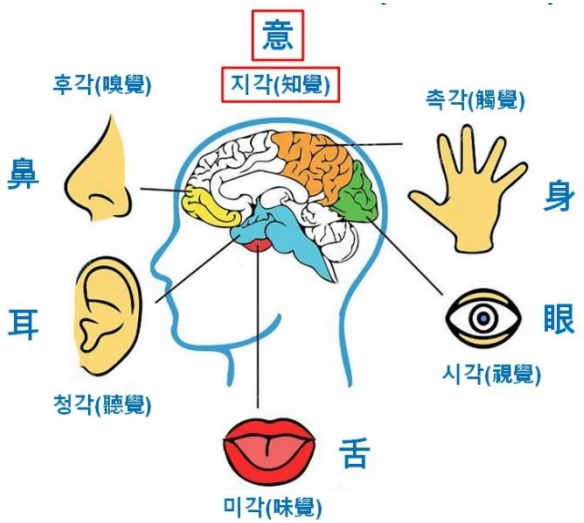
기계 학습과 인식

인식(Recognition)

인식

사물을 분별하고 판단할 줄 아는 것.
타인이 말한 소리, 얼굴, 감정 등을 인식
생존과 자기 발전에 필수



인공지능 인식

자율 주행차, 음성인식 챗봇, 주문 받는 로봇 등

AI, ML, DL


인공지능 > 머신러닝 > 딥러닝 포함 관계
딥러닝 ⊂ 머신러닝 ⊂ 인공지능
인공지능 | Artificial Intelligence
사람의 지적 능력을 컴퓨터를 통해 구현하는 기술
머신러닝 | Machine Learning
사람이 정한 모델과 특징 추출 방법을 이용하여
데이터를 기반으로 학습해서 추론할 수 있게 하는 기술
딥러닝 | Deep Learning
인공신경망 방법을 이용해 만든 머신러닝 기술로,
빅데이터 학습에 적합한 기술

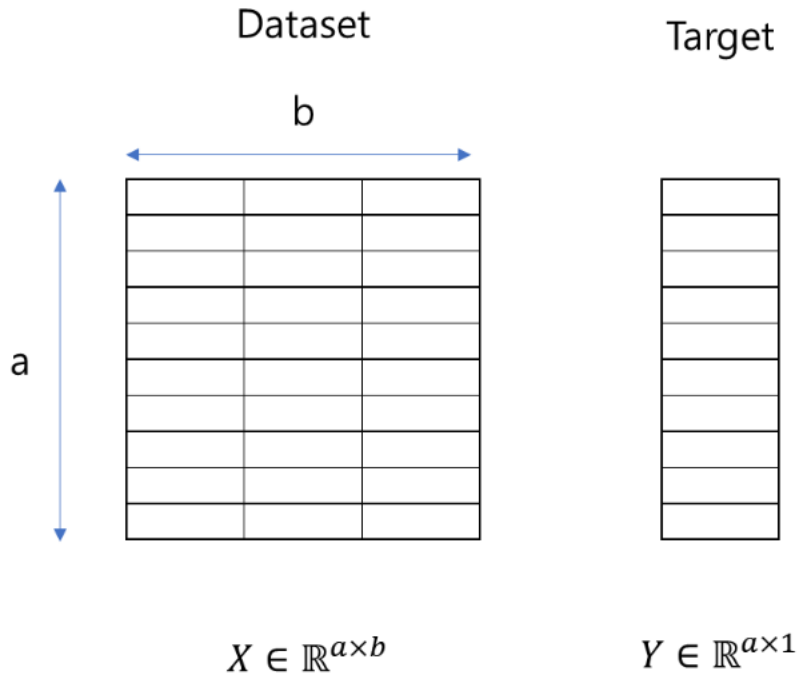
기계학습(Machine Learning) 기초

데이터셋(dataset)

기계 학습의 근간(데이터를 통해 학습)

에너지를 만드는 연료에 해당

데이터가 없으면 기계학습 적용이 불가능



scikit-learn 데이터셋(dataset) 종류

Toy Dataset

load_boston: 보스톤 집값 데이터

load_iris: 아이리스 붓꽃 데이터

load_diabetes: 당뇨병 환자 데이터

load_digits: 손글씨 데이터

load_linnerud: multi-output regression 용 데이터

load_wine: 와인 데이터

load_breast_cancer: 위스콘신 유방암 환자 데이터

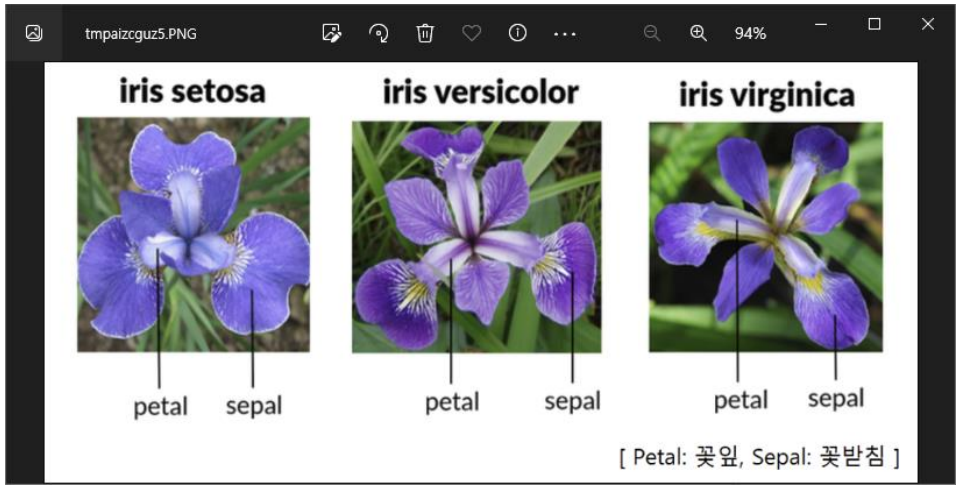
scikit-learn

데이터셋(dataset)

