

기계 학습 기초

Preview

- 사람은 끊임없이 주위 환경을 인식
 - 타인이 말한 소리, 얼굴, 감정 등을 인식
 - 생존과 자기 발전에 필수
- 주위 환경을 인식하는 인공지능
 - 자율 주행차, 음성인식 챗봇, 주문 받는 로봇 등
 - 이 장에서는 인식 프로그램을 만드는데 필요한 기계 학습의 기초 지식을 공부
- 인공지능, 기계 학습, 신경망, 딥러닝의 관계

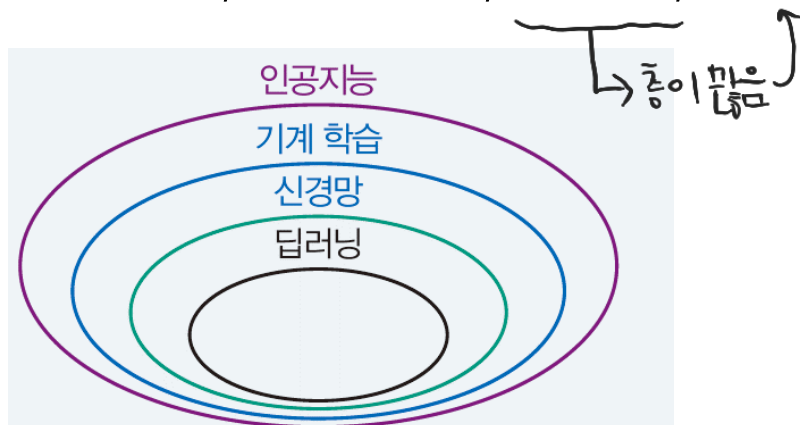


그림 3-1 인공지능, 기계 학습, 신경망, 딥러닝의 관계

3.1 기계학습 기초

- 기계학습에서 데이터의 중요성
 - 에너지를 만드는 연료에 해당
 - 데이터가 없으면 기계학습 적용이 불가능
- 가장 단순한 iris 데이터로 시작해보자.

3.1.1 데이터셋 읽기

■ 사이킷런(scikit-learn) 라이브러리 설치

- pip install scikit-learn 명령어로 라이브러리 설치

TIP https://scikit-learn.org/dev/_downloads/scikit-learn-docs.pdf에 접속하면 가장 최신 버전의 사이킷 런 사용 설명서를 무료로 다운로드할 수 있다. 무려 2,500여 쪽에 달하는 방대한 문서다. 그렇다고 겁먹을 필요는 없다. 필요한 부분을 선택적으로 참조하면 된다.

→ 딥러닝을 제외한
기계 학습 알고리즘 보충

3.1.1 iris 데이터셋 읽기

■ [프로그램 3-1(a)]: iris 데이터셋 읽기

프로그램 3-1(a) iris 데이터셋 읽기

```
01 from sklearn import datasets
02
03 d=datasets.load_iris()    # iris 데이터셋을 읽고
04 print(d.DESCR)           # 내용을 출력
```

- 01행: sklearn 모듈의 datasets 클래스를 불러옴
- 03행: load_iris 함수를 호출해 iris 데이터셋을 읽어 객체 d에 저장
- 04행: 객체 d의 DESCR 변수를 출력

■ 기계 학습의 용어

- 샘플로 구성되는 데이터셋
- 특징으로 구성되는 특징 벡터(feature vector) → 입력값
- 부류(class)

TIP 기계 학습이 사용하는 데이터는 여러 개의 샘플을 담고 있어서 데이터셋(data set)이라 부르기도 한다. 이 책에서는 데이터와 데이터셋을 엄밀히 구분하지 않고 함께 사용하는데, 데이터셋은 iris처럼 특정한 데이터를 가리킬 때 주로 사용한다.

3.1.1 iris 데이터셋 읽기

Iris plants dataset

****Data Set Characteristics:****

- 150개의 샘플
- :Number of Instances: 150 (50 in each of three classes)
- :Number of Attributes: 4 numeric, predictive attributes and the class
- :Attribute Information:
 - sepal length in cm
 - sepal width in cm
 - petal length in cm
 - petal width in cm
 - class:
 - Iris-Setosa
 - Iris-Versicolour
 - Iris-Virginica

네 개의 특징(feature) | 데이터

세 개의 부류

:Summary Statistics:

Standard Deviation 분포도

	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

:Missing Attribute Values: None

:Class Distribution: 33.3% for each of 3 classes.

:Creator: R.A. Fisher

:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

:Date: July, 1988

...

그림 3-2 iris의 세 가지 품종(왼쪽부터 Setosa, Versicolor, Virginica)



3.1.1 iris 데이터셋 읽기

■ iris의 내용 살펴보기

프로그램 3-1(b) iris의 내용 살펴보기

```
05 for i in range(0, len(d.data)):      # 샘플을 순서대로 출력
06     print(i+1, d.data[i], d.target[i])
```

```
1 [5.1 3.5 1.4 0.2] 0
2 [4.9 3. 1.4 0.2] 0
3 [4.7 3.2 1.3 0.2] 0
4 [4.6 3.1 1.5 0.2] 0
...
51 [7. 3.2 4.7 1.4] 1
52 [6.4 3.2 4.5 1.5] 1
53 [6.9 3.1 4.9 1.5] 1
54 [5.5 2.3 4. 1.3] 1
...
101 [6.3 3.3 6. 2.5] 2
102 [5.8 2.7 5.1 1.9] 2
103 [7.1 3. 5.9 2.1] 2
104 [6.3 2.9 5.6 1.8] 2
...
```

d.data(특징 벡터)

d.target(레이블)

3.1.1 기계 학습에서 데이터셋의 표현

■ 샘플을 특징 벡터와 레이블로 표현

- 특징 벡터는 \mathbf{x} 로 표기(d 는 특징의 개수로서 특징 벡터의 차원이라 부름)

$$\text{특징 벡터: } \mathbf{x} = (x_1, x_2, \dots, x_d) \quad (3.1)$$

- 레이블은 $0, 1, 2, \dots, c-1$ 의 값 또는 $1, 2, \dots, c-1, c$ 의 값 또는 원핫 코드
 - 원핫 코드는 한 요소만 1인 이진열
 - 예) Setosa는 (1,0,0), Versicolor는 (0,1,0), Virginica는 (0,0,1)로 표현

	특징 벡터 $\mathbf{x} = (x_1, x_2, \dots, x_d)$	레이블(참값) y	→ 상대적으로 크기 비교는 의미 없음
샘플 1:	(5.1, 3.5, 1.4, 0.2)	0	— iris 데이터셋 ($n=150, d=4$)
샘플 2:	(4.9, 3.0, 1.4, 0.2)	0	
...	
샘플 51:	(7.0, 3.2, 4.7, 1.4)	1	
샘플 52:	(6.4, 3.2, 4.5, 1.5)	1	
...	
샘플 101:	(6.3, 3.3, 6.0, 2.5)	2	
샘플 102:	(5.8, 2.7, 5.1, 1.9)	2	
...	
샘플 n:	(5.9, 3.0, 5.1, 1.8)	2	

그림 3-3 일반적인 데이터셋 표현 방법(iris 데이터셋 예시)

3.1.2 기계 학습 적용: 모델링과 예측

■ [프로그램 3-1(c)]

