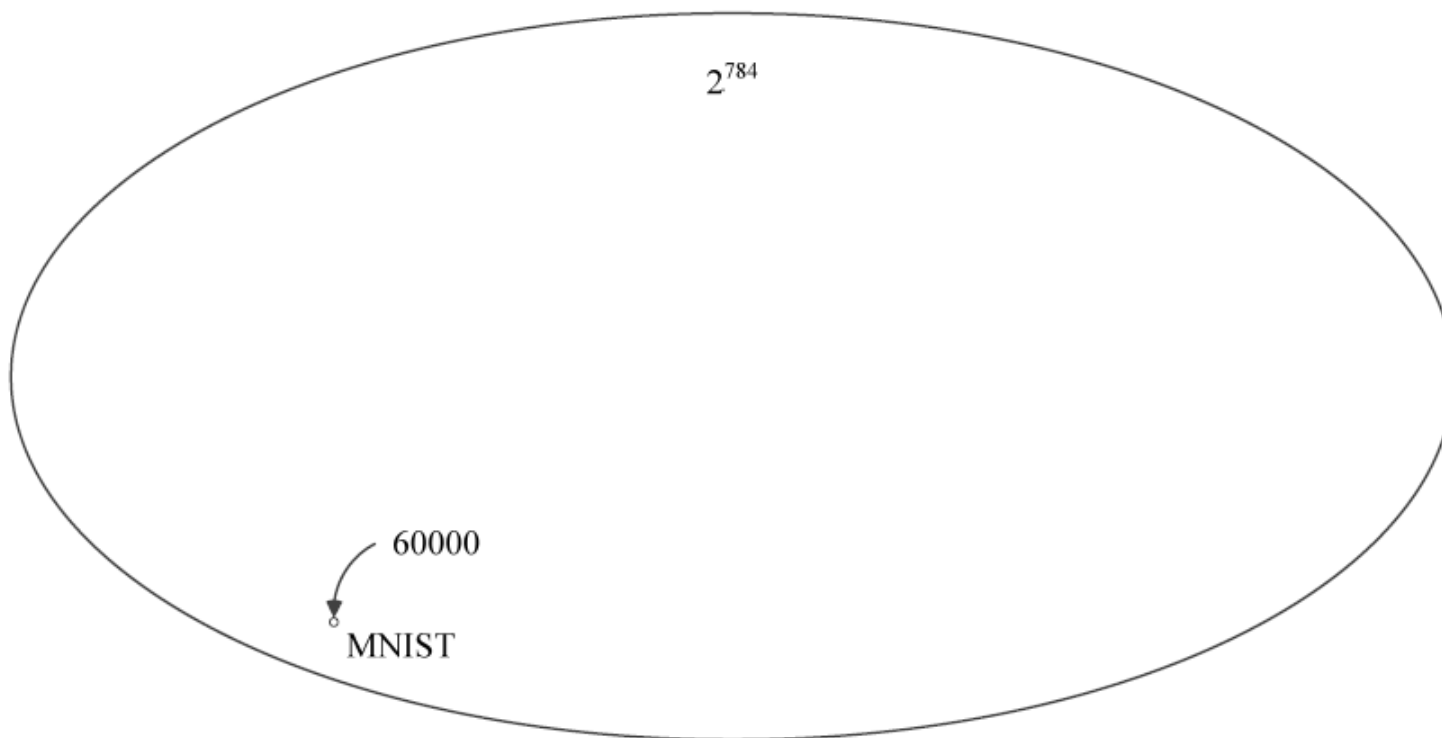


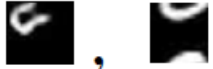
그림 1-9 방대한 특징 공간과 희소한 데이터베이스



1.3.3 데이터베이스 크기와 기계 학습 성능

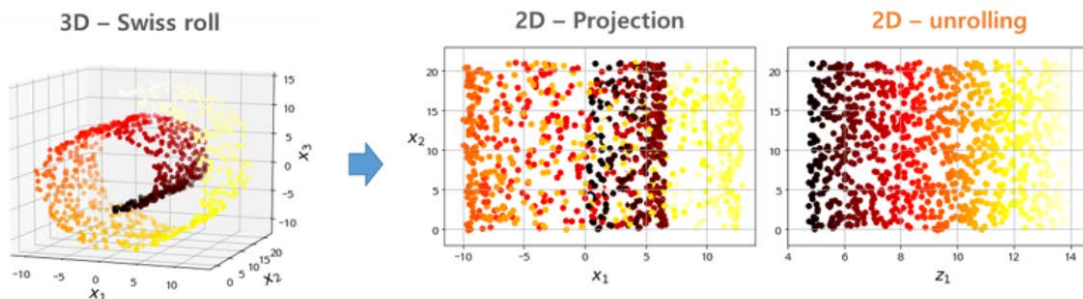
■ 왜소한 데이터베이스로 어떻게 높은 성능을 달성하는가?

