머신러닝의 기초

머신러닝이란?

- 머신러닝(machine learning)
- 인공지능의 한 분야. 1959년 아서 사무엘이 처음 용어를 만들었으며.
- 컴퓨터에 학습 기능을 부여하기 위한 연구 분야
- 머신러닝이 가능하다면 명시적으로 프로그램을 작성하지 않아도 컴퓨터 스스로 데이터에서 학습하여 적절한 결정을 내릴 수 있다

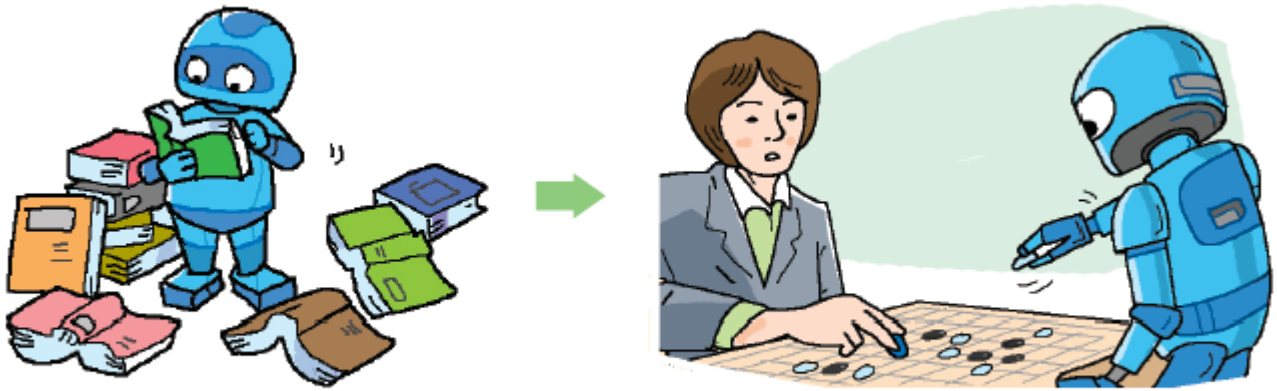
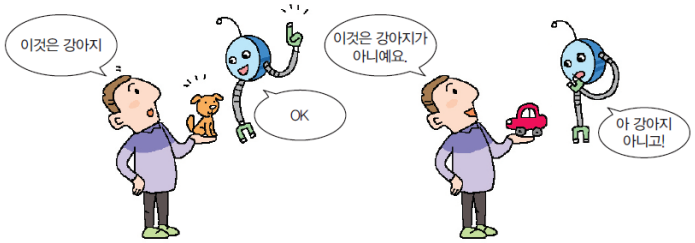


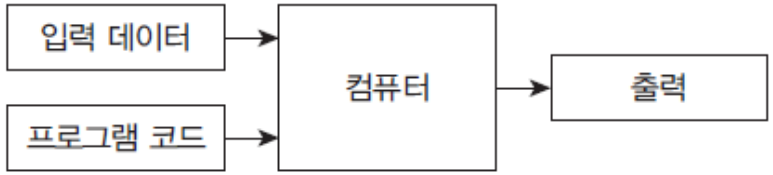
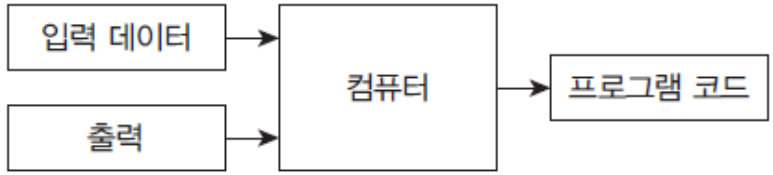
그림 3-1 머신러닝

