인공지능

Artificial Intelligence

기계학습과 인식

본 강의자료는 한빛아카데미 〈파이썬으로 만드는 인공지능〉을 내용을 기반으로 제작 및 보완되었음을 알립니다.

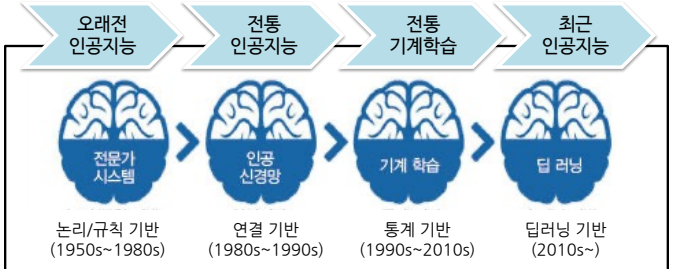
PREVIEW

- **사람은 끊임없이 주위환경을 인식함**
  - 다른 사람이 말한 소리를 인식해 응답하고, 길에서 만난 사람의 얼굴을 인식해 인사를 하며, 책에 쓰인 글씨를 인식해 지식을 습득함
  - 이와 같이, 인식Recognition or Understanding은 인간 생존과 자기 발전에 필수적인 지능 행위에 해당함

- **주위 환경을 인식할 수 있는 인공지능 기계는 인간의 삶에 편의를 제공함**
  - 얼굴을 인식해 친구와 찍은 사진만 골라내는 앱 ➔ 스마트폰
  - 도로 환경을 인식해 스스로 운행하는 자율주행자동차 ➔ 자동차
  - 음성 인식으로 주문을 받는 챗봇 ➔ 매장용 키오스크
  - 고객이 손으로 작성한 신청서를 인식해 입력하는 기계 ➔ 은행의 업무 처리용 서버

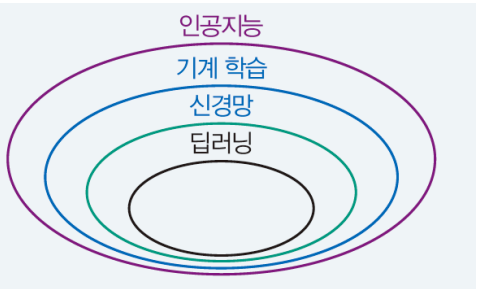
인공지능 SW 혹은 인공지능 SW가 탑재된 기계

PREVIEW

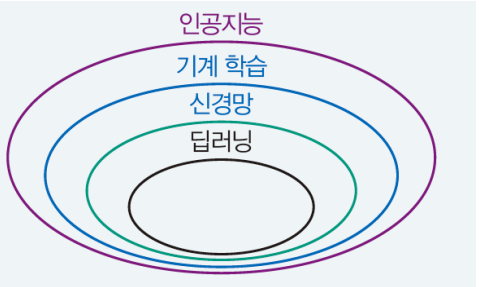


- **현대에는 주로 기계 학습을 이용해 인식 프로그램을 제작**
  - 기계 학습은 데이터로 학습을 수행해 인식 성능을 최대화하는 방법론
  - 기계 학습 모델에는 아주 다양한 종류가 있으며, 이들을 정교하게 구현한 소프트웨어가 공개되어 있음 (예. 사이킷런scikit-learn)

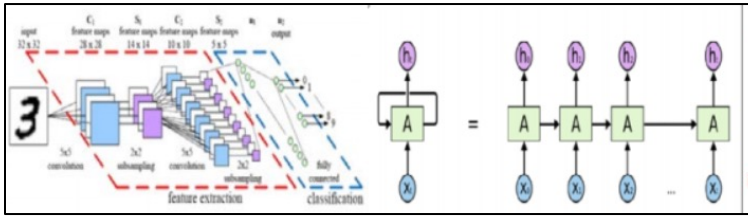
- **인공지능, 기계 학습, 신경망, 딥러닝의 관계**
  - (관점1) 인공지능과 기계 학습은 풀어야 하는 문제와 푸는 데 쓰는 도구의 관계
  - 현대에는 인공지능을 구현하는 데 주로 기계 학습을 활용
    - 예전에는 규칙 기반 방법을 주로 활용했음
  - 현재 기계 학습이 가장 널리 쓰는 모델은 신경망
    - 예전에는 주로 층의 개수가 적은 얕은 신경망을 사용했음
    - 현대 인공지능은 수십~수백 층을 가진 깊은 신경망을 사용하여 우수한 인공지능을 구현함
    - 깊은 신경망을 학습하고 활용하는 기술을 딥러닝 이라고 함



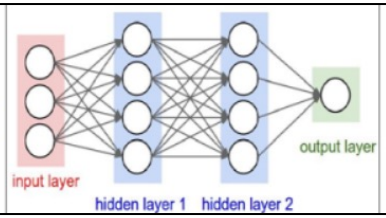
PREVIEW



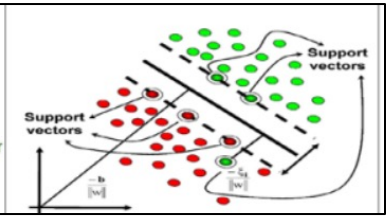
- **인공지능, 기계 학습, 신경망, 딥러닝의 관계**
  - 현재 기계 학습이 가장 널리 쓰는 모델은 신경망
    - 예전에는 주로 층의 개수가 적은 얕은 신경망을 사용했음
    - 현대 인공지능은 수십~수백 층을 가진 깊은 신경망을 사용하여 우수한 인공지능을 구현함
    - 깊은 신경망을 학습하고 활용하는 기술을 〈딥러닝〉 이라고 함



깊은 신경망(CNN, RNN)



얕은 신경망 (MLP)



기계학습(SVM)

3.1 기계 학습 기초

- 기계 학습에서 **데이터**는 에너지를 만드는 연로에 비유할 수 있으며,연료 없이 에너지를 생산할 수 없듯이 **데이터가 없으면 기계 학습을 적용할 수 없음**
- 기계 학습을 통해 배웠던 개념을 다음의 실습을 통해 복습해보도록 하겠음

3.1.1 데이터셋 읽기

- **사이킷런 라이브러리**
  - 초보자가 쉽게 기계 학습에 진입할 수 있도록 기본 데이터를 제공함
  - 아주 큰 데이터도 같은 형식으로 표현되기 때문에 iris 를 잘 이해하면 기계 학습이 사용하는 많은 데이터를 이해할 수 있음
  - datasets 클래스는 데이터를 읽어 들이고 데이터 내부를 살펴보는 기능을 제공함