- data: 샘플 데이터, Numpy 배열로 이루어져 있음
- Target: Label 데이터, Numpy 배열로 이루어져 있음
- feature_names: Feature 데이터의 이름
- target_names: Label 데이터의 이름
- DESCR: 데이터 셋의 설명
- filename: 데이터 셋의 파일 저장 위치 (csv)

iris 붓꽃 이미지 보기

```
from PIL import Image
img = Image.open('./img/iris.png')
img.show()
```



load_iris

샘플 데이터 불러오기

```
from sklearn.datasets import load_iris  # 사이킷런에서 붓꽃 데이터셋 로드
iris = load_iris()
print(iris.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

데이터셋의 정보 확인하기

```
print(iris.DESCR)
```

```
: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
```

- **총 150개의 붓꽃 샘플 데이터**
- **4개의 피처(feature)**
- **3가지의 품종**

load_iris

feature와 label 이름 확인하기

```
print(iris.feature_names)
print(iris.target_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
```

Target 속성

```
print("데이터셋 크기 : ", iris['target'].shape)
print("데이터셋 내용 : ", iris['target'])
```

```
데이터셋 크기 :  (150,)
데이터셋 내용 :  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

load_iris

Data 속성

```
print("데이터셋 크기 : ", iris['data'].shape)
print("데이터셋 내용 : ", iris['data'][:10, :])
print("데이터셋 내용 : ", iris['data'][:3, :2])
```

```
데이터셋 크기 :  (150, 4)
데이터셋 내용 :  [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

```
데이터셋 내용 :  [[5.1 3.5]
 [4.9 3. ]
 [4.7 3.2]]
```

Data frame 변환

```
import pandas as pd                # pandas 라이브러리 읽어들이기
df = pd.DataFrame(iris.data)
print(df.head(5))
```

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

load_iris

Data frame 변환

```
df = pd.DataFrame(iris.data, columns=iris['feature_names'])
print(df.head(5))
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
df.columns = ['꽃받침길이', '꽃받침너비', '꽃잎길이', '꽃잎너비']
print(df.head(5))
```

	꽃받침길이	꽃받침너비	꽃잎길이	꽃잎너비
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

iris 데이터셋 읽기

iris의 내용 살피기

```
for i in range(0,len(d.data)):                    # 샘플을 순서대로 출력
    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.target(레이블)

d.data(특징 벡터)

ML에서 데이터셋의 표현

샘플을 특징 벡터와 레이블로 표현

- 특징 벡터는 x로 표기(d는 특징의 개수로서 특징 벡터의 차원이라 부름)

  특징 벡터: $\mathbf{x}=(x_1,x_2,\cdots,x_d)$

- 레이블은 0,1,2,...,c-1의 값 또는 1,2,...,c-1,c의 값 또는 원핫 코드

  원핫 코드는 한 요소만 1인 이진열
  ex) Setosa는 (1,0,0), Versicolor는 (0,1,0), Virginica는 (0,0,1)로 표현

	특징 벡터 $\mathbf{x}=(x_1, x_2,\cdots, x_d)$	레이블(참값) y
샘플 1:	(5.1, 3.5, 1.4, 0.2)	0
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

iris 데이터셋
(n=150, d=4)

기계학습적용 : 모델링과 예측

SVM(Support Vector Machine)

하이퍼 매개변수

```
from sklearn import svm
s=svm.SVC(gamma=0.1,C=10)          # svm 분류 모델 SVC 객체 생성하고
s.fit(d.data,d.target)             # iris 데이터로 학습
                훈련 집합

# 101번째와 51번째 샘플을 변형하여 새로운 데이터 생성
new_d=[[6.4,3.2,6.0,2.5],[7.1,3.1,4.7,1.35]]
          테스트 집합
res=s.predict(new_d)
print("새로운 2개 샘플의 부류는", res)
```

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

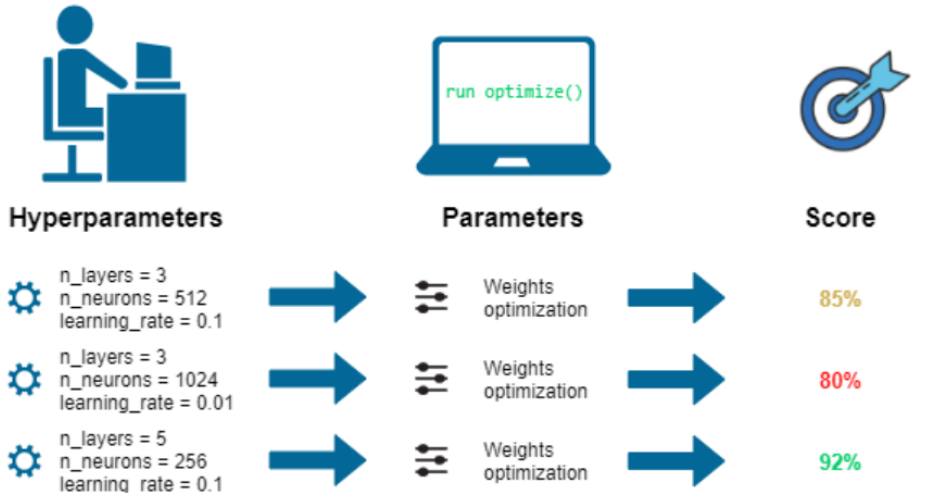
기계학습적용 : 모델링과 예측

SVM의 하이퍼 파라미터

- 모델이 학습을 하면서 최적의 값을 자동으로 찾는 것이 아니라 사람이 직접 지정해 주어야 하는 변수
- 모델의 구조나 학습 방법을 설정하는 값
- 적절하게 설정하면 모델의 성능을 크게 향상시킬 수 있음
  - 커널 유형(kernel type): 데이터 사이의 유사도를 측정하는 방법을 결정
  - 커널 매개변수(kernel parameter): 커널의 특성을 결정
  - C 값(C value): SVM의 규제 강도를 결정
  - 에포크 수(epoch): 모델이 전체 데이터 세트를 몇 번 학습할지를 결정