머신러닝과 전통적인 프로그래밍과의 차이점

- 전통적인 프로그램
  - 문제를 해결하는 알고리즘을 고안한 후에 이것을 프로그램으로 변환하여서 컴퓨터로 실행
  - 예. 2개의 수 중에서 큰 수를 찾는 알고리즘
  - 한계가 있다. 강아지와 고양이를 구별하는 문제
- 머신러닝
  - 많은 수의 사진을 사진을 제공하면 마신러닝 시스템이 스스로 사진에서 어떤 패턴을 발견하여 강아지를 인식
  - 문제를 해결하기 위해 알고리즘을 개발할 필요없이 데이터만 제공



머신러닝과 전통적인 프로그래밍과의 차이점

전통적인 프로그래밍	머신러닝
	
프로그램을 개발하여 컴퓨터로 입력 데이터를 처리하면 출력 데이터가 생성된다.	입력 데이터와 출력 데이터를 컴퓨터에 제공하면 컴퓨터가 스스로 프로그램 코드를 생성한다.

인공지능, 머신러닝, 딥러닝

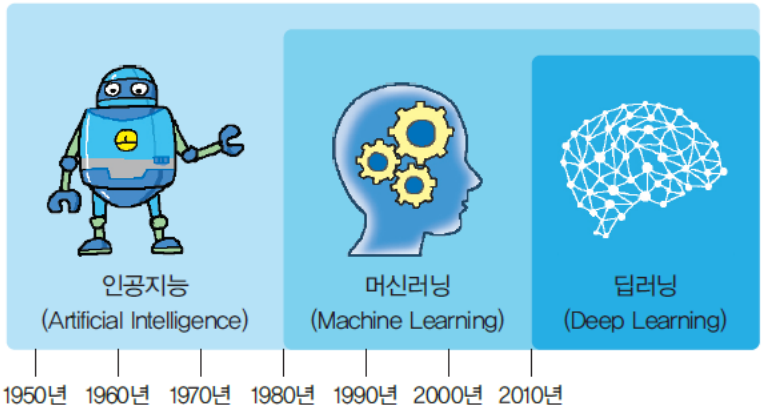


그림 3-4 인공지능, 머신러닝, 딥러닝 간의 관계

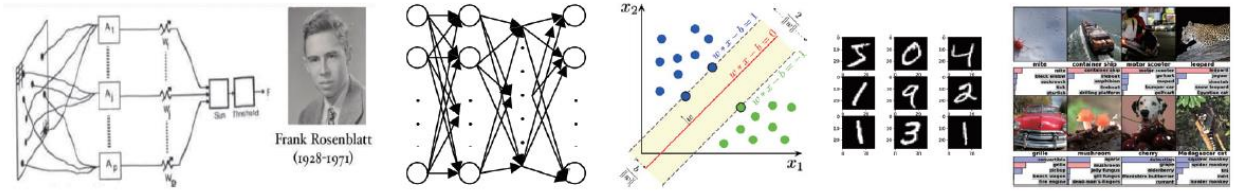
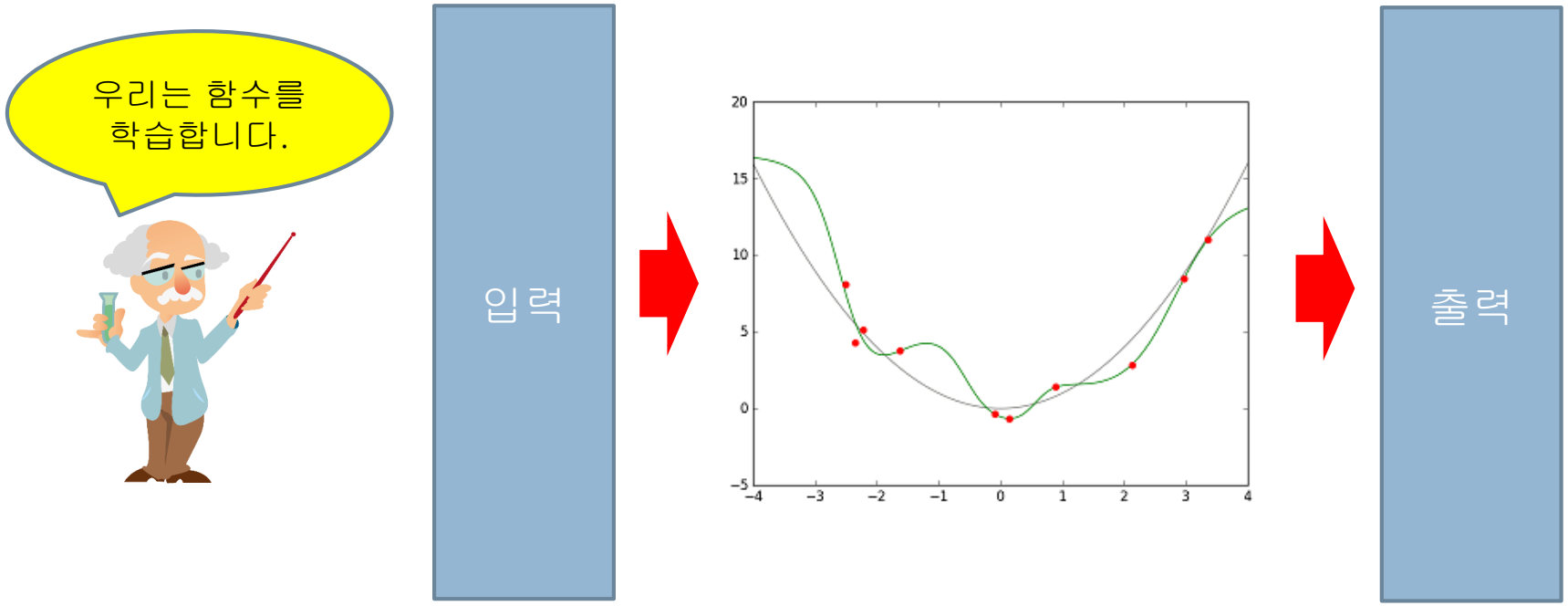