- SVM(support vector machine)이라는 기계 학습 모델을 사용

프로그램 3-1(c) iris에 기계 학습 적용: 모델링과 예측

```
07 from sklearn import svm 하이퍼 매개변수
08
09 s=svm.SVC(gamma=0.1,C=10) # svm 분류 모델 SVC 객체 생성하고
10 s.fit(d.data,d.target) # iris 데이터로 학습
11
12 new_d=[[6.4,3.2,6.0,2.5],[7.1,3.1,4.7,1.35]] # 101번째와 51번째 샘플을 변형하여
13 res=s.predict(new_d) 새로운 데이터 생성
14 print("새로운 2개 샘플의 부류는", res)
```

→ 학습을 하는 함수
→ 예측
Score: 짐작
→ label이 없는 데이터라
잘 예측했다고 단정지을 수 없음

새로운 2개 샘플의 부류는 [2 1]

- 09행: SVM의 분류기 모델 SVC 클래스의 객체를 생성하여 s에 저장
- 10행: 객체 s의 fit 함수는 훈련 집합을 가지고 학습을 수행(매개변수로 특징 벡터 iris.data와 레이블 iris,target을 설정)
- 13행: 객체 s의 predict 함수는 테스트 집합을 가지고 예측 수행

3.1.2 기계 학습 적용: 모델링과 예측

■ 훈련 집합과 테스트 집합 / 반드시 분리

- 훈련 집합: 기계 학습 모델을 학습하는데 쓰는 데이터로서 특징 벡터와 레이블 정보를 모두 제공
- 테스트 집합: 학습을 마친 모델의 성능을 측정하는데 쓰는 데이터로서 예측할 때는 특징 벡터 정보만 제공하고, 예측 결과를 가지고 정확률을 측정할 때 레이블 정보를 사용

$y = wx + b$
→ 모델 전체를 관찰 / 학습
↓
1차 모델? / 2차 모델?

NOTE 하이퍼 매개변수 설정 / 모델 파라미터: 학습 과정에서 최적화

하이퍼 매개변수(hyper parameter)란 모델의 동작을 제어하는 데 쓰는 변수이다. 모델의 학습을 시작하기 전에 설정해야 하는데, 적절한 값으로 설정해야 좋은 성능을 얻을 수 있다. 최적의 하이퍼 매개변수 값을 자동으로 설정하는 일을 하이퍼 매개변수 최적화(hyper parameter optimization)라 하는데, 이것은 기계 학습의 중요한 주제 중 하나다. 하이퍼 매개변수 최적화는 4.10절에서 다룬다.

3.2 인공지능 제품의 설계와 구현

■ [프로그램 3-1]은 인공지능 제품의 핵심

- 데이터를 읽고 모델링과 예측을 수행
- 붓꽃 영상을 획득하고 특징을 추출하는 컴퓨터 비전 모듈을 전처리로 붙이면 붓꽃 인식 프로그램 완성



(a) 인공지능이 인식할 붓꽃



(b) 과일의 등급 인식 인공지능



(c) 딸기 따는 로봇

그림 3-4 인공지능 제품

■ 실용적인 시스템 사례

- 과일 등급 분류([그림 3-4(b)])
- 딸기 따는 로봇([그림 3-4(c)])
- 모두 [프로그램 3-1]의 과정을 따름

3.2.1 인공지능 설계 사례: 과일 등급을 분류하는 기계

■ 사과를 상중하의 세 부류로 분류하는 인공지능 기계의 설계

1. 데이터 확보 → 하기 전에 *정확한 데이터 수집을 위해 하드웨어 환경이 필요함*
 - 상중하 비율이 비슷하게 수천 개의 사과 수집(데이터 편향 data bias 을 방지하기 위해 여러 농장에서 수집)
 - 카메라로 촬영하여 파일에 저장
2. 특징 벡터와 레이블 준비 → *딥 러닝이면 찾아줌*
 - 어떤 특징을 사용할까? 예) 사과의 크기, 색깔, 표면의 균일도는 분별력이 높은 특징
 - 컴퓨터 비전 기술로 특징 추출 프로그램 작성. 특징 추출하여 apple.data 파일에 저장
 - 사과 분류 전문가를 고용하여 레이블링. apple.target 파일에 저장
3. 학습하는 과정을 프로그래밍(훈련 데이터 사용)

```
from sklearn import svm
s=svm.SVC(gamma=0.1,C=10)
s.fit(apple.data, apple.target) # apple 데이터로 모델링
```

4. 예측 과정을 프로그래밍(새로 수집한 테스트 데이터 사용)

```
s.predict(x) # 새로운 사과에서 추출한 특징 벡터 x를 예측
```

3.2.1 인공지능 설계 사례: 과일 등급을 분류하는 기계

NOTE 데이터 편향

데이터 편향은 다양한 형태로 발생한다. 예를 들어 필기 숫자 데이터셋을 만들 때 편의상 대학생들 대상으로 수집했다면 정자체에 가까운 샘플의 비율이 높을 수 있다. 은행 창구에서 발생하는 전표나 우편 봉투에 쓰인 우편번호에서 수집하면 데이터 편향을 크게 줄일 수 있다. 예를 들어 3.3.3항에서 소개하는 유명인의 얼굴 영상을 담은lfw 데이터셋은 인종별, 나이별, 성별 편향이 있어 상업용 얼굴 인식 프로그램을 만드는 데 한계가 있다.

ex) 백화점에 있으면 층을 꼭 지켜서 쇼핑으로 예측, 현금으로 결제하면 고령자로 예측

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

■ 규칙 기반 방법

- 분류하는 규칙을 사람이 구현. 예) "꽃잎의 길이가 a보다 크고, 꽃잎의 너비가 b보다 작으면 Setosa"라는 규칙에서 a와 b를 사람이 결정해 줌
- 큰 데이터셋에서는 불가능하고, 데이터가 바뀌면 처음부터 새로 작업해야 하는 비효율성

■ 기계 학습 방법 → 특징 골라내기 학습

- 특징 벡터를 추출하고 레이블을 붙이는 과정은 규칙 기반과 동일(수작업 특징_{hand-crafted feature})
- 규칙 만드는 일은 기계학습 모델을 이용하여 자동으로 수행(이 책의 3~4장)

■ 딥러닝 방법

- 레이블을 붙이는 과정은 기계 학습과 동일
- 특징 벡터를 학습이 자동으로 알아냄. 특징 학습_{feature learning} 또는 표현 학습_{representation learning}을 한다고 말함
- 장점
 - 특징 추출과 분류를 동시에 최적화하므로 뛰어난 성능 보장
 - 인공지능 제품 제작이 빠름
- 이 책의 5장 이후

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

NOTE 원샷 학습, 퓨샷 학습, 준지도 학습

레이블을 붙이는 작업은 전문가가 해야 하므로 비용이 많이 든다. 따라서 딥러닝에서는 레이블이 있는 샘플을 하나만 사용해 학습하는 원샷 학습(1-shot learning), 몇 개의 샘플만 사용하는 퓨샷 학습(few-shot learning), 레이블이 있는 소량의 샘플과 레이블이 없는 대량의 샘플을 같이 사용하는 준지도 학습(semi-supervised learning)을 활용한다.