- 방대한 공간에서 **실제 데이터가 발생하는 곳은 매우 작은 부분 공간임**

-  와 같은 샘플의 발생 확률은 거의 0

- 매니폴드 가정(manifold assumption) (추후 비지도학습에서 다룰 예정)

- 매니폴드란? 고차원의 데이터를 공간에 뿌릴 때 샘플들을 잘 아우를 수 있는 subspace 를 의미한다.
- 매니폴드 학습(manifold learning)을 하게 되면, 고차원의 데이터를 잘 표현할 수 있는 manifold의 특성을 알게 되고, 이를 통해 샘플 데이터의 특징을 파악할 수 있고, 차원을 축소해볼 수 있음 (→ 즉 예측 모델의 성능 증대 기대)



1.3.4 데이터 가시화

■ 4차원 이상의 초공간은 한꺼번에 가시화 불가능

■ 여러 가지 가시화 기법

- 2개씩 조합하여 여러 개의 그래프 그림

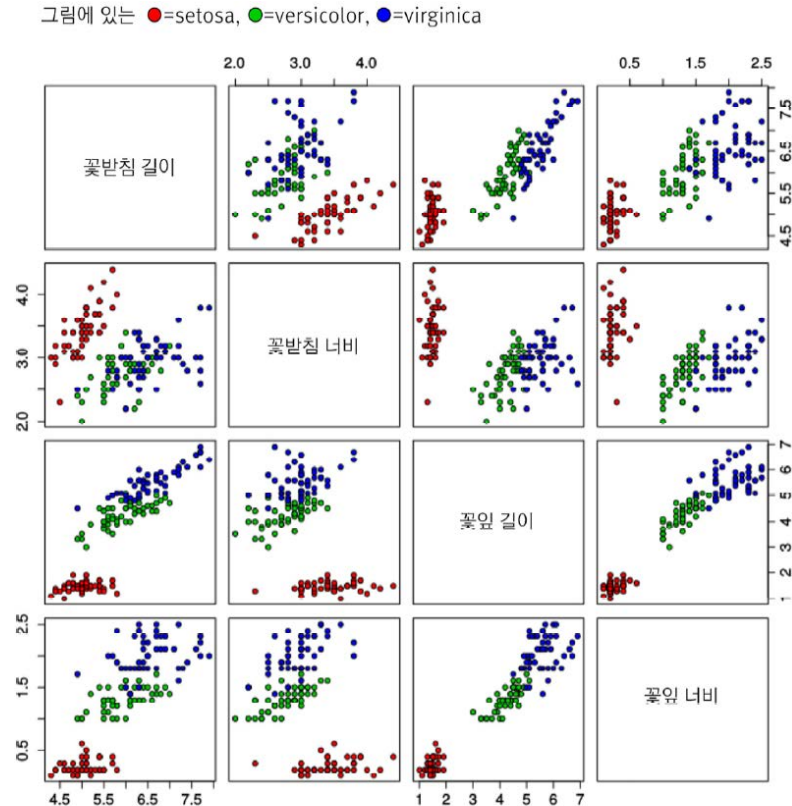


그림 1-10 고차원 특징 공간의 가시화

- 고차원 공간을 저차원으로 변환하는 기법들(6.6.1절)



1.4 간단한 기계 학습의 예

■ 선형 회귀 문제

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

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

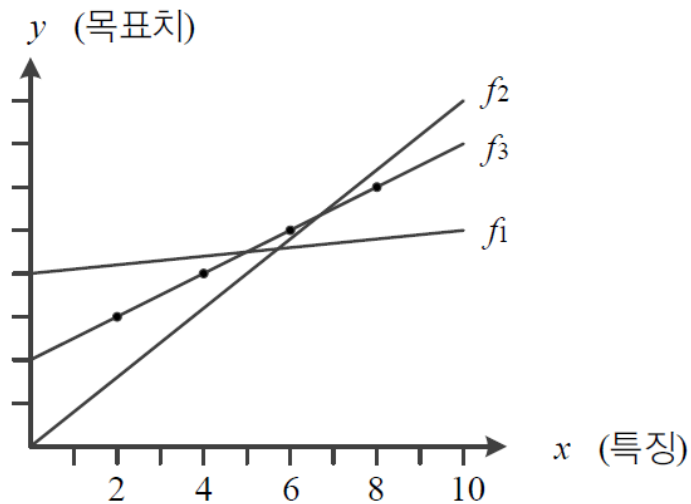


그림 1-4 간단한 기계 학습 예제



1.4 간단한 기계 학습의 예

■ 목적 함수objective function (또는 비용 함수cost 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