그림 3-5 머신러닝의 역사

머신러닝

- 머신러닝은 입력을 받아서 출력하는 함수 y=f(x)를 학습한다고 생각할 수 있다. (함수 근사)



머신러닝(machine learning) == 함수 근사(function approximation)

머신러닝의 종류

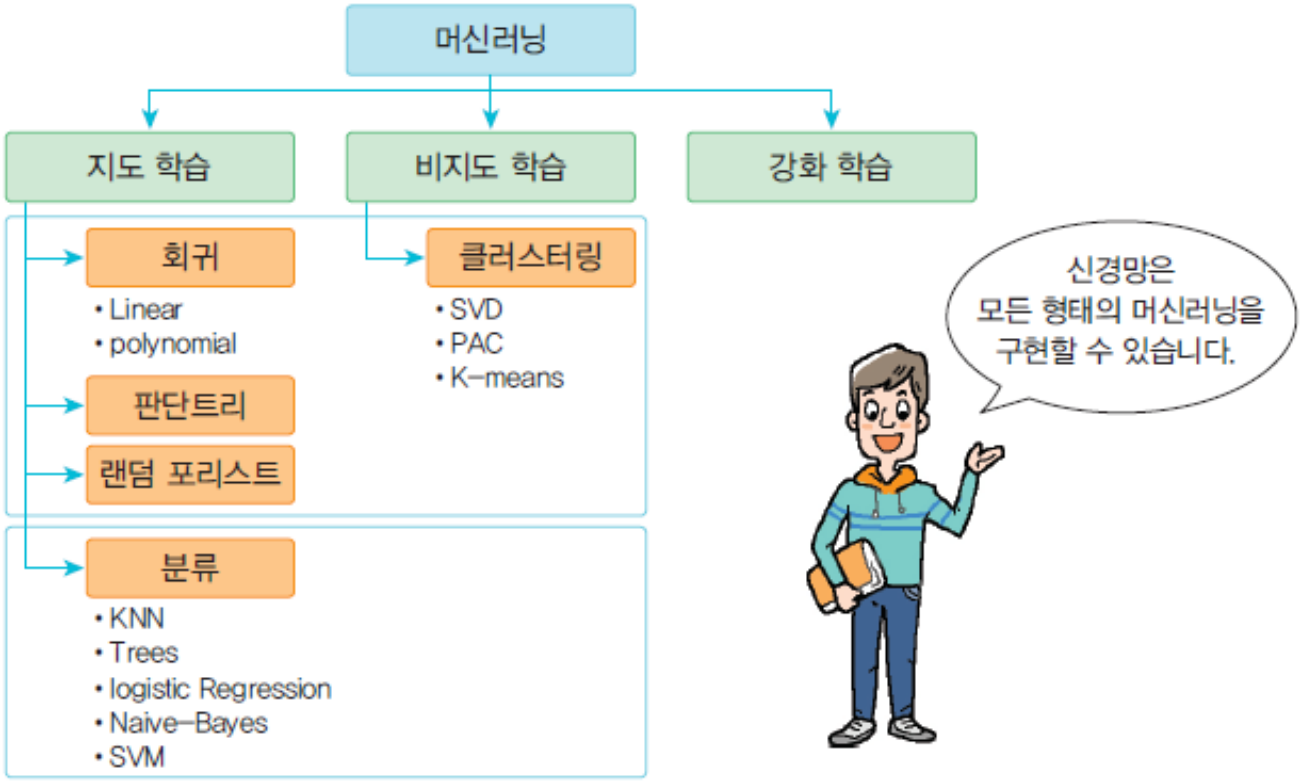
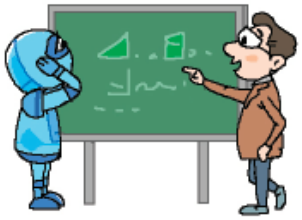