3.1 기계 학습 기초

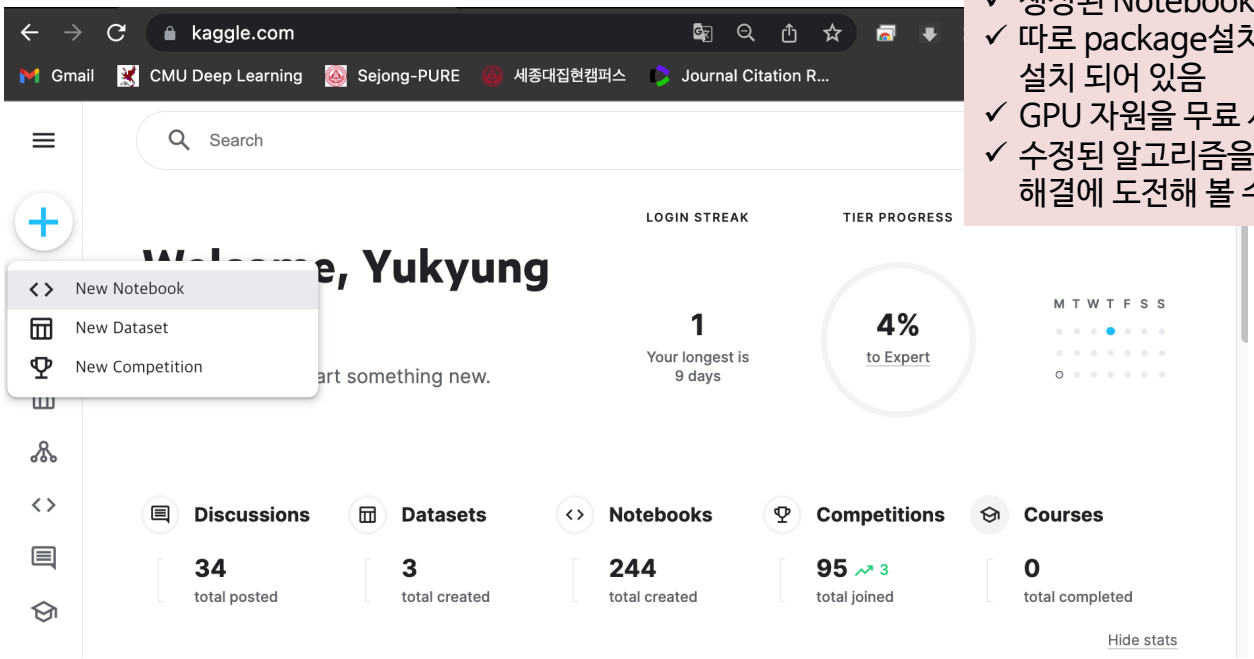
3.1.1 데이터셋 읽기 : iris 데이터셋 읽기

〉〉 실습 3-1. Kaggle 플랫폼을 통한 기계 학습 실습하기

캐글 플랫폼

- 캐글은 데이터 과학 및 머신러닝 경진대회를 주최하는 온라인 커뮤니티
- 캐글 노트북을 활용한 인공지능 공부의 장점 ➔

✓ 생성된 Notebook을 서버에 저장해 줌
✓ 생성된 Notebook을 공유하기 편함
✓ 따로 package설치가 필요하지 않아도 기본 패키지들이 설치 되어 있음
✓ GPU 자원을 무료 사용할 수 있음
✓ 수정된 알고리즘을 바로 commit 하여 인공지능 문제 해결에 도전해 볼 수 있음



3.1 기계 학습 기초

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

3.1.1 데이터셋 읽기 : iris 데이터셋 읽기

〉〉 실습 3-1 (a) iris 데이터셋 읽기

```
1 from sklearn import datasets
2
3 d=datasets.load_iris()
4 print(d.DESCR)
```

01행 sklearn의 datasets 클래스 호출 : scikit-learn 을 import로 불러올 때는 sklearn으로 표기함

03행 load_iris() 라는 함수를 호출해 iris 데이터셋을 읽어들이고 객체 d에 저장
04행 객체 d의 DESCR 이라는 변수는 iris 데이터셋에 대한 설명문을 제공함

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :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

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    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
```

부류별로 50개씩 총 150개의 샘플이 있고 각 샘플은 4개의 변수로 표현됨
4개의 특징 ➔ sepal length, sepal width, petal length, petal width

iris 데이터는 Setosa, Versicolor, Virginica의 세부류로 구성됨

기계 학습에서는 각 변수를 특징 feature 이라 부르며, 샘플 하나에 여러 개의 특징이 있으므로 이를 특징 벡터 feature vector 라 부름
** [참고] 머신러닝에서 Attribute와 Feature는 같은 용어로 이용됨

← 특징 벡터에 있는 특징별로 간단한 통계량을 보여주고 있음

3.1 기계 학습 기초

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

3.1.1 데이터셋 읽기 : iris 데이터셋 읽기

〉〉 실습 3-1 (b) iris 내용 살펴보기

```
# 샘플을 순서대로 출력

for i in range(0,len(d.data)):
    print(i+1, d.data[i],d.target[i])
```

d.data [i] 는 iris의 i번째 샘플의 특징 벡터
d.target [i] 는 i번째 샘플이 어떤 부류 속하는지 알려주는 정보

기계 학습에서는 분류 정보를 레이블 label 또는 참값 ground truth 라고 부름
➔ 레이블은 일반적으로 해당 데이터에 전문성을 갖춘 사람이 신중하게 붙임

```
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
5 [5.  3.6 1.4 0.2] 0
6 [5.4 3.9 1.7 0.4] 0
7 [4.6 3.4 1.4 0.3] 0
8 [5.  3.4 1.5 0.2] 0
9 [4.4 2.9 1.4 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
55 [6.5 2.8 4.6 1.5] 1
56 [5.7 2.8 4.5 1.3] 1
57 [6.3 3.3 4.7 1.6] 1
58 [4.9 2.4 3.3 1. ] 1
59 [6.6 2.9 4.6 1.3] 1
60 [5.2 2.7 3.9 1.4] 1
```

```
141 [6.7 3.1 5.6 2.4] 2
142 [6.9 3.1 5.1 2.3] 2
143 [5.8 2.7 5.1 1.9] 2
144 [6.8 3.2 5.9 2.3] 2
145 [6.7 3.3 5.7 2.5] 2
146 [6.7 3.  5.2 2.3] 2
147 [6.3 2.5 5.  1.9] 2
148 [6.5 3.  5.2 2. ] 2
149 [6.2 3.4 5.4 2.3] 2
150 [5.9 3.  5.1 1.8] 2
```

3.1 기계 학습 기초

3.1.1 데이터셋 읽기 : 기계 학습에서 데이터셋의 표현

- 하나의 샘플은 d개 특징을 갖는 특징 벡터로 표현되며, d차원 특징 벡터는 다음과 같이 표기

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

- 샘플의 개수, 즉 데이터셋의 크기는 보통 n 으로 표기
- 샘플의 레이블은 y로 표기하며, 부류class의 개수를 c라 하면 y는 0~c-1 중의 한 값 또는 1~c 중의 한 값을 가짐

	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

그림 3-3 일반적인 데이터셋 표현 방법(iris 데이터셋 예시)

iris 데이터 셋은 d=4, n=150 에 해당됨

iris 데이터 셋은 c=3이고, y=0,1,2 중의 한 값을 가짐

3.1 기계 학습 기초

3.1.1 데이터셋 읽기 : 기계 학습에서 데이터셋의 표현

- 기계학습은 때때로 **원핫 코드**one-hot code **로 y**샘플의 레이블 **를 표현**함
- **원핫 코드**는 요소 하나만 이진 벡터이며, iris의 경우 다음과 같이 표현됨

  Setosa ➔ (1,0,0)

  Verisicolor ➔ (0,1,0)

  Virginica ➔ (0,0,1)

- 때때로 **다중 레이블**을 허용하는 경우도 있음
- 예를 들어 영상에 고양이와 쥐가 같이 들어 있을 때, 고양이는 두 번째 부류이고 쥐는 다섯 번째 부류라면 레이블은 (0,1,0,0,1,0,0,…,0)로 표현 가능함

3.1 기계 학습 기초

3.1.2 기계 학습 적용: 모델링과 예측

- **데이터가 준비**[1]되었다면, 기계 학습의 핵심인 모델링과 예측을 수행할 수 있음
- 기계 학습에는 아주 다양한 모델이 존재하기 때문에 먼저 **어떤 모델을 사용할지 결정**[2]해야 함
- **모델 학습**[train,3] 을 마치면 학습된 모델을 이용하여 **예측**[prediction,4] 을 할 수 있음
  - 예측이란 샘플의 부류[class]를 알아내는 작업으로 샘플을 인식한다고 말할 수 있음
- 인식 프로그램을 만드는 목적은 학습을 마친 모델을 현장에 설치하여 사용하는 것이며, 현장에서는 학습에 사용하지 않은 새로운 샘플이 발생하므로, **인식 프로그램은 새로운 데이터에 대한 예측을 잘 수행해야 함**
  - 모델 학습에 사용된 데이터를 학습용 데이터[train set], 예측에 사용되는 새로운 데이터를 테스트 데이터[test set]라고 함

3.1 기계 학습 기초

3.1.2 기계 학습 적용: 모델링과 예측

〉〉 실습 3-1(c). Iris에 기계 학습 적용: 모델링과 예측

```
1 from sklearn import svm
2
3 s=svm.SVC(gamma=0.1, C=10)
4 s.fit(d.data,d.target)
5
6 new_d=[[6.4, 3.2, 6.0, 2.5],[7.1, 3.1, 4.7, 1.35]]
7
8 res=s.predict(new_d)
9 print("새로운 2개 샘플의 부류는", res)
```