$$\mathbf{x}_1, y_1 \rightarrow (f_{\theta_1}(2.0) - 3.0)^2 = ((0.1 * 2.0 + 4.0) - 3.0)^2 = 1.44$$

$$\mathbf{x}_2, y_2 \rightarrow (f_{\theta_1}(4.0) - 4.0)^2 = ((0.1 * 4.0 + 4.0) - 4.0)^2 = 0.16$$

$$\mathbf{x}_3, y_3 \rightarrow (f_{\theta_1}(6.0) - 5.0)^2 = ((0.1 * 6.0 + 4.0) - 5.0)^2 = 0.16$$

$$\mathbf{x}_4, y_4 \rightarrow (f_{\theta_1}(8.0) - 6.0)^2 = ((0.1 * 8.0 + 4.0) - 6.0)^2 = 1.44$$

$$\longrightarrow J(\theta_1) = 0.8$$



1.4 간단한 기계 학습의 예

■ [예제 1-1] 훈련집합

- θ_1 을 개선하여 $\theta_2 = (0.8, 0.0)^T$ 가 되었다고 가정

$$x_1, y_1 \rightarrow (f_{\theta_2}(2.0) - 3.0)^2 = ((0.8 * 2.0 + 0.0) - 3.0)^2 = 1.96$$

$$x_2, y_2 \rightarrow (f_{\theta_2}(4.0) - 4.0)^2 = ((0.8 * 4.0 + 0.0) - 4.0)^2 = 0.64$$

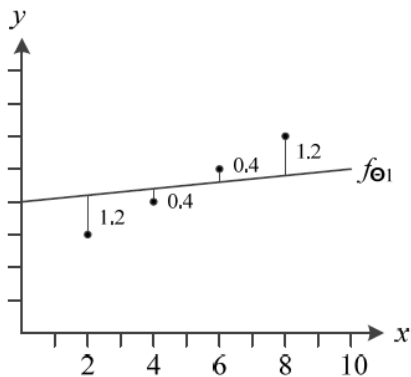
$$x_3, y_3 \rightarrow (f_{\theta_2}(6.0) - 5.0)^2 = ((0.8 * 6.0 + 0.0) - 5.0)^2 = 0.04$$

$$x_4, y_4 \rightarrow (f_{\theta_2}(8.0) - 6.0)^2 = ((0.8 * 8.0 + 0.0) - 6.0)^2 = 0.16$$

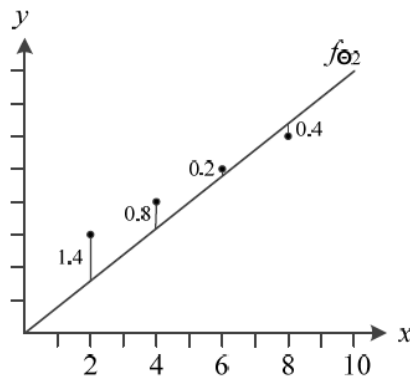
$$\longrightarrow J(\theta_2) = 0.7$$

- θ_2 를 개선하여 $\theta_3 = (0.5, 2.0)^T$ 가 되었다고 가정

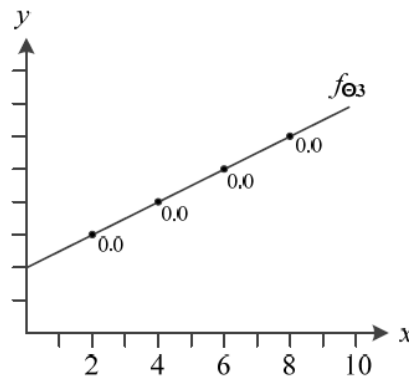
- 이때 $J(\theta_3) = 0.0$ 이 되어 θ_3 은 최적값 $\hat{\theta}$ 이 됨



(a) 초기 매개변수 θ_1



(b) θ_1 을 개선하여 θ_2 가 됨



(c) θ_2 를 개선하여 최적의 θ_3 을 찾음



1.4 간단한 기계 학습의 예

- 기계 학습이 할 일을 공식화하면,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta) \quad (1.9)$$

- 기계 학습은 작은 개선을 반복하여 최적해를 찾아가는 **수치적 방법**으로 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  $\theta_1$ 을 설정한다.
2   $t=1$ 
3  while ( $J(\theta_t)$ 가 0.0에 충분히 가깝지 않음)    // 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   $\hat{\theta} = \theta_t$ 
```



간단한 기계학습 파이썬 코드 예