3.3 데이터에 대한 이해

■ 이 절은

- 특징 공간에서 데이터 분포를 살펴보고
- sklearn 라이브러리가 제공하는 여러 가지 데이터셋을 소개함

3.3.1 특징 공간에서 데이터 분포

■ iris 데이터

- 특징이 4개이므로 4차원 특징 공간을 형성
- 150개 샘플 각각은 4차원 특징 공간의 한 점
- [프로그램 3-2]는 차원을 하나 제외하고 3차원 공간에 데이터 분포를 그림

프로그램 3-2

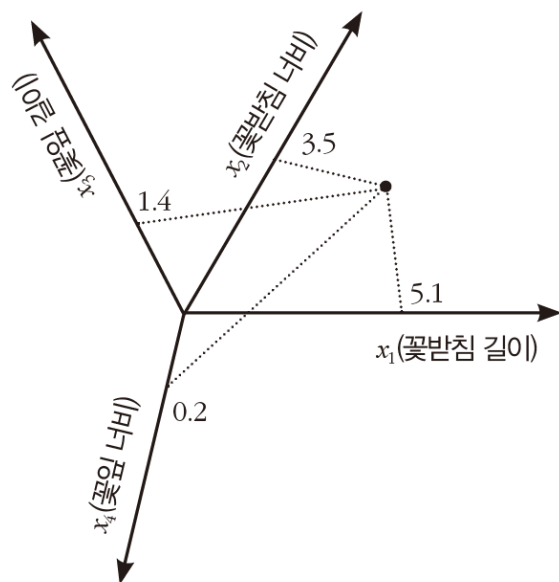
iris 데이터의 분포를 특징 공간에 그리기

```
01 import plotly.express as px
02
03 df = px.data.iris()
04 fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_width',
    color='species')      # petal_length를 제외하여 3차원 공간 구성
05 fig.show(renderer="browser")
```

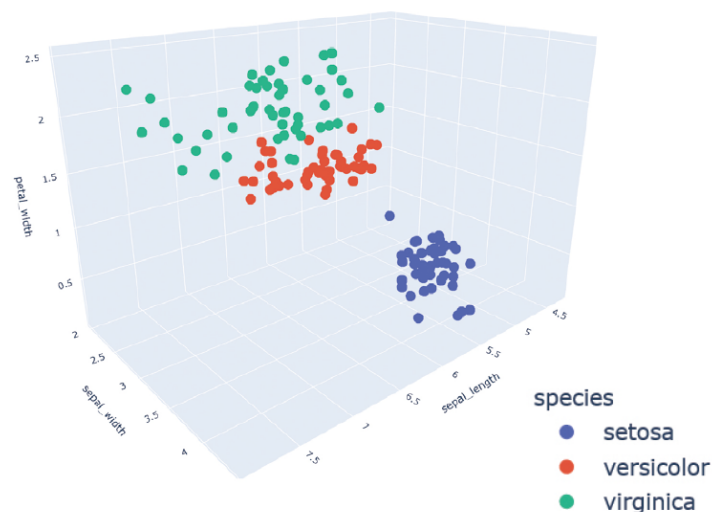
3.3.1 특징 공간에서 데이터 분포

■ 특징 공간에서 데이터 분포 관찰([프로그램 3-2]의 실행 결과인 [그림 3-5(b)])

- petal width(수직 축)에 대해 Setosa는 아래쪽, Virginica는 위쪽에 분포([프로그램 3-1(b)]의 실행 결과와 일치) → petal width 특징은 분별력^{discriminating power}이 뛰어남
- sepal width 축은 세 부류가 많이 겹쳐서 분별력이 낮음
- 전체적으로 보면, 세 부류가 3차원 공간에서 서로 다른 영역을 차지하는데 몇 개 샘플은 겹쳐 나타남



(a) 4차원 특징 공간(가상의 그림)



(b) 꽃잎 길이 축을 제외한 3차원 특징 공간

그림 3-5 iris 데이터를 특징 공간에 그리기

3.3.1 특징 공간에서 데이터 분포

NOTE 다차원 특징 공간

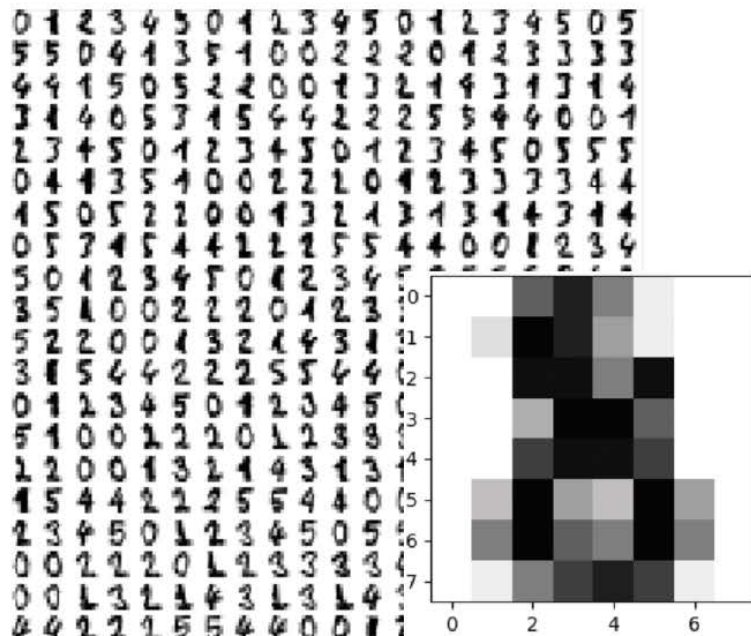
종이에 그릴 수 있는 공간은 3차원으로 제한되지만, 수학은 아주 높은 차원까지 다룰 수 있다. 예를 들어 2차원 상의 두 점 $\mathbf{x}=(x_1, x_2)$ 와 $\mathbf{y}=(y_1, y_2)$ 의 거리를 $d(\mathbf{x}, \mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ 으로 계산할 수 있는데, 4차원 상의 두 점 $\mathbf{x}=(x_1, x_2, x_3, x_4)$ 와 $\mathbf{y}=(y_1, y_2, y_3, y_4)$ 의 거리는 $d(\mathbf{x}, \mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2+(x_4-y_4)^2}$ 로 계산할 수 있다.

일반적으로 d 차원 상의 두 점의 거리는 $d(\mathbf{x}, \mathbf{y})=\sqrt{\sum_{i=1}^d (x_i - y_i)^2}$ 로 계산한다. 기계 학습에서는 d =수백~수만에 달하는 매우 고차원 특징 공간의 데이터를 주로 다룬다.