그림 3-6 머신러닝의 종류

머신러닝의 종류

지도 학습



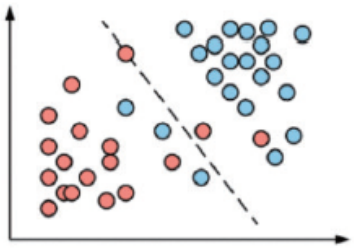
비지도 학습

강화 학습

지도 학습(Supervised Learning)

컴퓨터는 "교사"에 의해 주어진 예제(샘플)와 정답(레이블)을 제공받는다. 지도 학습의 목표는 입력을 출력에 매핑하는 일반적인 규칙(함수, 패턴)을 학습하는 것이다.

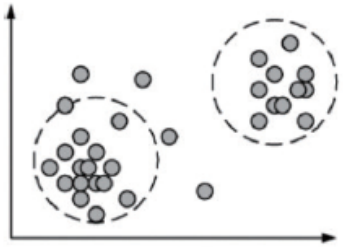
예를 들어서 강아지와 고양이를 구분하는 문제라면 강아지와 고양이에 대한 사진을 제공한 후에, 교사가 어떤 사진이 강아지인지, 어떤 사진이 고양이인지를 알려주는 것이다.



비지도 학습(Unsupervised learning)

외부에서 정답(레이블)이 주어지지 않고 학습 알고리즘이 스스로 입력 데이터에서 어떤 패턴을 발견하는 학습이다.

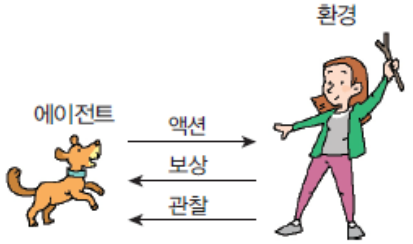
예를 들어 이름(레이블)이 붙어 있지 않은 과일을 분류하는 문제를 생각해 보자. 그래도 우리는 과일의 모양, 색상, 크기 등 다양한 특징을 이용하여 유사한 과일들을 분류할 수 있다.