```
새로운 2개 샘플의 부류는 [2 1]
```

01행 sklearn 라이브러리가 제공하는 svm 클래스를 불러옴
03행 svm 클래스가 제공하는 분류 모델 SVC support vector classifier 를 생성해 객체 s에 저장
04행 fit 함수를 사용해 모델을 학습. fit함수는 데이터를 가지고 학습을 하는데 이때 사용하는 데이터를 훈련 집합이라고 하며, 특징 벡터와 샘플의 레이블 정보가 필요함

특징 벡터 : d.data, 샘플의 레이블 : d.target

08행 predict 함수를 사용해 새로운 샘플을 예측할 수 있음. 이처럼 예측에 사용한 데이터를 테스트 집합이라고 하며, 레이블 정보 없이 특징 벡터만 존재하면 됨

[노트] 하이퍼 매개변수 설정

〈하이퍼 매개변수hyper parameter〉 란 모델의 동작을 제어하는 데 쓰는 변수이며, 모델의 학습을 시작하기 전에 설정해야 하는데 적절한 값으로 설정해야 좋은 성능을 얻을 수 있음. 최적의 하이퍼 매개변수 값을 자동으로 설정하는 일을 〈하이퍼 매개변수 최적화 hyper parameter optimization〉라 하는데, 이것은 기계 학습의 중요한 주제 중 하나임

3.2 인공지능 제품의 설계와 구현

- 3.1에서 만든 프로그램은 데이터를 읽는 일부터 모델링과 예측까지 모두 수행하는 단순한 코드이지만, 붓꽃을 인식하는 인공지능 프로그램의 핵심적인 기능을 모두 수행함

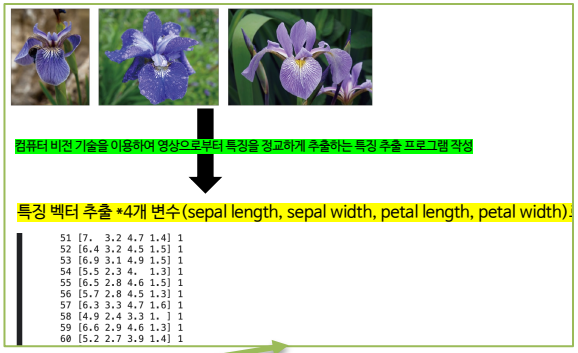
```
from sklearn import datasets

d=datasets.load_iris()
print(d.DESCR)
```

```
# 샘플을 순서대로 출력

for i in range(0,len(d.data)):
    print(i+1, d.data[i],d.target[i])
```

```
from sklearn import svm

s=svm.SVC(gamma=0.1, C=10)
s.fit(d.data,d.target)

new_d=[[6.4, 3.2, 6.0, 2.5],[7.1, 3.1, 4.7, 1.35]]