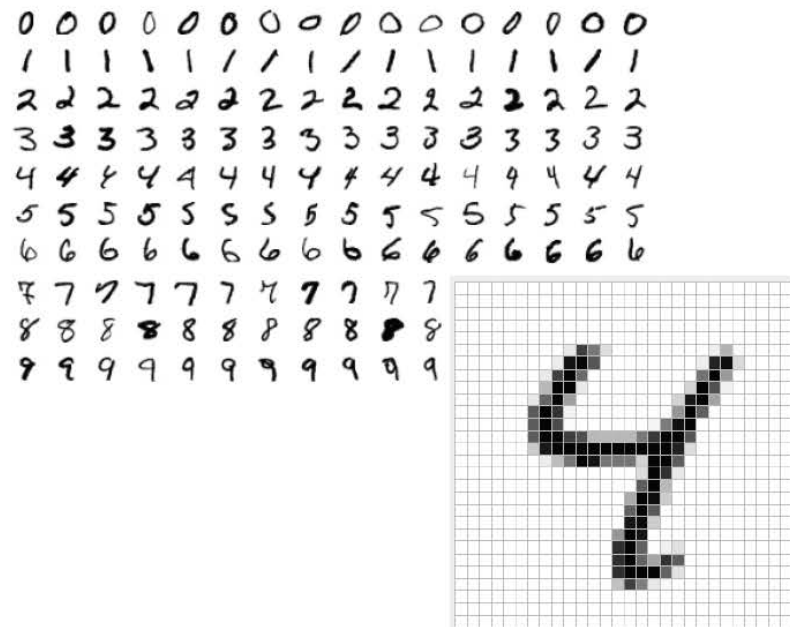
3.3.2 영상 데이터 사례: 필기 숫자

■ 두 가지 필기 숫자 데이터셋

- sklearn 데이터셋: 8*8 맵(64개 화소), 1797개 샘플, [0,16] 명암값
- MNIST 데이터셋: 28*28맵(784개 화소), 7만개 샘플, [0,255] 명암값



(a) sklearn에서 제공하는 데이터셋



(b) MNIST 데이터셋

그림 3-6 필기 숫자 데이터셋

3.3.2 영상 데이터 사례: 필기 숫자

■ [프로그램 3-3(a)]

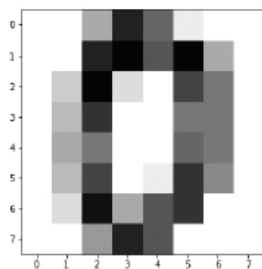
- matplotlib 라이브러리를 이용한 샘플 디스플레이와 샘플 내용(화소값) 출력

프로그램 3-3(a)

필기 숫자 데이터

```
01 from sklearn import datasets
02 import matplotlib.pyplot as plt
03
04 digit=datasets.load_digits()
05
06 plt.figure(figsize=(5,5))
07 plt.imshow(digit.images[0],cmap=plt.cm.gray_r,interpolation='nearest')
                                # 0번 샘플을 그림
08 plt.show()
09 print(digit.data[0])          # 0번 샘플의 화소값을 출력
10 print("이 숫자는 ",digit.target[0],"입니다.")
```

3.3.2 영상 데이터 사례: 필기 숫자



어떤 특징을 사용해야 높은 성능을 얻을 수 있을까? (3.4 절의 주제)

```
[ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
  0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
  0.  0.  0.  0.  6. 13. 10.  0.  0.  0.  0.]
```

이 숫자는 0입니다.

NOTE matplotlib을 이용한 시각화

파이썬에서 matplotlib 라이브러리는 시각화에 가장 널리 쓰인다. 인공지능은 학습 과정이나 예측 결과를 시각화하는 데 matplotlib을 자주 사용한다. matplotlib 사용이 처음이라면 부록 B를 공부해 기초를 먼저 다진다. matplotlib의 공식 사이트에서 제공하는 튜토리얼 문서를 공부하는 것도 효과적인 방법이다. [표 2-1]에서 제시한 <https://matplotlib.org/users>에 접속해 [Tutorials] 메뉴를 선택한다. 튜토리얼은 Introductory, Intermediate, Advanced로 나뉘어 있으니 최소한 Introductory 코스를 숙지하고 넘어간다.

3.3.3 영상 데이터 사례: lfw 얼굴 데이터셋

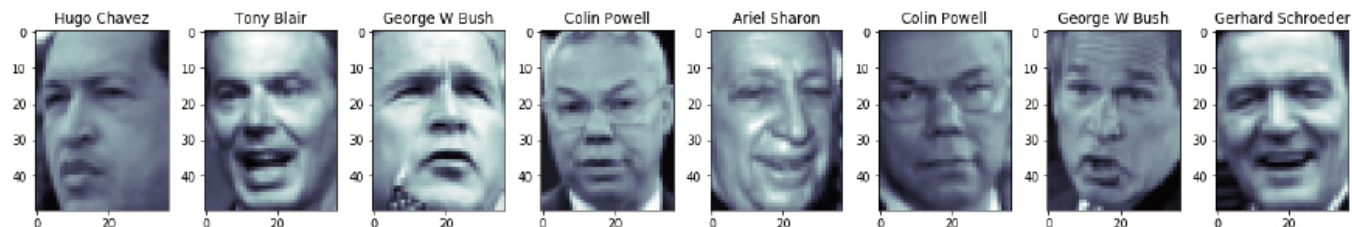
■ lfw(labeled faces in the wild) 데이터셋

- 5749명의 유명인의 얼굴 영상 13233장. 50*37맵. [0,255] 명암 값
- 데이터 편향 주의(어린이, 흑인 등이 적어 얼굴 인식 프로그램을 제작하는데 부적절)

프로그램 3-3(b)

lfw 유명인 얼굴 데이터셋

```
11 lfw=datasets.fetch_lfw_people(min_faces_per_person=70,resize=0.4) # 데이터셋 읽기
12
13 plt.figure(figsize=(20,5))
14
15 for i in range(8): # 처음 8명을 디스플레이
16     plt.subplot(1,8,i+1)
17     plt.imshow(lfw.images[i],cmap=plt.cm.bone)
18     plt.title(lfw.target_names[lfw.target[i]])
19
20 plt.show()
```



3.3.4 텍스트 데이터 사례: 20newsgroups

■ 20newsgroups 데이터셋

- 웹에서 수집한 문서를 20개 부류로 구분. 텍스트로 구성되어 샘플의 길이가 다름
- 시계열 데이터(단어가 나타나는 순서가 중요. 8장의 순환 신경망에서 다룸)

프로그램 3-3(c) 20newsgroups 데이터셋

```
21 news=datasets.fetch_20newsgroups(subset='train') # 데이터셋 읽기
22 print("*****Wn",news.data[0],"Wn*****")      # 0번 샘플 출력
23 print("이 문서의 부류는 <",news.target_names[news.target[0]],"> 입니다.")
```

From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is

...

이 문서의 부류는 < rec.autos >입니다.

3.4 특징 추출과 표현

- 기계 학습의 전형적인 과정
 - 실제에서는 다양한 형태로 나타남



그림 3-7 기계 학습의 과정

3.4.1 특징의 분별력

■ 사람은 직관적으로 분별력^{discriminating power}이 높은 특징을 사용

- 예) [그림 3-8]의 두 텀블러를 구분하는 특징
 - 글씨 방향, 몸통 색깔, 손잡이 유무, 뚜껑 유무 등
 - 뚜껑 유무라는 특징은 분별력이 없음
 - 손잡이 유무라는 특징은 높은 분별력



그림 3-8 텀블러를 구분하기 위한 특징으로 무엇이 좋을까?