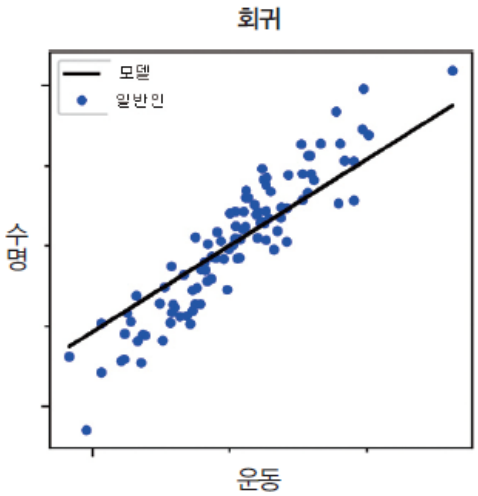
강화 학습(reinforcement Learning)

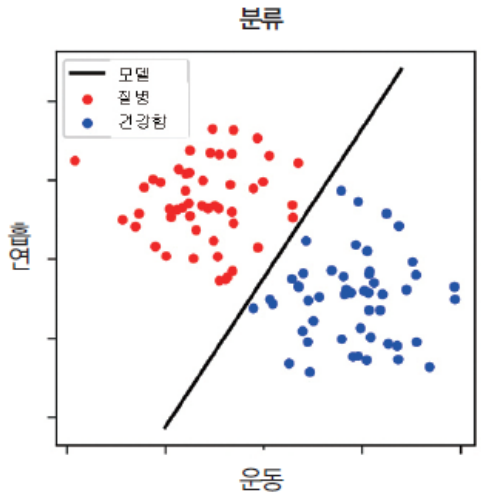
보상 및 처벌의 형태로 학습 데이터가 주어진다. 주로 차량 운전이나 상대방과의 경기 같은 동적인 환경에서 프로그램의 행동에 대한 피드백만 제공되는 경우이다.

예를 들어서 바둑에서 어떤 수를 두어서 승리하였다면 보상이 주어지는 식이다. 강화 학습에서는 보상과 처벌을 통하여 학습이 이루어진다.



지도학습





회귀(regression)

회귀는 주어진 입력–출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것이다. 회귀에서는 학습시키는 데이터가 이산적이 아니고 연속적이다. 즉 입력과 출력이 모두 실수이다. 회귀 모델은 연속적인 값을 예측한다. 예를 들어 운동과 수명에 대하여 학습한 후에, 회귀 모델은 다음과 같은 질문에 대한 답을 예측할 수 있다. "운동을 하루 5시간 한다면 그 사람의 예측 수명은 어떻게 될까?"

분류(classification)

입력을 두 개 이상의 레이블(유형)로 분할하는 것이다. 해당 모델을 학습시킬 때 우리는 레이블을 제공해야 한다. 학습 시에는 교사가 있어서 입력의 올바른 레이블을 알려준다. 학습이 끝나면 학습자가 한 번도 보지 못한 입력을 이들 레이블 중의 하나로 분류하는 시스템이다. 예를 들어서 운동시간을 입력하였을 때 "건강함", "질병상태"로 분류할 수 있다. 이때 입력은 운동시간이고 레이블은 "건강함", "질병상태"이다.

회귀(regression)

- 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것
- 입력(x)과 출력(y)이 주어질 때, 입력에서 출력으로의 매핑 함수를 학습하는 것이라 할 수 있다.

$$y = f(x)$$

- 학습이 끝나면 매핑 함수가 만들어지고 새로운 입력 데이터(x)가 있을 때 이 매핑 함수를 통하여 해당 데이터에 대한 출력 변수값(y)를 예측할 수 있다.