res=s.predict(new_d)
print("새로운 2개 샘플의 부류는", res)
```

```
새로운 2개 샘플의 부류는 [2 1]
```

3.2 인공지능 제품의 설계와 구현



- 〈촬영된 사진에서 붓꽃의 서식 비율을 알아내는 시스템〉 설계
  - 그림 3-4(a)와 같이, 들판 영상을 카메라로 입력
  - 붓꽃 개체 분할
  - 분할된 개체에서 벡터를 추출
    - 꽃잎의 너비와 길이, 꽃받침의 너비와 길이는 **알고리즘** 컴퓨터비전기술을 통해 추출
  - 특징 4개 변수로 구성된 특징 벡터를 이용해 개체 품종 인식
  - Setosa, Versicolor, Verginica 품종의 서식 비율 자동 분석

** 사실 붓꽃의 서식 비율을 알아내는 시스템은 그다지 쓸모가 없을 수도 있음

(a) 인공지능이 인식할 붓꽃

(b) 과일의 등급 인식 인공지능

(c) 딸기 따는 로봇

그림 3-4 인공지능 제품

3.2 인공지능 제품의 설계와 구현

- 〈과일의 등급을 자동 분류하는 인공지능 시스템〉

  ** 사람의 노동력을 크게 줄여주며 사람이 범할 수 있는 오류도 줄여줄 수 있음

- 〈잘 익은 딸기를 인식하고 손으로 따서 바구니에 담는 로봇 시스템〉

  ** 스마트 팜에서 일하고 있음

(a) 인공지능이 인식할 붓꽃

(b) 과일의 등급 인식 인공지능

(c) 딸기 따는 로봇

그림 3-4 인공지능 제품

3.2 인공지능 제품의 설계와 구현

3.2.1 인공지능 설계 사례 : 과일 등급을 분류하는 기계

(질문) 사과를 상/중/하 세 부류로 분류하는 인공지능 기계를 어떻게 만들 수 있을까?

3.2 인공지능 제품의 설계와 구현

3.2.1 인공지능 설계 사례 : 과일 등급을 분류하는 기계

(질문) 사과를 상/중/하 세 부류로 분류하는 인공지능 기계를 어떻게 만들 수 있을까?

(답변)

(1) 데이터 확보 : 상/중/하의 비율이 비슷하도록 사과를 수천 개 수집

- 이 때 여러 농장에서 골고루 수집해 데이터의 다양성을 확보하는 것이 중요
- 만약 한두 농장에서만 수집하면 데이터 수집과정에서 데이터 편향 data bias 이 발생하여 분류 성능이 떨어질 가능성이 있음

[노트] 데이터 편향

데이터 편향은 다양한 형태로 발생함. 예를 들어 필기 숫자 데이터셋을 만들 때 편의상 대학생을 대상으로 수집했다면 정자체에 가까운 샘플의 비율이 높을 수 있음. 은행 창구에서 발생하는 전표나 우편 봉투에 쓰인 우편번호에서 수집하면 데이터 편향을 크게 줄일 수 있음.

3.2 인공지능 제품의 설계와 구현

3.2.1 인공지능 설계 사례 : 과일 등급을 분류하는 기계

(질문) 사과를 상/중/하 세 부류로 분류하는 인공지능 기계를 어떻게 만들 수 있을까?

(답변)

(1) 데이터 확보 : 상/중/하의 비율이 비슷하도록 사과를 수천 개 수집

- 이 때 여러 농장에서 골고루 수집해 데이터의 다양성을 확보하는 것이 중요
- 만약 한두 농장에서만 수집하면 데이터 수집과정에서 데이터 편향 data bias 이 발생하여 분류 성능이 떨어질 가능성이 있음

(2) 특징 벡터 정의 및 추출 및 레이블링 : 모델 학습을 위한 두 가지 데이터(특징 벡터와 레이블) 확보

- 수집한 사과를 카메라로 촬영해 특징 벡터와 레이블을 파일에 저장
- 특징 벡터는 분별력 높은 특징 discriminative power 을 선정해야 함

  예. 크기를 나타내는 〈지름〉, 익은 정도를 나타내는 〈색깔〉, 〈표면의 균일도〉
- 컴퓨터 비전 기술을 이용하여 영상으로부터 특징을 정교하게 추출하는 특징 추출 프로그램 작성
- 레이블링은 사과 분류하는 일을 오래한 전문가에게 상/중/하로 구분해 달라고 요청

(3) 모델 선정 및 학습

(4) 예측 : 새로운 사과가 기계에 들어오면 상/중/하 세 부류 중 하나로 분류

3.2 인공지능 제품의 설계와 구현

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

〉〉 규칙기반 Rule based method

- 규칙 기반에서는 분류 규칙을 사람이 구현함
  - 예를 들어 iris 데이터를 자세하게 분석해 “꽃잎의 길이가 a보다 크고, 꽃잎의 너비가 b보다 작으면 setosa”와 같은 분류 규칙을 여러 개 만들어 파이썬으로 코딩
- 실용적인 시스템을 만드는 경우 모든 샘플에 대한 경우의 수를 규칙 기반으로 사람이 구현하는 일은 불가능에 가깝고, 작은 데이터라도 규칙에 있는 a, b의 임계치를 정하는 일은 오류가 발생할 가능성이 높으며, 데이터가 바뀌면 규칙을 처음부터 새로 만들어야 함
  - 예를 들어 새로운 붓꽃 품종이 발견되어 부류가 4개가 된다면 같은 일을 처음부터 다시 해야 함

3.2 인공지능 제품의 설계와 구현

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

〉〉 고전적 기계 학습

규칙기반 vs 고전적 기계학습

- 기계 학습의 데이터를 만드는 과정(즉, 특징 벡터를 추출하고 레이블을 붙이는 과정) 규칙 기반과 같음
- **기계 학습은 〈규칙을 만드는 일〉을 모델 학습을 통해 자동으로 수행**
- 기계 학습에서는 규칙을 만드는 일을 **모델링**이라고 함 (예. SVM모델을 사용한 모델링)
- 품종이 하나 늘어 4개가 되어도 **데이터를 만드는 과정만 다시 수행**하고 나머지는 그대로 적용하면 됨

딥러닝 vs 고전적 기계학습

- 기계 학습을 〈고전적 기계학습〉 이라 지칭한 이유는 딥러닝과 구분하기 위해서임
- 현재 아주 많은 인공지능 제품을 딥러닝으로 구현하는데, 딥러닝은 기계 학습의 일종임
- 딥러닝 이전의 기계 학습 기법을 구분하기 위해 고전적 기계 학습이라는 용어사용

3.2 인공지능 제품의 설계와 구현

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

〉〉 딥러닝

- 데이터를 준비하는 과정이 규칙기반과 고전적 기계 학습에 비해 훨씬 쉬움
- 데이터는 특징 벡터와 레이블로 구성되며, 레이블은 동일하게 전문가의 손을 거쳐 만들어짐
- 딥러닝 특징 벡터는 학습 과정에서 자동으로 알아내기 때문에 특징 추출하는 프로그램을 별도로 작성하지 않고, 영상 자체를 입력하면 학습 알고리즘이 자동으로 최적의 특징을 추출함
- 딥러닝은 여러가지 장점이 있음
  - 학습 알고리즘이 특징 추출과 분류를 동시에 최적화 하므로 성능이 뛰어남
    - 대표적인 인공지능 제품인 음성 인식기, 영상 인식기, 언어 번역기 등은 최근 딥러닝의 원리를 통해 뛰어난 성능을 확보함
  - 인공지능 제품을 빨리 만들 수 있음
    - 예시1) 영어 음성 인식기를 한글 음성 인식기로 확장하려면 한국 사람이 말하는 소리 데이터만 새로 확보하면 됨
    - 예시2) 붓꽃 인식기를 장미 인식기로 확장하려면 장미 영상을 새롭게 확보하면 됨

3.2 인공지능 제품의 설계와 구현

3.2.2 규칙 기반 vs. 고전적 기계 학습 vs. 딥러닝

〉〉 용어 정의

- 수작업 특징 hand-crafted feature
  - 규칙 기반과 고전적 기계 학습에서는 특징을 사람이 설계하거나 추출하기 때문에 이를 수작업 특징이라 부름
- 특징 학습 feature learning 또는 표현 학습 representation learning
  - 딥러닝과 같이 특징을 자동으로 학습하는 과정을 특징 학습 또는 표현 학습이라고 부름

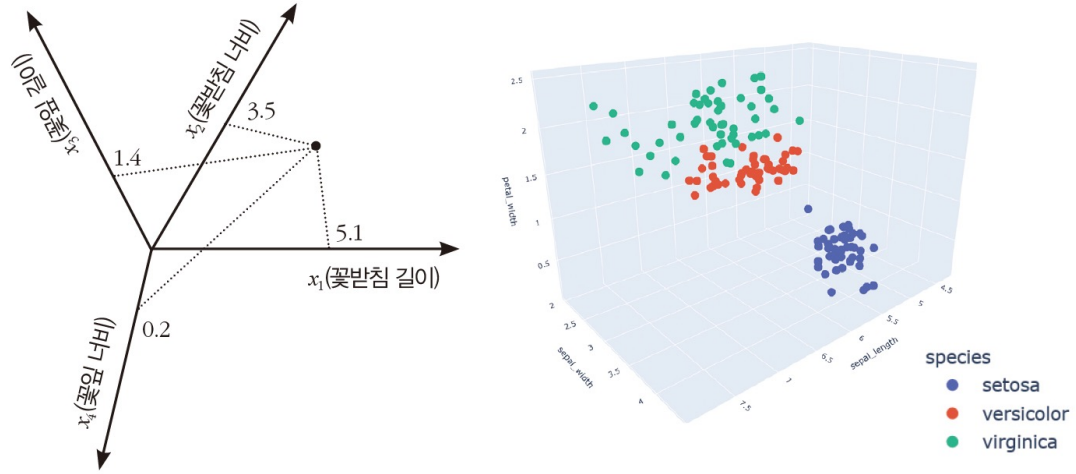
[노트] 원샷 학습 1-shot learning**, 퓨샷 학습** few-shot learning**, 준지도 학습** semi-supervised learning

레이블을 붙이는 작업은 전문가가 해야 하므로 비용이 많이 듬. 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 활용함

3.3 데이터에 대한 이해

3.3.1 특징 공간에서 데이터 분포

- Iris 데이터는 특징이 4개 이므로 4차원 특징 벡터로 표현되고, 4차원 특징 공간feature space을 구성함
- iris 데이터에 있는 150개의 샘플은 각각 4차원 공간의 한 점에 해당됨
- 첫 번째 샘플인 (5.1, 3.5, 1.4, 0.2)를 가상공간의 4차원 공간에 그림을 그리면 왼쪽 그림과 같음
  - 가상의 공간이라고 한 이유는 우리가 사는 세상이 3차원이어서 4차원 이상은 실제 그림으로 그릴 수 없기 때문



(a) 4차원 특징 공간(가상의 그림)

(b) 꽃잎 길이 축을 제외한 3차원 특징 공간

그림 3-5 iris 데이터를 특징 공간에 그리기

3.3 데이터에 대한 이해

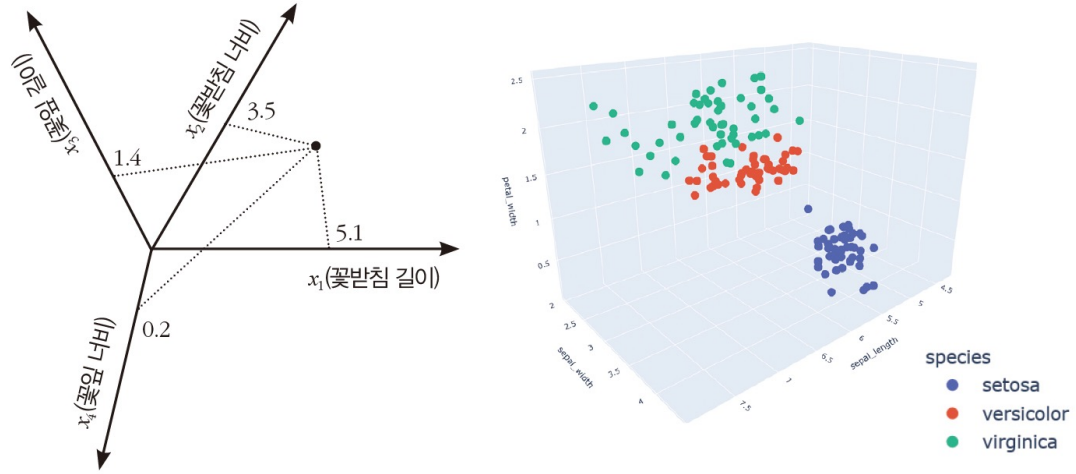
3.3.1 특징 공간에서 데이터 분포

- 오른쪽 그림에 있는 데이터의 분포를 자세히 살펴보면, 수직 축에 해당하는 꽃잎너비 즉 petal width 특징에 대해 Setosa는 아래쪽에 Virginica는 위쪽에 분포하며, Setosa의 petal width 특징 값이 다른 부류에 비해 작음

  ➔ 꽃잎 너비라는 특징은 세 부류를 구별하는 능력 즉, 분별력이 높은 편임

- 꽃받침 너비인 sepal width 특징은 세 부류가 많이 겹침

  ➔ 꽃받침 너비라는 특징은 세 부류를 구별하는 능력 즉, 분별력이 약한 편임



(a) 4차원 특징 공간(가상의 그림) (b) 꽃잎 길이 축을 제외한 3차원 특징 공간

그림 3-5 iris 데이터를 특징 공간에 그리기

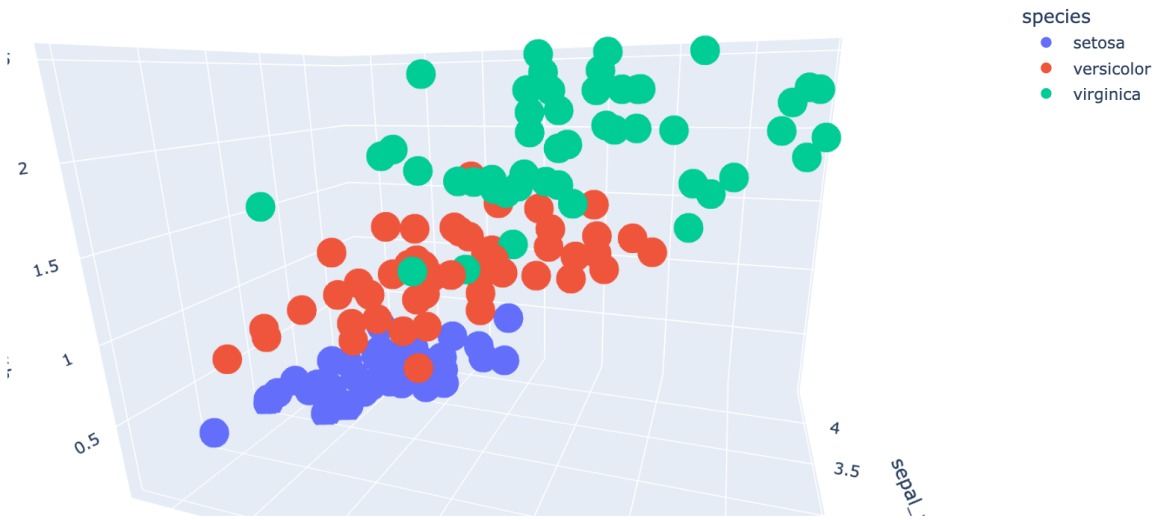
3.3 데이터에 대한 이해

3.3.1 특징 공간에서 데이터 분포

- 〉〉 실습 3-2. iris 데이터의 분포를 특징 공간에 그리기