3.4.1 특징의 분별력

■ 기계 학습은 높은 분별력을 지닌 특징을 사용해야 함

- 예) 100여년 전의 iris 데이터는 사람이 네 종류의 특징을 자를 들고 직접 추출
- 현대에서는 붓꽃 영상을 그대로 입력하면 딥러닝이 최적의 특징을 추출해 줌

■ 다양한 형태의 특징 공간

- 실제 세상은 [그림 3-9(c)~(d)]와 같은 비선형 데이터를 생성(데이터의 원천적인 성질 또는 측정이나 레이블링 오류, 비합리적인 특징 추출 알고리즘에 기인)
- 가급적 [그림 3-9(d)]보다 [그림 3-9(c)]와 같은 특징을 사용해야 함

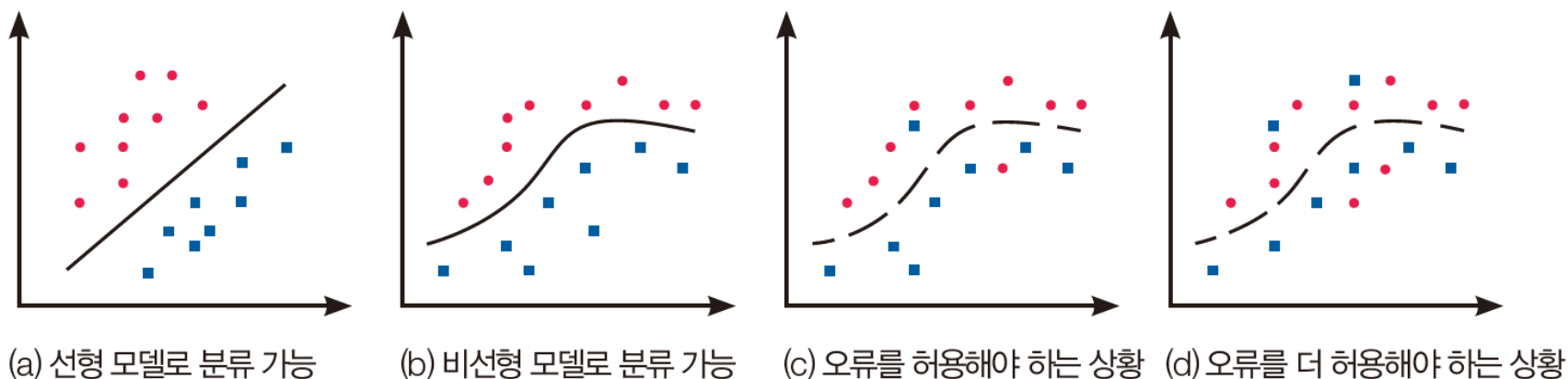


그림 3-9 특징의 분별력

3.4.2 특징 값의 종류

■ 수치형 특징

- 예) iris의 네 개 특징은 실수
- 거리 개념이 있음
- 실수 또는 정수 또는 이진값

■ 범주형 특징 → 원 핫코드 이용: 원 벡터로 표현 $[1], [0]$

- 학점, 수능 등급, 혈액형, 지역 등
- 순서형: 학점, 수능 등급 등
 - 거리 개념이 있음. 순서대로 정수를 부여하면 수치형으로 취급 가능
- 이름형
 - 혈액형, 지역 등으로 거리 개념이 없음
 - 보통 원핫_{one-hot} 코드로 표현. 예) A형(1,0,0,0), B형(0,1,0,0), O형(0,0,1,0), AB형(0,0,0,1)

3.5 필기 숫자 인식

- [그림 3-7]의 절차에 따라 필기 숫자 데이터셋을 가지고 프로그래밍 연습
 - 특징 추출을 위한 코드 작성
 - sklearn이 제공하는 fit 함수로 모델링(학습)
 - predict 함수로 예측

3.5.1 화소 값을 특징으로 사용

■ 화소 각각을 특징으로 간주

- sklearn의 필기 숫자는 8*8 맵으로 표현되므로 64차원 특징 벡터
- 2차원 구조를 1차원 구조로 변환
- 예) [프로그램 3-3(a)]의 샘플

```
x=(0,0,5,13,9,1,0,0,0,0,13,15,10,15,5,0,0,3,15,2,0,11,8,0,0,4,12,0,0,8,8,  
0,0,5,8,0,0,9,8,0,0,4,11,0,1,12,7,0,0,2,14,5,10,12,0,0,0,0,6,13,10,0,0,0)
```

3.5.1 화소 값을 특징으로 사용 [프로그램 3-1]과 매우 비슷함

■ [프로그램 3-4]

- 07~08행: SVC로 학습 수행(특징 벡터 digit.data, 레이블 digit.target 사용)
- 11~14행: 맨 앞의 세 개 샘플을 테스트 집합으로 간주하고 예측을 해봄
- 17~20행: 훈련 집합을 테스트 집합으로 간주하고 정확률을 측정

프로그램 3-4

필기 숫자 인식 - 각 화소를 특징으로 간주하여 64차원 특징 벡터 사용

```
01 from sklearn import datasets
02 from sklearn import svm
03
04 digit=datasets.load_digits()
05
06 # svm의 분류기 모델 SC를 학습
07 s=svm.SVC(gamma=0.1,C=10)
08 s.fit(digit.data,digit.target) # digit 데이터로 모델링
09
10 # 훈련 집합의 앞에 있는 샘플 3개를 새로운 샘플로 간주하고 인식해봄
11 new_d=[digit.data[0],digit.data[1],digit.data[2]]
12 res=s.predict(new_d)
13 print("예측값은", res)
14 print("참값은", digit.target[0],digit.target[1],digit.target[2])
15
16 # 훈련 집합을 테스트 집합으로 간주하여 인식해보고 정확률을 측정
17 res=s.predict(digit.data)
18 correct=[i for i in range(len(res)) if res[i]==digit.target[i]]
19 accuracy=len(correct)/len(res)
20 print("화소 특징을 사용했을 때 정확률=",accuracy*100, "%")
```

데이터 불러와서
학습하고
검증하고

예측값은 [0 1 2]

참값은 0 1 2

화소 특징을 사용했을 때 정확률=100.0%

훈련 집합으로 했기 때문에 잘 작동하는지 알 수 없음

3.5.1 화소 값을 특징으로 사용

NOTE 컴퓨터 프로그래밍에 쓰이는 패턴의 중요성

[프로그램 3-1]과 [프로그램 3-4]는 코드의 많은 부분을 공유한다. 여기에서 알 수 있는 것처럼 컴퓨터 프로그래밍에도 패턴이 있다. 이 패턴을 잘 기억하고 따라 하는 것은 좋은 프로그래머로 성장하는 지름길이다. 4.6.3항에서는 '디자인 패턴'이라는 용어를 사용해 프로그래밍에 쓰이는 패턴의 중요성을 다시 한 번 강조한다.

3.6 성능 측정

■ 객관적인 성능 측정의 중요성

- 모델 선택할 때 중요
- 현장 설치 여부 결정할 때 중요

■ 일반화_{generalization} 능력

- 학습에 사용하지 않았던 새로운 데이터에 대한 성능
- 가장 확실한 방법은 실제 현장에 설치하고 성능 측정 → 비용 때문에 실제 적용 어려움
- 주어진 데이터를 분할하여 사용하는 지혜 필요

3.6.1 혼동 행렬과 성능 측정 기준

■ 혼동 행렬_{confusion matrix}

- 부류 별로 옳은 분류와 틀린 분류의 개수를 기록한 행렬
 - n_{ij} 는 모델이 i 라고 예측했는데 실제 부류는 j 인 샘플의 개수

		참값(그라운드 트루스)					
		부류 1	부류 2	...	부류 j	...	부류 c
예 측 한 부 류	부류 1	n_{11}	n_{12}		n_{1j}		n_{1c}
	부류 2	n_{21}	n_{22}		n_{2j}		n_{2c}
	...						
	부류 i	n_{i1}	n_{i2}		n_{ij}		n_{ic}
	...						
	부류 c	n_{c1}	n_{c2}		n_{cj}		n_{cc}