SVM의 모델 파라미터

- 학습 과정 중에 데이터를 기반으로 조절되며 최적화 됨
- 모델의 학습 과정에서 학습되는 값
- 모델 파라미터를 적절하게 학습시키면 모델의 성능을 향상
  - 가중치(weight): 데이터 포인트와 결정 경계 사이의 거리를 결정
  - 편향(bias): 결정 경계를 이동하는 데 사용



https://www.kdnuggets.com/2020/02/practical-hyperparameter-optimization.html

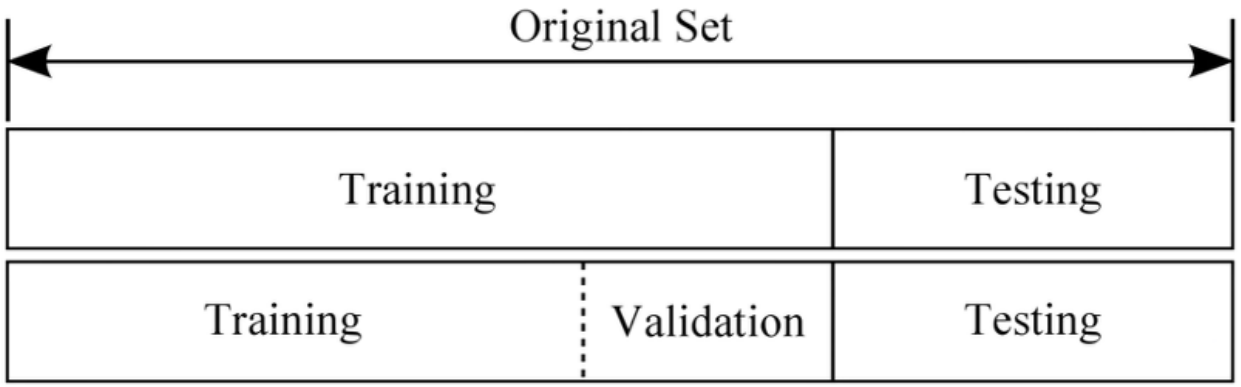
기계학습적용 : 모델링과 예측

훈련, 검증, 테스트 집합

훈련 집합: 모델을 학습시킬 때 사용하는 데이터 집합

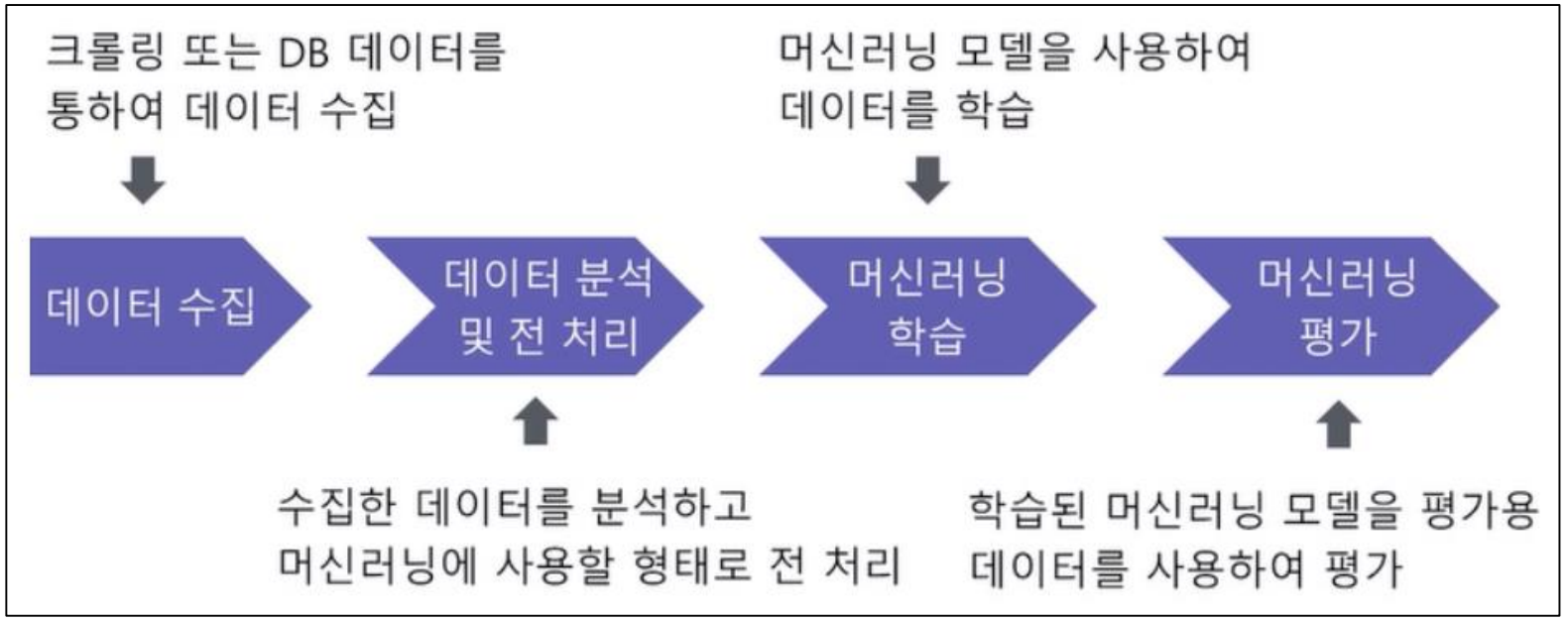
검증 집합: 수많은 종류의 학습된 모델이 있을 때, 그 중 최적의 모델(즉 최적 하이퍼 파라미터)을 선택하기 위해 사용하는 데이터 집합

테스트 집합: 검증을 거친 최종 모델의 일반화 성능을 검증할 때 사용하는 데이터 집합



기계학습적용 : 모델링과 예측

ML 4단계



인공지능 제품의 설계와 구현

인공지능 제품의 핵심

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

(a) 인공지능이 인식할 붓꽃

(b) 과일의 등급 인식 인공지능

(c) 딸기 따는 로봇

인공지능 설계 사례 : 과일 등급 분류

사과를 상중하의 세 부류로 분류하는 인공지능 기계의 설계

1. 데이터 확보
   - 상중하 비율이 비슷하게 수천 개의 사과 수집
     (데이터 편향 data bias을 방지하기 위해 여러 농장에서 수집)
   - 카메라로 촬영하여 파일에 저장