```
1 import plotly.express as px
2
3 df = px.data.iris()
4 fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_width', color='species')
5 fig.show()
```

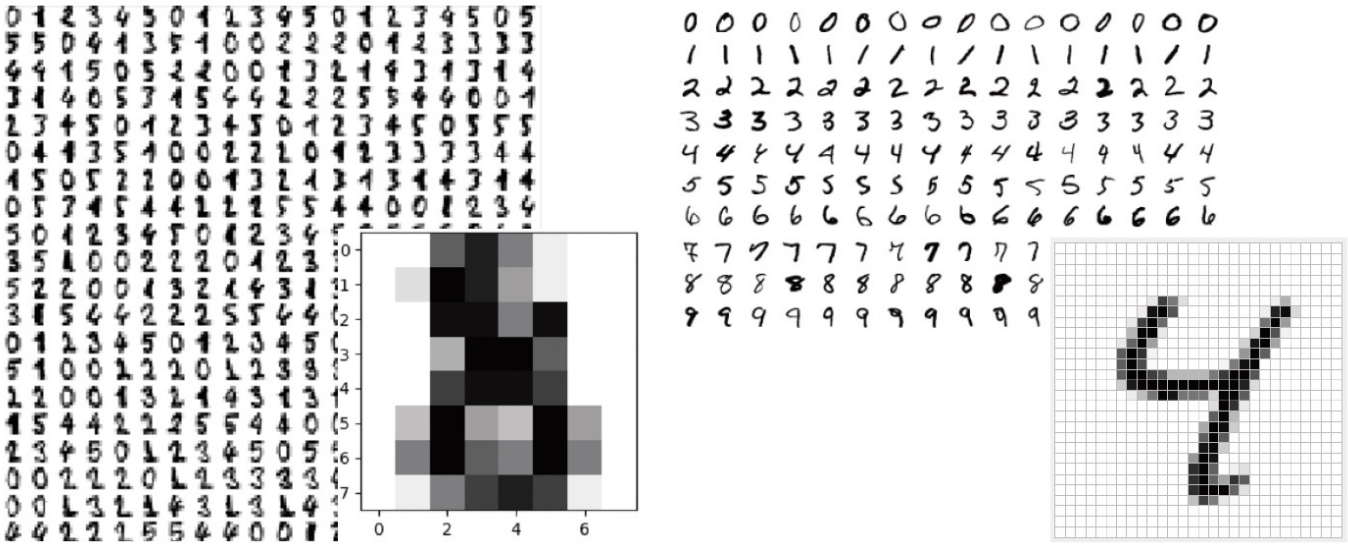
Plotly는 대화식으로 작동하는 그림을 그릴 때 사용하는 라이브러리



3.3 데이터에 대한 이해

3.3.2 영상 데이터 사례 : 필기 숫자

- (왼쪽 영상) sklearn 라이브러리에서 기본으로 제공하는 필기 숫자 데이터셋 샘플임
- (오른쪽 영상) 미국 국립표준기술연구소에서 미국인을 대상으로 수집한 필기 숫자 데이터셋 샘플임



(a) sklearn에서 제공하는 데이터셋

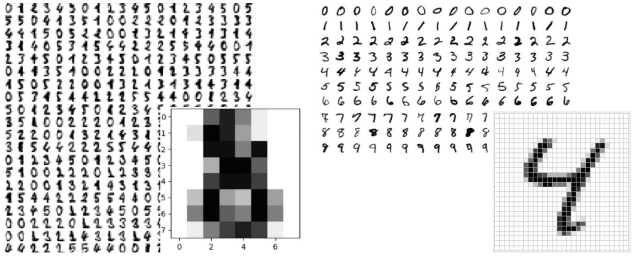
(b) MNIST 데이터셋

그림 3-6 필기 숫자 데이터셋

3.3 데이터에 대한 이해

3.3.2 영상 데이터 사례 : 필기 숫자

- (왼쪽 영상) sklearn 라이브러리에서 기본으로 제공하는 필기 숫자 데이터셋 샘플임
  - 8x8 맵으로 표현되며 총 1797자가 포함되어 있음
  - Sklearn 데이터는 원래 32x32 맵인데 영상 처리 프로그램으로 8x8로 축소해 저장한 것
  - 샘플 하나가 64개 화소로 구성되어 있고, 한 화소는 [0,16] 사이의 명암값으로 표현됨
- (오른쪽 영상) 미국 국립표준기술연구소에서 미국인을 대상으로 수집한 필기 숫자 데이터셋 샘플임
  - MNIST 데이터셋이라고 불리움
  - 총 7만 자가 포함되어 있고, 숫자는 0~9의 10개 부류이며, 부류당 7천 개의 샘플이 존재함
  - 샘플은 28x28 맵으로 표현되어 있으며, 샘플 하나가 총 784개의 화소로 구성되어 있음
  - 한 화소는 [0,255] 사이의 명암 값으로 표현됨



(a) sklearn에서 제공하는 데이터셋 (b) MNIST 데이터셋

그림 3-6 필기 숫자 데이터셋

3.3 데이터에 대한 이해

3.3.2 영상 데이터 사례 : 필기 숫자

- 〉〉 실습 3-3(a). 필기 숫자 데이터

[10]:

```
1  from sklearn import datasets
2  import matplotlib.pyplot as plt
3
4  digit=datasets.load_digits()
5
6  plt.figure(figsize=(5,5))
7  plt.imshow(digit.images[0], cmap=plt.cm.gray_r, interpolation='nearest')
8
9  plt.show()
10 print(digit.data[0])
11 print("이 숫자는 ",digit.target[0],"입니다")
12
```

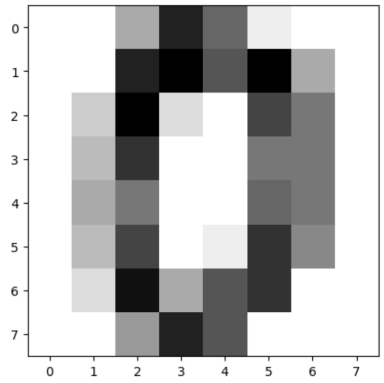
01행 데이터셋을 읽는 데 필요한 datasets 클래스를 불러오고
02행 matplotlib 라이브러리를 불러옴

04행 load_digits 함수로 필기 숫자 데이터셋을 읽어 digit 객체에 저장함

07행 0번 인덱스에 있는 숫자 영상을 imshow 함수로 보여줌

09행 print 함수로 64개 화솟값을 출력하고

10행 이 샘플의 레이블을 출력함