(a) 부류가 c 개인 경우

		그라운드 트루스	
		긍정	부정
예 측 값	긍정	TP	FP
	부정	FN	TN

T: ~~과~~원 = 예측값
F: ≠

(b) 부류가 2개인 경우

그림 3-10 혼동 행렬

- 이진 분류에서 긍정_{positive}과 부정_{negative}
 - 검출하고자 하는 것이 긍정(환자가 긍정이고 정상인이 부정, 불량품이 긍정이고 정상이 부정)
- 참 긍정(TP), 거짓 부정(FN), 거짓 긍정(FP), 참 부정(TN)의 네 경우

3.6.1 혼동 행렬과 성능 측정 기준

■ 널리 쓰이는 성능 측정 기준

■ 정확률_{accuracy}

- 부류가 불균형일 때 성능을 제대로 반영하지 못함

$$\text{정확률} = \frac{\text{맞힌 샘플 수}}{\text{전체 샘플 수}} = \frac{\text{대각선 샘플 수}}{\text{전체 샘플 수}} \quad (3.2)$$

■ 특이도_{specificity}와 민감도_{sensitivity} (의료에서 주로 사용)

$$\text{특이도} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \text{민감도} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.3)$$

■ 정밀도_{precision}와 재현률_{recall} (정보검색에서 주로 사용)

$$\text{정밀도} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{재현율} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.4)$$

3.6.2 훈련/검증/테스트 집합으로 쪼개기

- 주어진 데이터를 적절한 비율로 훈련, 검증, 테스트 집합으로 나누어 씀
 - 모델 선택 포함: 훈련/검증/테스트 집합으로 나눔
 - 모델 선택 제외: 훈련/테스트 집합으로 나눔



(a) 모델 선택 포함



(b) 모델 선택 제외

그림 3-11 훈련/검증/테스트 집합으로 쪼개기

3.6.2 훈련/검증/테스트 집합으로 쪼개기

■ [프로그램 3-5]는 모델 선택 제외

- 08행: train_test_split 함수로 훈련 60%, 테스트 40%로 랜덤 분할
- 12행: 훈련 집합 x_train, y_train을 fit 함수에 주어 학습 수행
- 14행: 테스트 집합의 특징 벡터 x_test를 predict 함수에 주어 예측 수행
- 17~20행: 테스트 집합의 레이블 y_test를 가지고 혼동 행렬 계산

프로그램 3-5

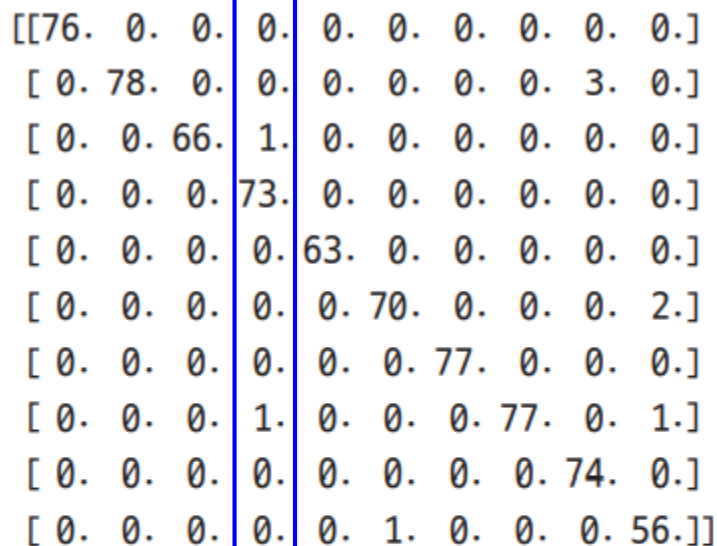
필기 숫자 인식 - 훈련 집합으로 학습하고 테스트 집합으로 성능 측정

```
01  from sklearn import datasets
02  from sklearn import svm
03  from sklearn.model_selection import train_test_split
04  import numpy as np
05
06  # 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
07  digit=datasets.load_digits()
08  x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)
09
```

3.6.2 훈련/검증/테스트 집합으로 쪼개기

```
10 # svm의 분류 모델 SVC를 학습
11 s=svm.SVC(gamma=0.001)
12 s.fit(x_train,y_train)
13
14 res=s.predict(x_test)
15
16 # 혼동 행렬 구함
17 conf=np.zeros((10,10))
18 for i in range(len(res)):
19     conf[res[i]][y_test[i]]+=1
20 print(conf)
21
22 # 정확률 측정하고 출력
23 no_correct=0
24 for i in range(10):
25     no_correct+=conf[i][i]
26 accuracy=no_correct/len(res)
27 print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

예) 부류 3에 속하는 75개 샘플 중 73개를 3, 1개를 2, 1개를 7로 인식



[76.	0.	0.	0.	0.	0.	0.	0.	0.	0.]
[0.	78.	0.	0.	0.	0.	0.	0.	3.	0.]
[0.	0.	66.	1.	0.	0.	0.	0.	0.	0.]
[0.	0.	0.	73.	0.	0.	0.	0.	0.	0.]
[0.	0.	0.	0.	63.	0.	0.	0.	0.	0.]
[0.	0.	0.	0.	0.	70.	0.	0.	0.	2.]
[0.	0.	0.	0.	0.	0.	77.	0.	0.	0.]
[0.	0.	0.	1.	0.	0.	0.	77.	0.	1.]
[0.	0.	0.	0.	0.	0.	0.	0.	74.	0.]
[0.	0.	0.	0.	0.	1.	0.	0.	0.	56.]]

테스트 집합에 대한 정확률은 98.74826147426981%입니다.

3.6.2 훈련/검증/테스트 집합으로 쪼개기

NOTE 난수를 사용하기 때문에 실행할 때마다 다른 결과가 나오는 프로그램

[프로그램 3-5]는 실행할 때마다 출력이 다르게 나온다. 08행의 `train_test_split` 함수가 난수를 사용해 데이터를 분할하기 때문이다. 앞으로 등장하는 프로그램에서도 난수를 사용하는 경우가 있는데 마찬가지로 실행할 때마다 다른 결과를 얻게 된다. 동일한 실행 결과를 얻으려면 08행 이전에 `np.random.seed(0)`을 추가하면 된다. 매개 변수 0은 다른 값을 사용해도 된다. 어떤 값이든 고정시키면 매번 같은 난수 열이 생성된다.

3.6.3 교차 검증

■ 훈련/테스트 집합 나누기의 한계

- 우연히 높은 정확률 또는 우연히 낮은 정확률 발생 가능성

■ k -겹 교차 검증 k-fold cross validation

- 훈련 집합을 k 개의 부분집합으로 나누어 사용. 한 개를 남겨두고 $k-1$ 개로 학습한 다음 남겨둔 것으로 성능 측정. k 개의 성능을 평균하여 신뢰도 높임



(a) 모델 선택 포함



(b) 모델 선택 제외

그림 3-12 k -겹 교차 검증($k=5$ 인 경우)

3.6.3 교차 검증

- [프로그램 3-6]은 digit 데이터에 교차 검증 적용(모델 선택 제외)
 - cross_val_score 함수가 교차 검증 수행해줌(cv=5는 5-겹 교차 검증하라는 뜻)

프로그램 3-6

필기 숫자 인식 - 교차 검증으로 성능 측정