```
import tensorflow as tf
tf.enable_eager_execution()
```

```
x_data = [1, 2, 3, 4, 5]
y_data = [1, 2, 3, 4, 5]
```

```
W = tf.Variable(2.9)
b = tf.Variable(0.5)
```

W, b initialize

```
learning_rate = 0.01
```

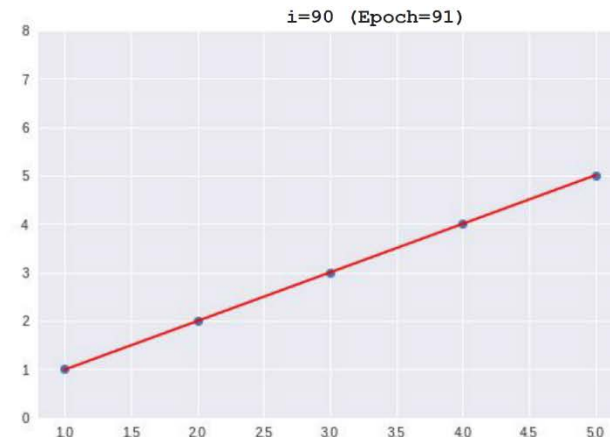
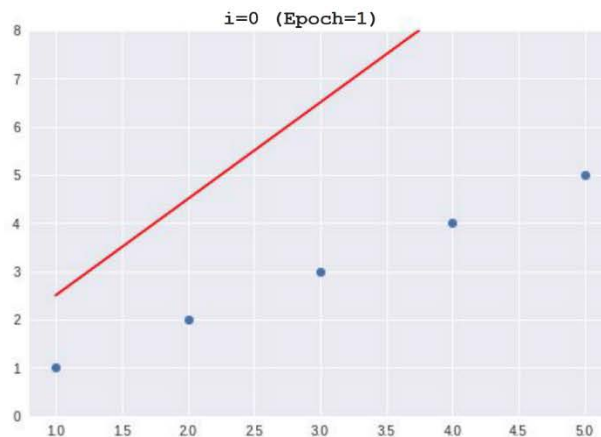
```
for i in range(100+1):
```

```
    with tf.GradientTape() as tape:
        hypothesis = W * x_data + b
        cost = tf.reduce_mean(tf.square(hypothesis - y_data))
    W_grad, b_grad = tape.gradient(cost, [W, b])
    W.assign_sub(learning_rate * W_grad)
    b.assign_sub(learning_rate * b_grad)
```

Gradient descent

```
    if i % 10 == 0:
        print("{:5}|{:10.4f}|{:10.4f}|{:10.6f}".format(i, W.numpy(), b.numpy(), cost))
```

i	W	b	cost
0	2.4520	0.3760	45.660004
10	1.1036	0.0034	0.206336
20	1.0128	-0.0209	0.001026
30	1.0065	-0.0218	0.000093
40	1.0059	-0.0212	0.000083
50	1.0057	-0.0205	0.000077
60	1.0055	-0.0198	0.000072
70	1.0053	-0.0192	0.000067
80	1.0051	-0.0185	0.000063
90	1.0050	-0.0179	0.000059



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

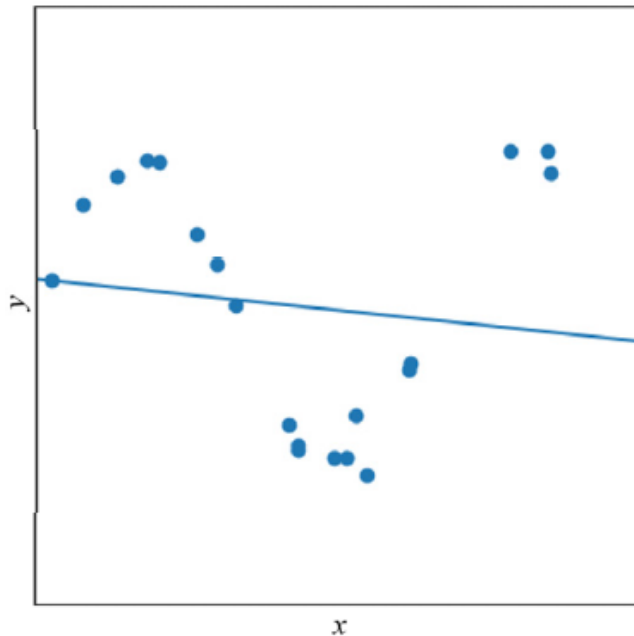


그림 1-12 선형 모델의 한계



-
- 나머지는 다음 시간에