```
[ 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  0 입니다
```

[노트] matplotlib 을 이용한 시각화

파이썬에서 matplotlib 라이브러리는 시각화에 가장 널리 쓰임. 인공지능은 학습 과정이나 예측 결과를 시각화하는 데 matplotlib을 자주 사용함.

3.3 데이터에 대한 이해

3.3.3 영상 데이터 사례: lfw 얼굴 데이터셋

- Lfw 는 유명인의 얼굴 영상을 모아둔 데이터셋
- 5,749명에 대한 13,233장의 영상이 들어 있으며, 영상 한 장은 50x37 맵으로 표현되고 한 화소는 [0,255] 사이의 명암값으로 표현됨
- 이 데이터셋에는 사람당 평균 두 장 남짓한 영상이 있기 때문에 얼굴 인식보다는 얼굴 영상 두 장이 주어졌을 때 같은 사람인지 알아내는 얼굴 검증에 주로 사용됨
- Lfw 데이터셋 공식 웹사이트에는 데이터셋 편향에 대한 인정과 함께 섣불리 상업용 얼굴 인식 프로그램을 제작하는데 활용하면 안된다는 주의사항이 명시되어 있음
- 어린이 사진이 매우 적고, 갓난아기는 아예 없으며, 80세 이상이 매우 적고, 여성이 상대적으로 적으며, 아예 없거나 매우 희소한 인종이 많다고 설명되어 있음. 이런 편향은 유명인 위주로 사진을 모았기 때문에 발상한 것임

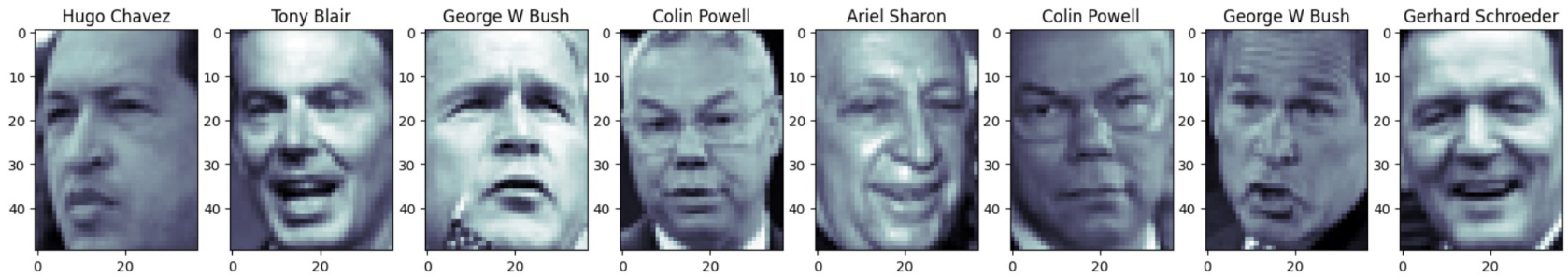
3.3 데이터에 대한 이해

3.3.3 영상 데이터 사례: lfw 얼굴 데이터셋

- 〉〉 실습 3-3(b). lfw 유명인 얼굴 데이터셋

[13]:

```
lfw = datasets.fetch_lfw_people(min_faces_per_person=70, resize=0.4)

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

for i in range(8):
    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()
```



3.3 데이터에 대한 이해

3.3.4 텍스트 데이터 사례: 20newsgroups

- 20newsgroups 데이터셋은 웹에서 수집한 문서를 20개 부류로 구분한 텍스트를 담고 있음
- 20newsgroups 데이터셋은 부류 정보를 가지고 있으므로 분류 문제에 해당됨
- 그런데 앞에서 살펴본 iris나 숫자 또는 얼굴 영상과는 성질이 판이하게 다름
- Iris와 영상 데이터의 샘플이 같은 크기로 표현되는 것과 달리 문서는 가변 길이임
- 또한 문서는 단어가 나타나는 순서가 매우 중요한 시계열 데이터에 속함

3.3 데이터에 대한 이해

3.3.4 텍스트 데이터 사례: 20newsgroups

- 〉〉 실습 3-3(c). 20newsgroups 데이터셋 데이터셋을 읽고 0번째 문서의 내용과 부류 정보를 출력

[14]:

```
1 news=datasets.fetch_20newsgroups(subset='train')
2 print("*****\n", news.data[0], "\n*****")
3 print("이 문서의 부류는 <",news.target_names[news.target[0]] , "> 입니다.")
```

03행 target_names는 숫자로 표시된 부류 정보를 문자열로 변환해줌

```
*****
 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.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----



*****
이 문서의 부류는 < rec.autos > 입니다.
```

0번째 문서는 이메일에서 수집하였고, 부류는 rec.autos로서 자동차에 관한 내용임

3.4 특징 추출과 표현

- 기계학습의 전형적인 과정

그림 3-7 기계 학습의 과정

- 고전적인 기계 학습을 통한 붓꽃 분류기 제작
  - Iris 데이터의 경우 채집한 붓꽃의 꽃잎과 꽃받침 크기를 사람이 직접 재서 기록했음
    - 데이터 수집과 특징 추출을 사람이 직접 수행한 셈임
  - 고전적인 기계 학습을 사용한다면 분별력 높은 특징 추출을 사람이 설계하고 구현함

- 딥러닝을 통한 사과 등급 분류기 제작
  - 사람은 사과 영상을 수집하고 등급 레이블을 붙이는 것이 전부임
  - 딥러닝은 특징 추출과 모델링을 동시에 최적화함

3.4 특징 추출과 표현

3.4.1특징의 분별력

- 사람은 직관적으로 분별력이 높은 특징을 선택해 사용함
- (질문) 텀블러 중에서 왼쪽 것을 고른 후 친구에게 알리려면 어떤 특징을 들어 설명해야 할까?
- (답변1) “글씨가 가로 방향으로 쓰여 있는 것”, “몸통 색이 더 하얀 것”, “손잡이가 있는 것”, “뚜껑이 있는 것” 등 으로 설명 가능할 것으로 생각 됨
- (답변2) 기계 학습 방식으로 말하면, 글씨 방향, 몸통 색깔, 손잡이 유무, 뚜껑 유무를 특징으로 사용

그림 3-8 텀블러를 구분하기 위한 특징으로 무엇이 좋을까?

3.4 특징 추출과 표현

3.4.1특징의 분별력

- 두 텀블러 뚜껑 유무라는 특징은 분별력이 전혀 없음
  - 두 텀블러 모두 뚜껑이 있음
- 손잡이 유무라는 특징을 선택하면 분별력을 높일 수 있음
  - 눈에 가장 잘 띄고, 조명의 변화에 영향을 덜 받기 때문

➔ 기계 학습도 사람처럼 분별력이 높은 특징을 추출해야 함

그림 3-8 텀블러를 구분하기 위한 특징으로 무엇이 좋을까?

3.4 특징 추출과 표현

3.4.1특징의 분별력

- 다양한 형태의 특징 공간
  - (c) (d) 는 일정한 양의 오류를 허용해야 하는 복잡한 상황이며, 우리가 사는 세상은 이와 같이 오류를 허용할 수 밖에 없는 데이터를 생성함
  - 데이터의 원천적인 성질이 그럴 수도 있고, 데이터 수집 과정에서 사람이 측정이나 레이블링을 잘못해서, 또는 특징 설계를 잘못해서 (c) (d) 처럼 분포되었을 수 있음
  - 기계 학습은 (c) (d)와 같은 데이터를 처리해야 하며, 실습에서 사용하는 데이터는 (c) (d)에 해당함
  - 기계 학습에서 특징 추출 알고리즘이 해야 하는 일은 가급적 (d)와 같은 상황은 피하고 (c)와 같은 상황을 만들어 내는 것
  - **딥러닝이 고전적이 기계 학습보다 뛰어난 점을 감안한다면 딥러닝은 (d) 보다 (c)에 가까운 특징 공간을 형성한다고 볼 수 있음**

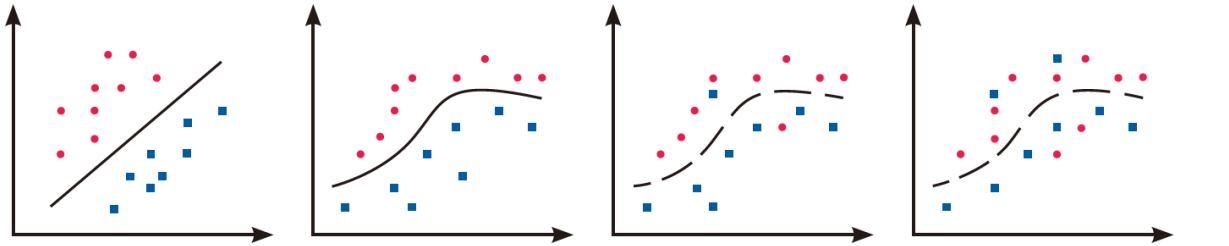


그림 3-9 특징의 분별력

3.4 특징 추출과 표현

3.4.2 특징 값의 종류 : 수치형, 범주형 (순서형, 이름형)

- Iris 데이터가 사용하는 특징은 꽃잎과 꽃받침의 길이와 너비를 나타내는 실수이며, 이 특징들은 거리 개념이 있어, 이런 특징을 수치형 특징numerical feature 이라고 함
  - 사과 등급 분류 문제에서 사과의 크기, 색상의 균일도 등도 수치형이며, 수치형은 실수일 수도 있고 정수일 수도 있음.
  - 또한 손잡이 유무와 같이 참/거짓의 이진 값일 수도 있음

- 범주형 특징categorical feature 도 있으며, 학점, 수능 등급, 혈액형, 지역 등이 범주형에 속함
  - 범주형은 크게 순서형과 이름형으로 나뉘는데, 학점과 수능 등급은 순서형이고 혈액형과 지역은 이름형임
  - 순서형은 거리 개념이 있음. 예를 들어 학점의 경우 A+~F가 있는데, A+은 B보다 B+에 가까움. 순서형 특징의 경우 순서대로 정수를 부여하면 수치형으로 취급할 수 있음
  - 이름형은 거리 개념이 없음. 혈액형에는 A, B, O, AB형이 있는데 A형은 AB형보다 O형에 가깝다는 말은 성립하지 않음. 이름형은 보통 원핫 코드로 표현함

3.5 필기 숫자 인식

- 기계 학습 절차를 따라 필기 숫자 데이터셋으로 프로그래밍 실습을 진행함
  - 특징 추출을 위한 코드 작성, sklearn 라이브러리를 이용하여 모델 학습 및 예측
  - 기계 학습 모델로는 SVM을 사용

- 화솟값을 특징으로 사용
  - **(질문) 필기 숫자를 인식하려면 어떤 특징을 써야 할까?**
  - (답변) 가장 단순한 방법은 화소 각각을 특징으로 간주하는 것.
    - 샘플 하나는 8x8 맵으로 표현되므로 64개의 특징이 있고, 결국 64차원의 특징 벡터로 표현
    - 2차원 구조 즉 **행렬 형태**로 표현된 패턴을 1차원 구조 즉 **벡터 형태**로 변환
    - 이 때 행 우선 이어 붙이기 방식을 사용 (0행 뒤에 1행, 1행 뒤에 2행을 이어 붙이는 방식)

- 붓꽃 인식기 vs. 필기 숫자 인식기
  - iris 데이터셋 : 4차원의 특징 벡터 150개
  - digit 데이터셋: 64차원의 특징 벡터 1796개

  ➔ 인식기 프로그램은 동일하고, 데이터만 다르게 사용!!

3.5 필기 숫자 인식

- 붓꽃 인식기 vs. 필기 숫자 인식기
  - iris 데이터셋 : 4차원의 특징 벡터 150개
  - digit 데이터셋: 64차원의 특징 벡터 1796개

  ➔ 인식기 프로그램은 동일하고, 데이터만 다르게 사용!!

[노트] 컴퓨터 프로그래밍에 쓰이는 패턴의 중요성

[프로그램 3-1: 붓꽃 인식기]과 [프로그램 3-4: 필기 숫자 인식기]은 코드의 많은 부분을 공유함. 여기에서 알 수 있는 것 처럼 컴퓨터 프로그래밍에도 패턴이 있음. 이 패턴을 잘 기억하고 따라 하는 것은 좋은 프로그래머로 성장하는 지름길임

3.5 필기 숫자 인식

- 붓꽃 인식기 vs. 필기 숫자 인식기
  - iris 데이터셋 : 4차원의 특징 벡터 150개
  - digit 데이터셋: 64차원의 특징 벡터 1796개

➔ **인식기 프로그램은 동일하고, 데이터만 다르게 사용!!**