```
01 from sklearn import datasets
02 from sklearn import svm
03 from sklearn.model_selection import cross_val_score
04 import numpy as np
05
06 digit=datasets.load_digits()
07 s=svm.SVC(gamma=0.001)
08 accuracies=cross_val_score(s,digit.data,digit.target,cv=5) # 5-겹 교차 검증
09
10 print(accuracies)
11 print("정확률(평균)=%0.3f, 표준편차=%0.3f"%(accuracies.mean()*100,accuracies.std()))
```

```
[0.97527473 0.95027624 0.98328691 0.99159664 0.95774648]
```

```
정확률(평균)=97.164, 표준편차=0.015
```

- 실행 결과 정확률이 들쭉날쭉. 한번만 시도하는 [프로그램 3-5]의 위험성을 잘 보여줌
- k를 크게 하면 신뢰도 높아지지만 실행 시간이 더 걸림

3.7 인공지능은 어떻게 인식을 하나?

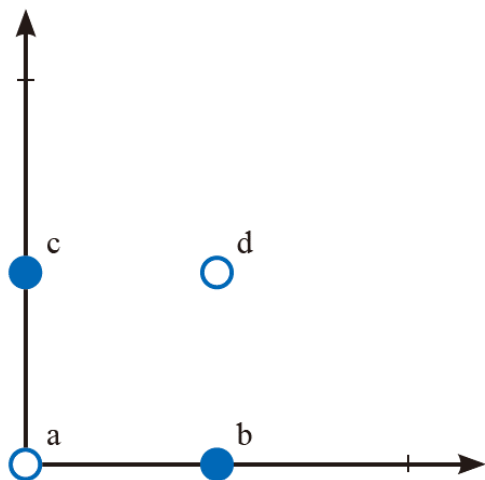
- 지금까지는 모델(분류기)의 원리에 대한 이해없이 프로그래밍 실습
 - 지금까지 기계 학습 모델인 SVM을 블랙 박스로 보고 프로그래밍
 - 동작 원리에 대한 이해 없으면 언젠가 한계가 드러남

3.7.1 특징 공간을 분할하는 결정 경계

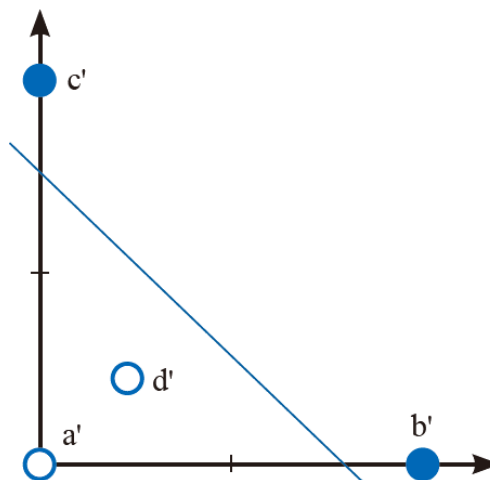
■ 인공지능의 인식은 철저히 수학에 의존

- 샘플은 특징 벡터로 표현되며, 특징 벡터는 특징 공간의 한 점에 해당
- 인식 알고리즘은 원래 특징 공간을 성능을 높이는데 더 유리한 새로운 특징 공간으로 여러 차례 변환한 다음 최종적으로 특징 공간을 분할하여 부류를 결정
- 특징 공간 변환 예

$$\text{원래 특징 공간 } \mathbf{x}=(x_1, x_2) \rightarrow \text{새로운 특징 공간 } \mathbf{x}'=\left(\frac{x_1}{2x_1x_2+0.5}, \frac{x_2}{2x_1x_2+0.5}\right) \quad (3.5)$$



(a) 원래 특징 공간



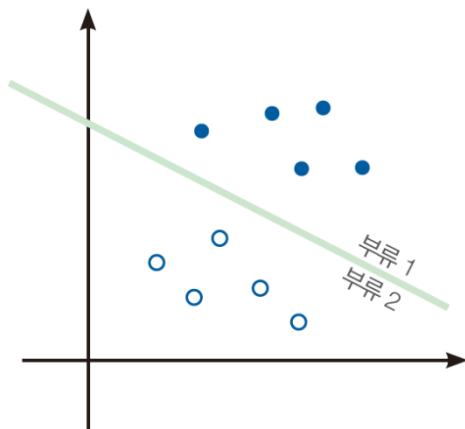
(b) 선형 분리가 가능하도록 변환된 새로운 공간

그림 3-13 특징 공간의 변환

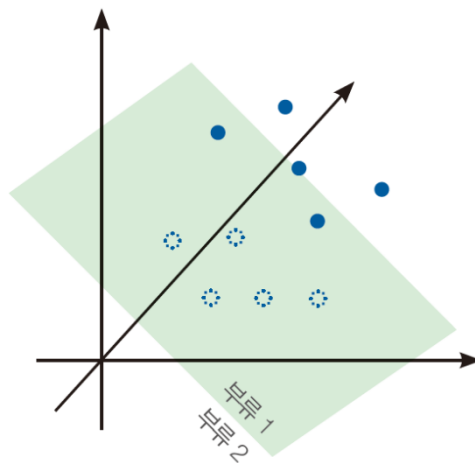
3.7.1 특징 공간을 분할하는 결정 경계

■ 특징 공간을 분할하는 결정 경계_{decision boundary}

- [그림 3-14]는 2차원과 3차원 예
- 2~3차원은 그림을 그릴 수 있는데 4차원 이상은 수학적 상상력 필요



(a) 2차원 특징 공간에서 결정 직선



(b) 3차원 특징 공간에서 결정 평면

그림 3-14 특징 공간의 분할

■ 현대 기계 학습이 다루는 데이터

- 수백~수만 차원 특징 공간