2. 특징 벡터와 레이블 준비
   - 어떤 특징을 사용할까? 예) 사과의 크기, 색깔, 표면의 균일도
   - 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
```

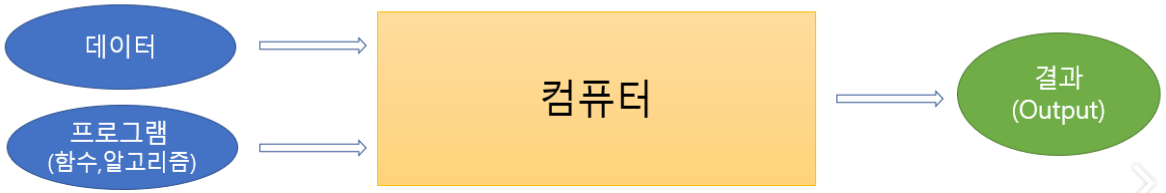
4. 예측 과정을 프로그래밍(새로 수집한 테스트 데이터 사용)

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

Programming vs. ML vs. DL

머신 러닝과 기존 프로그래밍과의 차이

전통적 프로그래밍 방식



머신러닝(Machine Learning)

Programming vs. ML vs. DL

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 벡터를 추출하고 레이블을 붙이는 과정은 규칙 기반과 동일(수작업 특징hand-crafted feature)
- 규칙 만드는 일은 기계학습 모델을 이용하여 자동으로 수행

딥러닝 방법

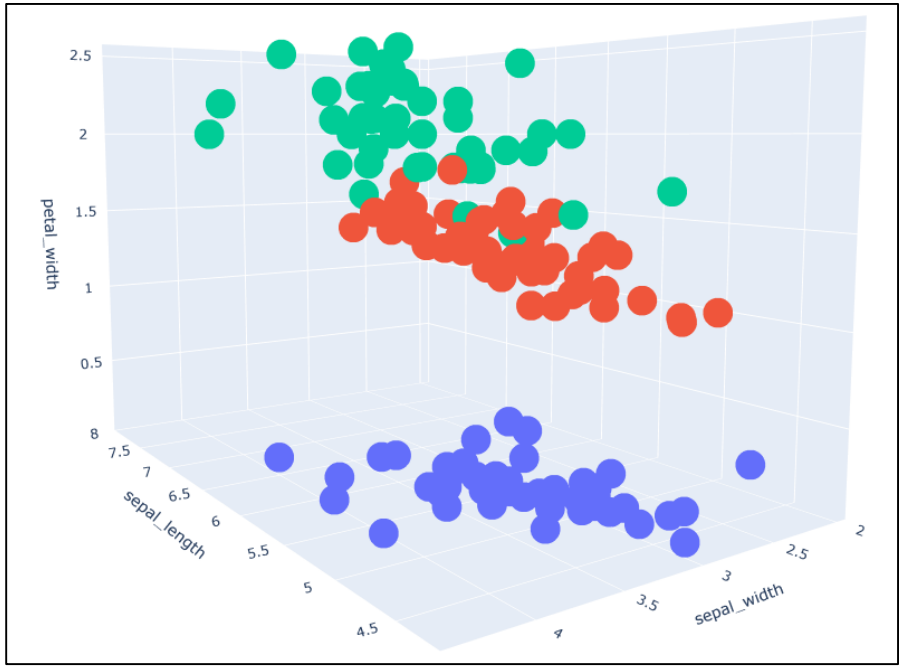
- 레이블을 붙이는 과정은 기계 학습과 동일
- 특징 벡터를 학습이 자동으로 알아냄. 특징 학습feature learning 또는 표현 학습representation learning을 한다고 말함
- 특징 추출과 분류를 동시에 최적화하므로 뛰어난 성능 보장
- 인공지능 제품 제작이 빠름

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

특징 공간에서 데이터 분포

iris 데이터

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