```
from sklearn import datasets

d=datasets.load_iris()
print(d.DESCR)
```

```
from sklearn import svm

s=svm.SVC(gamma=0.1, C=10)
s.fit(d.data,d.target)

new_d=[[6.4, 3.2, 6.0, 2.5],[7.1, 3.1, 4.7, 1.35]]

res=s.predict(new_d)
print("새로운 2개 샘플의 부류는", res)
```

```
새로운 2개 샘플의 부류는 [2 1]
```

```
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)

# 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])
```

```
예측값은 [0 1 2]
참값은 0 1 2
```

3.5 필기 숫자 인식

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

```python
from sklearn import datasets
from sklearn import svm

digit = datasets.load_digits()
```

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

# 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])
```

```
예측값은 [0 1 2]
참값은 0 1 2
```

+ Code + Markdown

```python
#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, "%")
```

```
화소 특징을 사용했을 때 정확률= 100.0 %
```

https://www.kaggle.com/yukyungchoi/2023-ai-w2-p2

3.6 성능 측정

- 주어진 데이터에서 가장 적합한 기계 학습 모델을 찾는 과정을 **모델 선택** model selection 이라고 함
  - 모델 선택 시 여러 모델의 **성능을 견주어 가장 좋은 것**을 선택해야 하기 때문에 **〈객관적인 성능 측정〉**은 매우 중요함
  - 모델 선택 시 사용 가능한 모델 예시
    - SVM, K-NN, 결정 나무 decision tree, 랜덤 포리스트 random forest, 신경망, 딥러닝 등
- 성능 평가에는 적절한 **평가 기준** evaluation metric 이 필요
  - 기계 학습에는 많은 기준이 존재하며, 가장 널리 쓰이는 기준 중 하나는 **정확률** accuracy 임
- 새로운 데이터로 성능을 측정하는 일을 **일반화** generalization **능력 측정**이라고 함
  - 테스트 집합 사용과 교차검증을 통해 일반화 성능 측정 가능함

3.6 성능 측정

3.6.1 혼동 행렬과 성능 측정 기준

- 혼동 행렬confusion matrix 이란 부류별로 옳은 분류와 틀린 분류의 개수를 기록한 행렬이다.

		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

- 혼동 행렬(a)는 부류가 C개인 경우의 표현법, 행에는 예측한 부류를 배치하고 열에는 참값을 배치함
  - n_{ij}는 모델이 i 라고 예측했는데 실제 부류는 j인 샘플의 개수임

3.6 성능 측정

3.6.1 혼동 행렬과 성능 측정 기준

		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

- 혼동 행렬(b)는 부류가 긍정/부정인 경우의 표현법, 긍정과 부정은 상황에 따라 달라짐
  - 보행자 검출의 경우 보행자가 있으면 긍정, 없으면 부정
  - 환자를 찾아내는 것이 목표인 경우 환자 긍정, 정상인 부정
  - 생산라인에서 불량품을 찾아내는 경우 불량품 긍정, 정상품 부정
  - 긍정을 긍정으로 예측하면 참 긍정 (TP True Positive), 긍정을 부정으로 잘못 예측하면 거짓 부정 (FN False Negative), 부정을 긍정으로 잘못 예측하면 거짓 긍정 (FP False Positive), 부정을 부정으로 예측하면 참 부정 (TN True Negative)

3.6 성능 측정

3.6.1 혼동 행렬과 성능 측정 기준

- 가장 널리 쓰이는 성능 측정 기준은 정확률 accuracy
- 정확률은 가장 널리 쓰이지만, 2부류 분류 문제에서는 종종 한계를 보임
- 예를 들어 의사가 환자를 진료하는 경우, 정상인이 암환자보다 훨씬 많기 때문에 모두 정상인이라고 진단해도 정확률이 꽤 높음
- 따라서 의사의 환자 진료와 같은 경우에는 특이도specificity와 민감도sensitivity를 성능 기준으로 사용함
- 웹에서 정보 검색을 수행하거나 영상에서 물체 검출을 하는 경우에는 정밀도precision와 재현율recall을 주로 사용함

$$\text{정확률} = \frac{\text{맞힌 샘플수}}{\text{전체 샘플수}} = \frac{\text{대각선 샘플수}}{\text{전체 샘플 수}}$$

$$\text{특이도} = \frac{TN}{TN+FP}, \text{민감도} = \frac{TP}{TP+FN} \qquad \text{정밀도} = \frac{TP}{TP+FP}, \text{재현율} = \frac{TP}{TP+FN}$$