- 고차원 공간에서 부류들이 서로 꼬여있는 매우 복잡한 분포
- 딥러닝은 층을 깊게 하여 여러 단계의 특징 공간 변환을 수행(특징 학습 또는 표현 학습이라 부름)

3.7.1 특징 공간을 분할하는 결정 경계

■ 결정 경계를 정하는 문제 에서 고려 사항

- 비선형 분류기 nonlinear classifier 사용
- 과잉 적합 overfitting 회피
 - 과잉 적합은 아웃라이어를 맞히려고 과다하게 복잡한 결정 경계를 만드는 현상
 - 훈련 집합에 대한 성능은 높지만 테스트 집합에 대해서는 형편없는 성능(낮은 일반화)
 - 원인과 대처법은 5.6.2항에서 설명

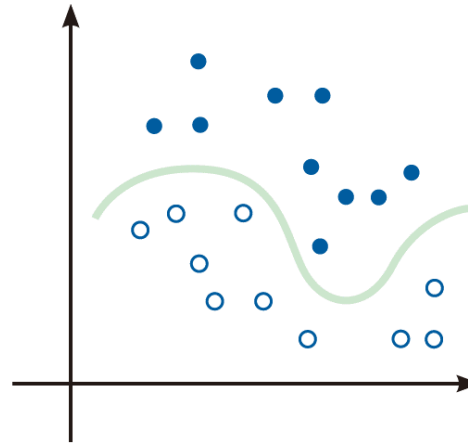
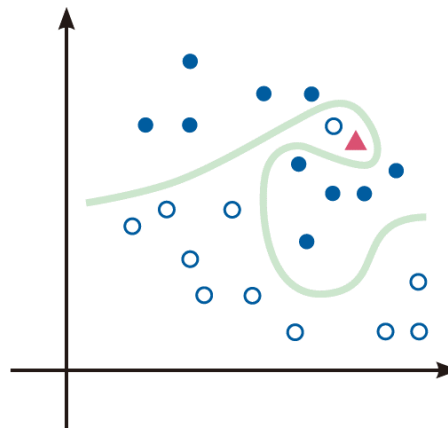
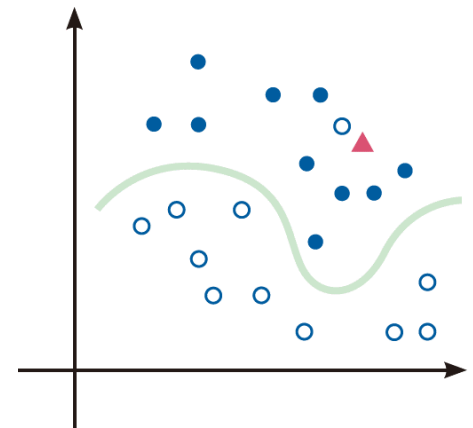


그림 3-15 선형 분리 불가능한 데이터를 위한 비선형 분류기



(a) 아웃라이어에 의한 과잉 적합



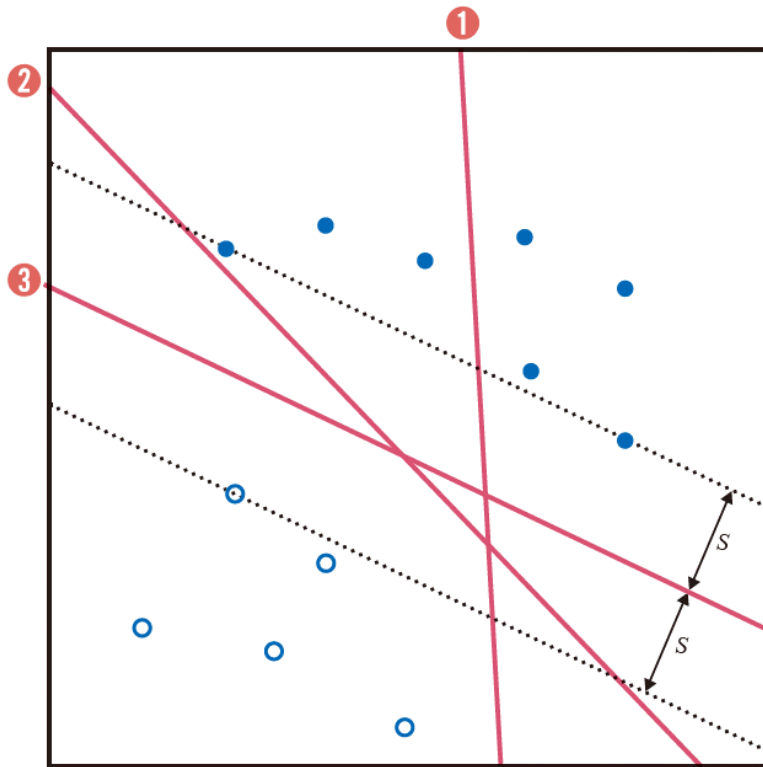
(b) 아웃라이어를 걸러낸 경우

그림 3-16 과잉 적합 현상

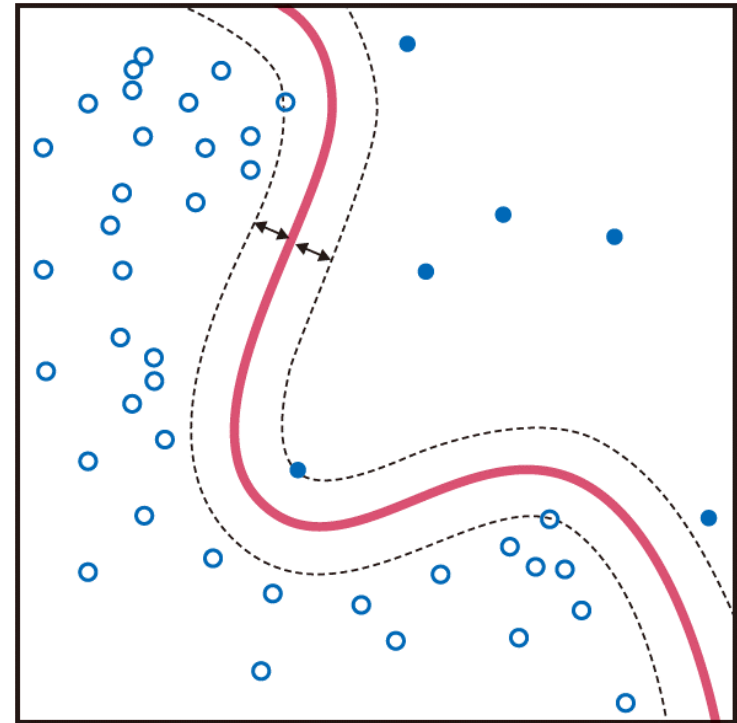
3.7.2 SVM의 원리

■ SVM의 동기

- 100% 정확률인 두 분류기 ②와 ③은 같은가?



(a) 선형 SVM



(b) 비선형 SVM

그림 3-17 여백을 최대화하여 일반화 능력을 극대화하는 SVM

3.7.2 SVM의 원리

■ 기계 학습의 목적은 일반화 능력을 극대화하는 것

- SVM은 일반화 능력을 높이려 여백을 최대화
- 분류기 ②는 빨간색 부류에 조금만 변형이 생겨도 결정 경계를 넘을 가능성. ③은 두 부류 모두에 대해 멀리 떨어져 있어 경계를 넘을 가능성이 낮음
- SVM은 두 부류까지의 거리인 $2s$ 를 여백_{margin}이라 부름. SVM 학습 알고리즘은 여백을 최대화하는 결정 경계를 찾음

■ SVM을 비선형 분류기로 확장

- 원래 SVM은 선형 분류기([그림 3-17(a)])
- 커널 트릭을 사용하여 비선형 분류기로 확장(커널 함수를 사용하여 선형 공간을 비선형 공간으로 변형[그림 3-17(b)])
- 커널 함수로는 polynomial function, radial basis function, sigmoid 함수를 사용
 - 커널 함수의 종류와 커널 함수의 모양을 조절하는 매개변수는 하이퍼 매개변수

3.7.2 SVM의 원리

■ C라는 하이퍼 매개변수

- 지금까지 모든 샘플을 옳게 분류하는 경우를 다룸. 실제로는 오류를 허용하는 수밖에 없음
- C를 크게 하면, 잘못 분류한 훈련 집합의 샘플을 적은데 여백이 작아짐(훈련 집합에 대한 정확률은 높지만 일반화 능력 떨어짐)
- C를 작게 하면, 여백은 큰데 잘못 분류한 샘플이 많아짐(훈련 집합에 대한 정확률은 낮지만 일반화 능력 높아짐)

■ [프로그램 3-4]의 07행

- 커널 함수로 기본값 rbf를 사용. gamma는 rbf 관련한 매개변수
- C=10 사용

```
07 s=svm.SVC(gamma=0.1,C=10)
```