```
df = px.data.iris()
# petal_length를 제외하여 3차원 공간 구성
fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_width', color='species')
fig.show(renderer="browser")
```

특징 공간에서 데이터 분포

NOTE 다차원 특징 공간

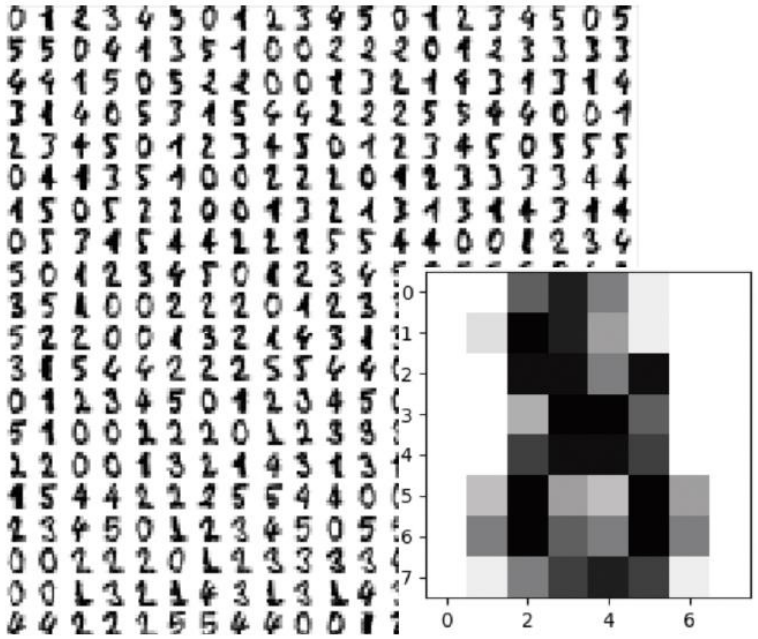
종이에 그릴 수 있는 공간은 3차원으로 제한되지만, 수학은 아주 높은 차원까지 다룰 수 있다. 예를 들어 2차원 상의 두 점 $\mathbf{x}=(x_1,x_2)$와 $\mathbf{y}=(y_1,y_2)$의 거리를 $d(\mathbf{x},\mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$으로 계산할 수 있는데, 4차원 상의 두 점 $\mathbf{x}=(x_1,x_2,x_3,x_4)$와 $\mathbf{y}=(y_1,y_2,y_3,y_4)$의 거리는 $d(\mathbf{x},\mathbf{y})=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2+(x_4-y_4)^2}$로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

일반적으로 d차원 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.

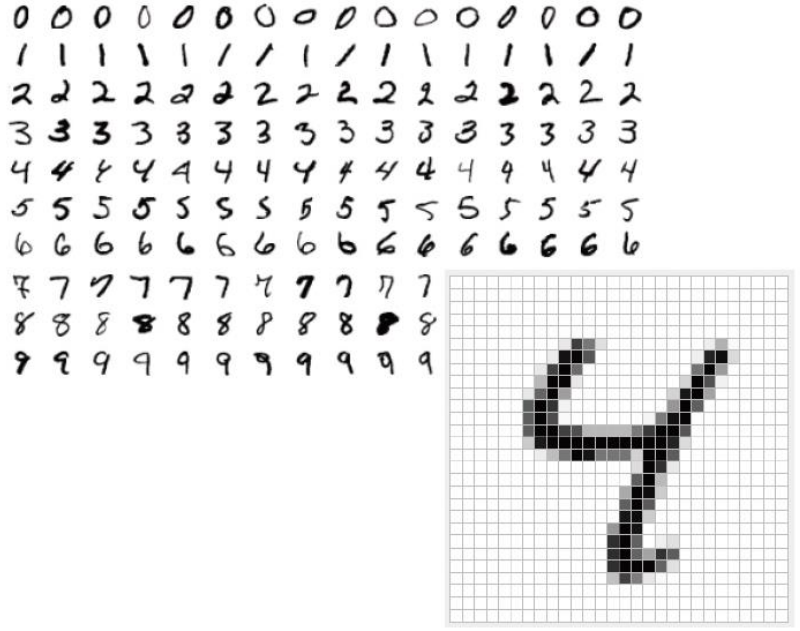
영상 데이터 사례 : 필기 숫자

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

영상 데이터 사례 : 필기 숫자

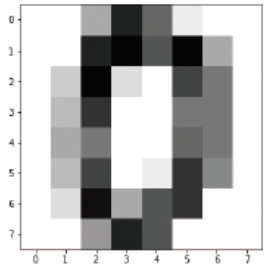
matplotlib 라이브러리를 이용한 샘플 디스플레이와 샘플 내용(화소값) 출력

```
from sklearn import datasets
import matplotlib.pyplot as plt

digit=datasets.load_digits()

plt.figure(figsize=(5,5))
# 0번 샘플을 그림
plt.imshow(digit.images[0],cmap=plt.cm.gray_r,interpolation='nearest')
plt.show()
print(digit.data[0])
# 0번 샘플의 화솟값을 출력
print("이 숫자는 ",digit.target[0],"입니다.")
```

영상 데이터 사례 : 필기 숫자



어떤 특징을 사용해야 높은 성능을 얻을 수 있을까?

```
[ 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.]
```

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/users에 접속해 [Tutorials] 메뉴를 선택한다. 튜토리얼은 Introductory, Intermediate, Advanced로 나뉘어 있으니 최소한 Introductory 코스를 숙지하고 넘어간다.

영상 데이터 사례 : lfw 얼굴 데이터셋

lfw(labeled faces in the wild) 데이터셋

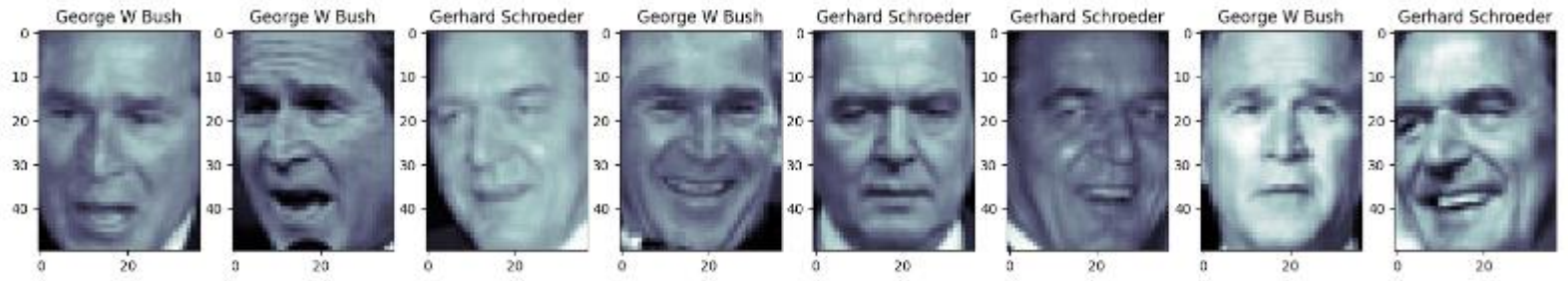
- 5749명의 유명인의 얼굴 영상 13233장. 50*37맵. [0,255] 명암 값
- 데이터 편향 주의(어린이, 흑인 등이 적어 얼굴 인식 프로그램을 제작하는데 부적절)

```
lfw(labeled faces in the wild) 데이터셋
lfw=datasets.fetch_lfw_people(min_faces_per_person=70,resize=0.4) # 데이터셋 읽기

plt.figure(figsize=(20,5))

for i in range(8): # 처음 8명을 디스플레이
    plt.subplot(1,8,i+1)
    plt.imshow(lfw.images[i],cmap=plt.cm.bone)
    plt.title(lfw.target_names[lfw.target[i]])

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



텍스트 데이터 사례 : 20newsgroups

20newsgroups 데이터셋

- 웹에서 수집한 문서를 20개 부류로 구분. 텍스트로 구성되어 샘플의 길이가 다름
- 시계열 데이터(단어가 나타나는 순서가 중요.

```
news=datasets.fetch_20newsgroups(subset='train') # 데이터셋 읽기
print("*****₩n",news.data[0],"₩n*****") # 0번 샘플 출력
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
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.
```

특징추출과 표현

기계 학습에서의 전형적인 과정

- 실제에서는 다양한 형태로 나타남

특징의 분별력

사람은 직관적으로 분별력discriminating power이 높은 특징을 사용

- 두 텀블러를 구분하는 특징
  - 글씨 방향, 몸통 색깔, 손잡이 유무, 뚜껑 유무 등
  - 뚜껑 유무라는 특징은 분별력이 없음
  - 손잡이 유무라는 특징은 높은 분별력

텀블러를 구분하기 위한 특징으로 무엇이 좋을까?