3.6 성능 측정

3.6.2 훈련/검증/테스트 집합으로 쪼개기

- 객관적인 성능 측정을 위한 한 가지 좋은 방법은 데이터를 적절한 비율로 훈련 집합train set, 검증 집합validation set, 테스트 집합test set 으로 나누어 사용하는 것

그림 3-11 훈련/검증/테스트 집합으로 쪼개기

- 여러 모델을 학습하고 성능을 비교할 때는 훈련 집합과 검증 집합을 사용함
- 예를 들어 훈련 집합으로 SVM, K-NN, 결정 트리, 랜덤 포리스트, 신경망 모델 등 을 학습하고 검증 집합으로 정확률을 측정하여 정확률이 가장 높은 모델을 최종 선택
- 검증 집합을 통해 선택된 모델을 사용하여 한 번도 사용하지 않은 테스트 집합으로 최종 성능을 측정, 이 성능이 일정 기준을 넘으면 합격 판정을 하고 현장에 사용
- 모델 선택 과정이 제외된 경우에는 검증 집합 없이 훈련 집합과 테스트 집합으로 나누어 사용함

3.6 성능 측정

3.6.2 훈련/검증/테스트 집합으로 쪼개기

〉〉 실습 3-5. 필기 숫자 인식-훈련 집합으로 학습하고 테스트 집합으로 성능 측정 (혼동 행렬, 정확률)

```
1  from sklearn import datasets
2  from sklearn import svm
3  from sklearn.model_selection import train_test_split
4  import numpy as np
5
6
7  # 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
8  np.random.seed(0)
9  digit = datasets.load_digits()
10 x_train, x_test, y_train, y_test = train_test_split(digit.data, digit.target, train_size=0.6)
11
12
13 # svm의 분류 모델 SVC를 학습
14 s=svm.SVC(gamma=0.001)
15 s.fit(x_train, y_train)
16 res = s.predict(x_test)
17
18
19 # 혼동 행렬 구함
20 conf=np.zeros((10,10))
21 for i in range(len(res)):
22     conf[res[i]][y_test[i]]+=1
23 print(conf)
24
25
26 # 정확률 측정하고 출력
27 no_correct=0
28 for i in range(10):
29     no_correct+=conf[i][i]
30 accuracy=no_correct/len(res)
31 print("테스트 집합에 대한 정확률은",accuracy*100, "%입니다.")
```

03행 model_selection 클래스가 제공하는 train_test_split 함수를 불러옴

08행 난수 생성 고정을 위해 사용 (train_test_split 함수가 난수를 사용)

10행 train_test_split함수를 이용하여 digit 데이터를 훈련 집합과 테스트 집합으로 분할. 첫 번째와 두 번째 매개변수는 분할할 데이터의 특징 벡터와 레이블이며, 세 번째 매개변수는 훈련 집합의 비율

14~15행 훈련 집합으로 SVM을 학습

16행 테스트 집합으로 예측을 수행

20~22행 혼동 행렬 계산

27~30행 정확률을 계산

```
[[60.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 73.  1.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0. 69.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 70.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 63.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 87.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1. 76.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0. 65.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 77.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0. 74.]]
테스트 집합에 대한 정확률은 99.30458970792768 %입니다.
```

3.6 성능 측정

3.6.3 교차 검증

- [실습 3-5]에서는 digit 데이터를 6:4 비율로 훈련 집합과 테스트 집합으로 나누고 정확률을 측정함
- 그런데 랜덤하게 샘플링을 진행하므로 우연히 높은 정확률을 얻게 데이터가 분할될 수 있고 낮은 정확률을 얻게 분할될 수 있음
- 이런 우연성을 줄이는 방법은 여러 번 분할해 각각의 성능을 측정하고 평균을 구하는 것

- K-겹 교차 검증은 그림에서 보여주는 것처럼 훈련 집합을 같은 크기로 부분집합 K개로 나눔 (예. K=5)
- (그림(a)) 첫 번째 시도에서는 첫 번째 부분집합을 남겨두고 나머지 4개로 모델을 학습함. 그리고 남겨둔 첫 번째 검증 집합을 사용하여 테스트 집합으로 간주하고 성능을 측정. 이 과정을 실험5 까지 반복 후 5개의 정확률 평균을 이용하여 최종 모델 선정
- (그림(b)) 그림(a)와 동일하나 5개의 정확률 평균이 모델의 성능을 나타냄

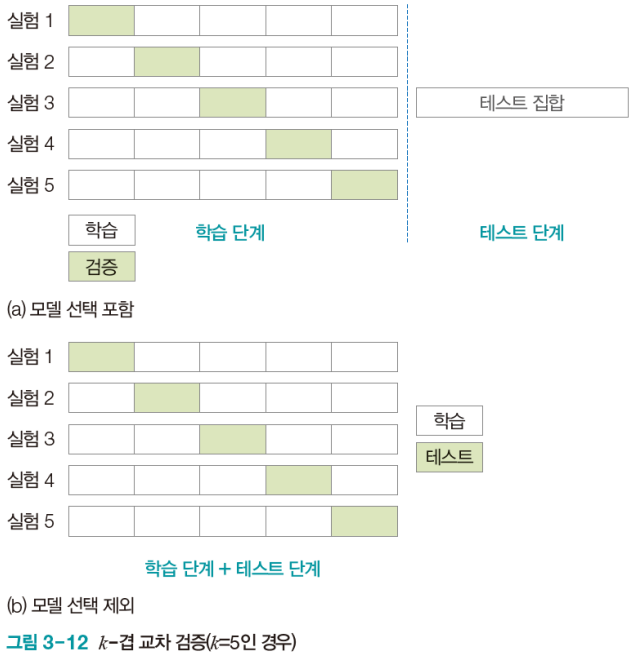


그림 3-12 k-겹 교차 검증(k=5인 경우)

3.6 성능 측정

3.6.3 교차 검증

〉〉 실습 3-6. 필기 숫자 인식-교차 검증으로 성능 측정 ← 모델 선택은 제외하므로 그림 3-12(b)에 해당

```
1  from sklearn import datasets
2  from sklearn import svm
3  from sklearn.model_selection import cross_val_score
4  import numpy as np
5
6  digit=datasets.load_digits()
7  s=svm.SVC(gamma=0.001)
8  accuracies=cross_val_score(s,digit.data,digit.target,cv=5) #5-겹 교차 검증
9
10 print(accuracies)
11 print("정확률(평균)=%0.3f, 표준편차=%0.3f"%(accuracies.mean()*100,accuracies.std()))
```

03행 교차 검증을 대행해주는 sklearn의 cross_val_score 함수를 불러옴

08행 교차 검증 적용, 첫 번째 매개변수는 분류기 모델, 두 번째와 세 번째 매개변수는 각각 특징 벡터와 레이블 정보, 네 번째 매개변수는 교차 검증 부분 집합의 수 (k)

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

5번의 교차검증으로 얻은 정확률을 살펴보면 상당히 들쭉날쭉한 것을 확인 가능함
따라서, 한번만 시도하여 의사결정을 하는 것은 위험함

- 교차 검증은 성능 측정 결과에 대한 신뢰도를 높일 목적으로 수행함
- K 를 더 크게 하면, 매번 더 큰 훈련 집합을 사용할 뿐만 아니라 더 많은 수의 성능을 평균하므로 신뢰도가 높아짐. 단, 실행 시간이 더 걸리므로 상황에 따라 적절한 값을 설정해야 함
- 보통 K=5 혹은 10을 사용함

3.7 인공지능은 어떻게 인식을 하나?

- 지금까지 필기 숫자 패턴을 인식하는 인공지능 시스템을 제작하였음
- 이런 실습을 통해 기계 학습의 절차를 이해하고 기계 학습 모델(예.SVM)을 경험했음
- 하지만, 기계 학습 모델(예.SVM) 동작 원리를 전혀 이해하지 못한 채 블랙박스로 보고 사용했음
- 동작 원리를 이해하고 사용하는 것과 모른 채 사용하는 것의 차이는 생각 이상으로 큼
- 모르고 사용하면 언젠가 한계가 드러남
- 따라서, 인공지능 시스템 제작을 위해 새로운 기계 학습 모델을 적용하는 경우 별도의 시간을 내어 모델의 동작 원리도 함께 이해하려는 노력을 기울이기 바람

➔ 필요시 기계학습 수업에서 배운 내용을 복습하는 것도 좋음