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

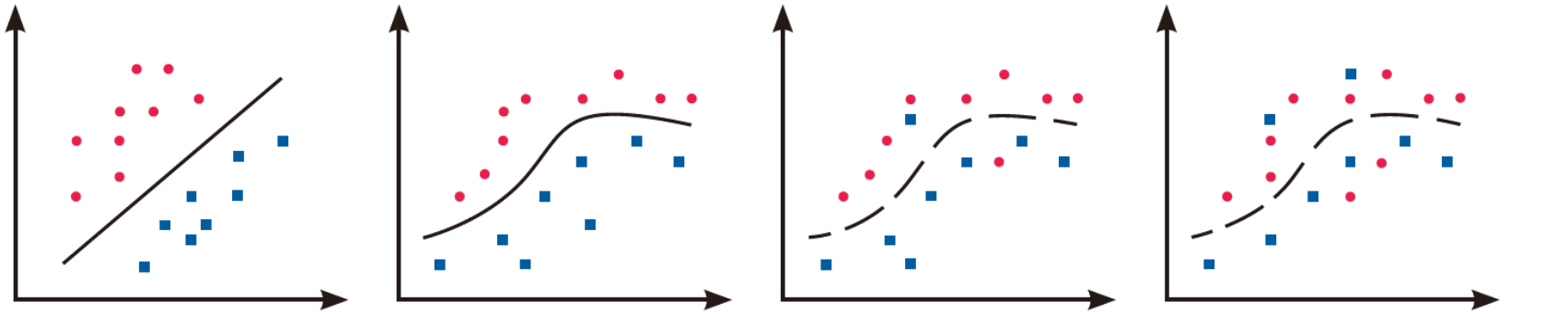


그림 3-9 특징의 분별력

특징 값의 종류

수치형 특징

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

범주형 특징

- 학점, 수능 등급, 혈액형, 지역 등
- 순서형: 학점, 수능 등급 등
  - 거리 개념이 있음. 순서대로 정수를 부여하면 수치형으로 취급 가능

이름형 특징

- 혈액형, 지역 등으로 거리 개념이 없음
- 보통 원핫one-hot 코드로 표현. 예) A형(1,0,0,0), B형(0,1,0,0), O형(0,0,1,0), AB형(0,0,0,1)

필기 숫자 인식

필기 숫자 데이터셋을 가지고 프로그래밍 연습

1. 특징 추출을 위한 코드 작성
2. sklearn이 제공하는 fit 함수로 모델링(학습)
3. predict 함수로 예측

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)
```

필기 숫자 인식

```
from sklearn import datasets
from sklearn import svm

digit=datasets.load_digits()

# svm의 분류기 모델 SC를 학습
s=svm.SVC(gamma=0.1,C=10)
s.fit(digit.data,digit.target) # digit 데이터로 모델링

# 훈련 집합의 앞에 있는 샘플 3개를 새로운 샘플로 간주하고 인식해봄
new_d=[digit.data[0],digit.data[1],digit.data[2]]
res=s.predict(new_d)
print("예측값은", res)
print("참값은", digit.target[0],digit.target[1],digit.target[2])

# 훈련 집합을 테스트 집합으로 간주하여 인식해보고 정확률을 측정
res=s.predict(digit.data)
correct=[i for i in range(len(res)) if res[i]==digit.target[i]]
accuracy=len(correct)/len(res)
print("화소 특징을 사용했을 때 정확률=",accuracy*100, "%")
```

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

성능 측정

객관적인 성능 측정의 중요성

- 모델 선택할 때 중요
- 현장 설치 여부 결정할 때 중요

일반화generalization 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

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

혼동 행렬과 성능 측정 기준

혼동 행렬

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

		참값(그라운드 트루스)					
		부류 1	부류 2	…	부류 j	…	부류 c
예측한 부류	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 c개인 경우

		그라운드 트루스	
		긍정	부정
예측값	긍정	TP	FP
	부정	FN	TN

(b) 부류가 2개인 경우

그림 3-10 혼동 행렬

- 이진 분류에서 긍정positive과 부정negative

혼동 행렬과 성능 측정 기준

널리 쓰이는 훈련 측정 기준

- 정확률accuracy
  - 부류가 불균형일 때 성능을 제대로 반영하지 못함

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

- 특이도specificity와 민감도sensitivity (의료에서 주로 사용)

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

- 특이도specificity와 민감도sensitivity (의료에서 주로 사용)

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

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

그림 3-11 훈련/검증/테스트 집합으로 쪼개기

훈련/검증/테스트 집합으로 쪼개기

```
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np

# 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
digit=datasets.load_digits()
x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)
# svm의 분류 모델 SVC를 학습
s=svm.SVC(gamma=0.001)
s.fit(x_train,y_train)

res=s.predict(x_test)

# 혼동 행렬 구함
conf=np.zeros((10,10))
for i in range(len(res)):
    conf[res[i]][y_test[i]]+=1
print(conf)

# 정확률 측정하고 출력
no_correct=0
for i in range(10):
    no_correct+=conf[i][i]
accuracy=no_correct/len(res)
print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

- 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

훈련/검증/테스트 집합으로 쪼개기

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.
```

교차 검증

훈련/테스트 집합 나누기의 한계

- 우연히 높은 정확률 또는 우연히 낮은 정확률 발생 가능성

k-겹 교차 검증k-fold cross validation

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

(a) 모델 선택 포함

(b) 모델 선택 제외

그림 3-12 k-겹 교차 검증(k=5인 경우)

교차 검증

[프로그램 3-6]은 digit 데이터에 교차 검증 적용(모델 선택 제외)

- cross_val_score 함수가 교차 검증 수행 해줌(cv=5는 5-겹 교차 검증하라는 뜻)

```
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import cross_val_score
import numpy as np

digit=datasets.load_digits()
s=svm.SVC(gamma=0.001)
accuracies=cross_val_score(s,digit.data,digit.target,cv=5) # 5-겹 교차 검증

print(accuracies)
print("정확률(평균)=%0.3f, 표준편차 =%0.3f"%(accuracies.mean()*100,accuracies.std()))
```

```
[0.975      0.95       0.98328691 0.99164345 0.96100279]
정확률(평균)=97.219, 표준편차 =0.015
```

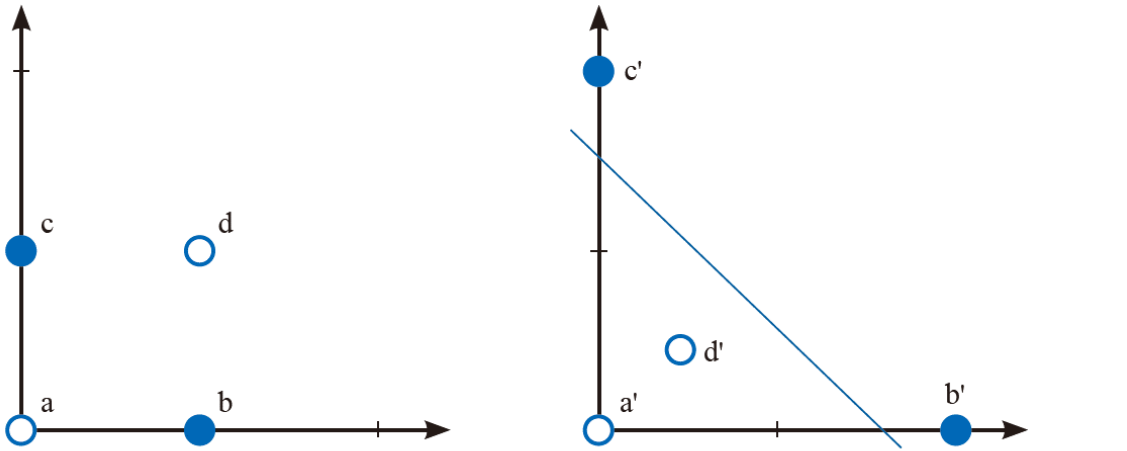
- 실행 결과 정확률이 들쭉날쭉. 한번만 시도하는 [프로그램 3-5]의 위험성을 잘 보여줌
- k를 크게 하면 신뢰도 높아지지만 실행 시간이 더 걸림

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

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



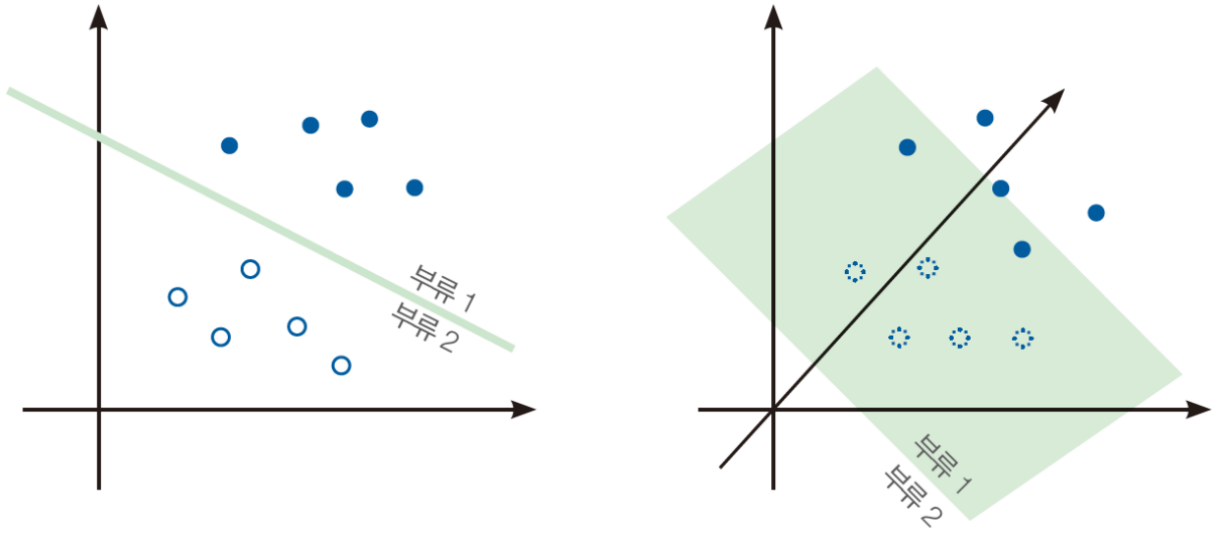
(a) 원래 특징 공간 (b) 선형 분리가 가능하도록 변환된 새로운 공간

그림 3-13 특징 공간의 변환

특징 공간을 분할하는 결정 경계

특징 공간을 분할하는 결정 경계decision 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)

특징 공간을 분할하는 결정 경계

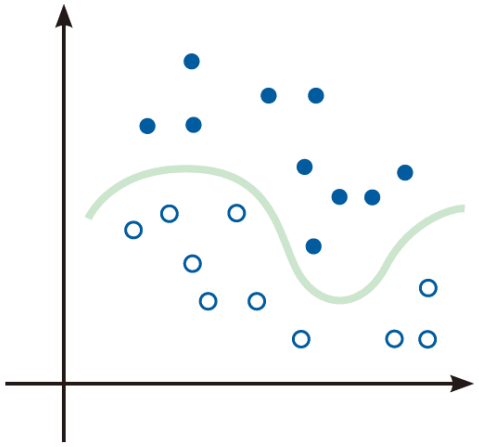
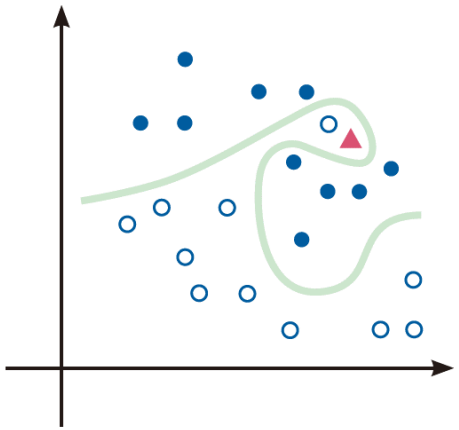


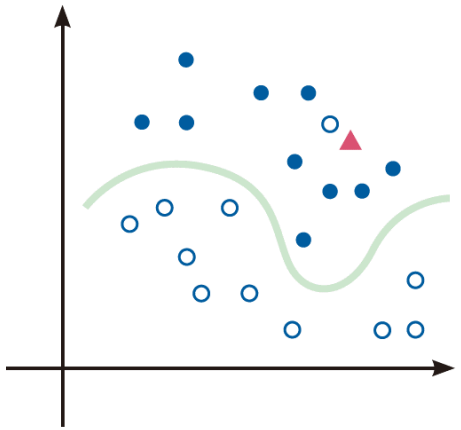
그림 3-15 선형 분리 불가능한 데이터를 위한 비선형 분류기

결정 경계를 정하는 문제에서 고려 사항

- 비선형 분류기nonlinear classifier 사용
- 과잉 적합overfitting 회피
- 과잉 적합은 아웃라이어를 맞히려고 과다하게 복잡한 결정 경계를 만드는 현상
- 훈련 집합에 대한 성능은 높지만 테스트 집합에 대해서는 형편없는 성능(낮은 일반화)



(a) 아웃라이어에 의한 과잉 적합



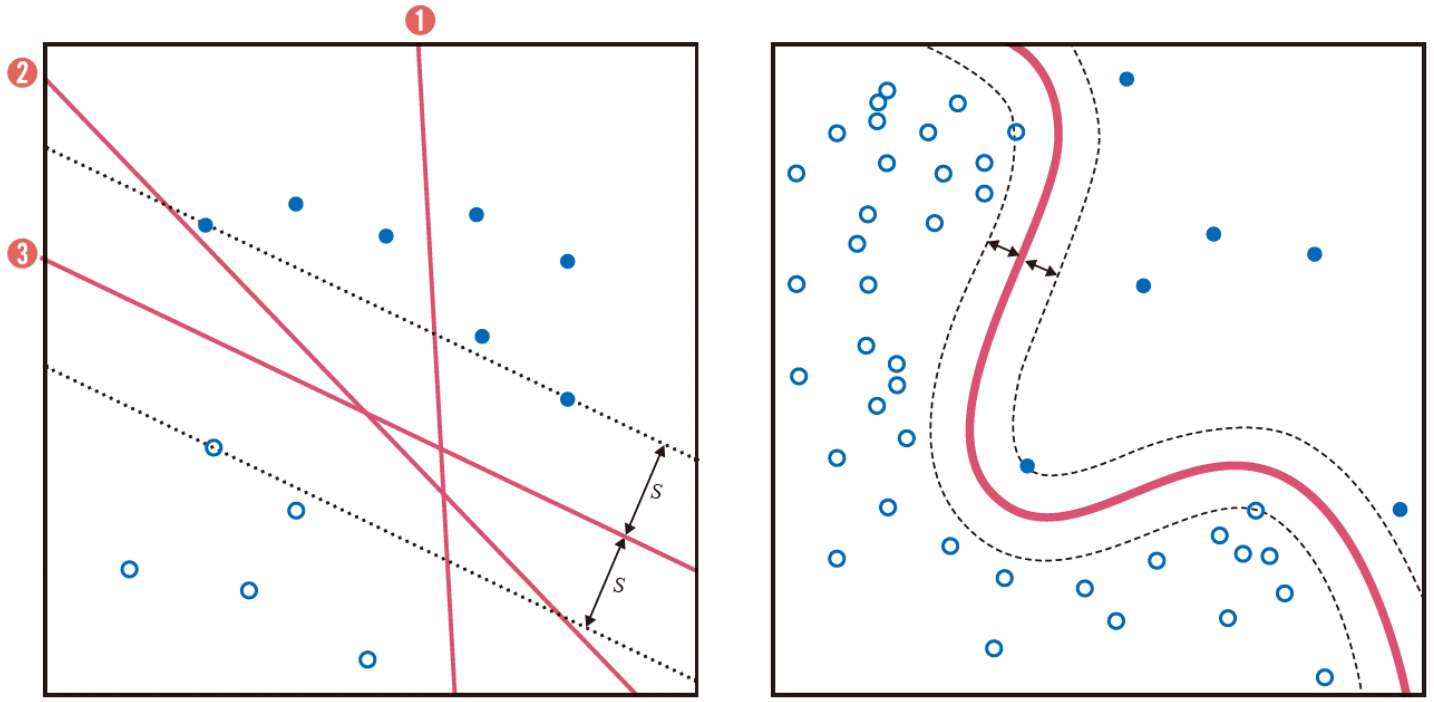
(b) 아웃라이어를 걸러낸 경우

그림 3-16 과잉 적합 현상

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

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 2s를 여백margin이라 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

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)
```

Reference

- 모두의 인공지능 with 파이썬, 이영호, 길벗
- 처음 만나는 인공지능, 김대수, 생능출판
- 인공지능, 이건명, 생능출판
- 처음 배우는 인공지능, 다다 사토시, 송교석, 한빛미디어
- https://kh-kim.github.io/nlp_with_deep_learning_blog/
- https://learning-sarah.tistory.com/entry/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D%EB%94%A5%EB%9F%AC%EB%8B%9D-%EB%8D%B0%EC%9D%B4%ED%84%B0%EC%85%8B-%EC%A0%9C%EA%B3%B5-%EC%82%AC%EC%9D%B4%ED%8A%B8