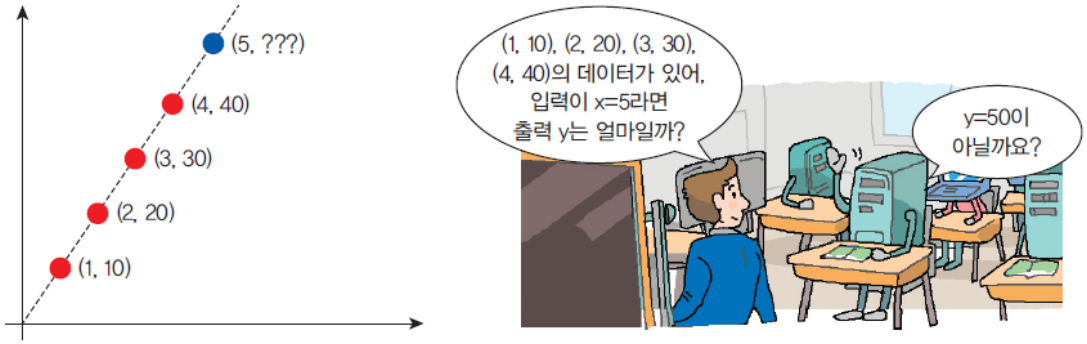
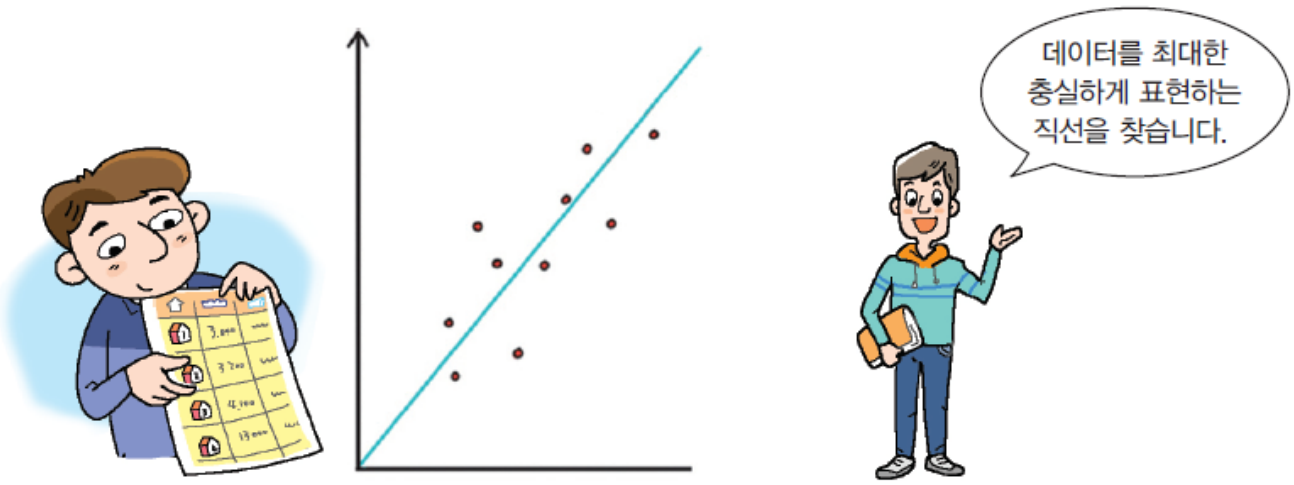


그림 3-7 회귀

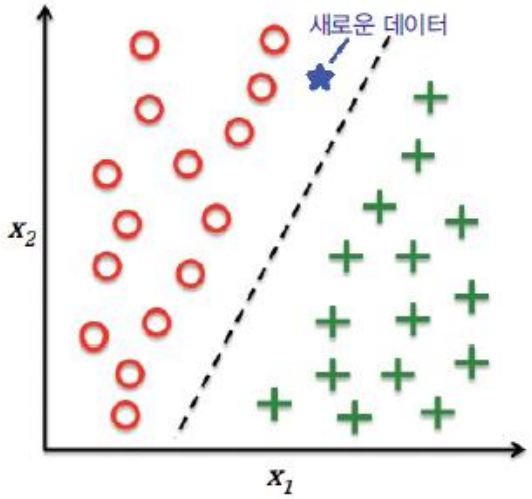
회귀(regression)

- 회귀에서는 입력과 출력이 모두 연속적인 수치값이다.
  - 분류는 비연속적(이산적)
  - 예. 면적을 기반으로 특정 지역에 있는 아파트 가격을 예측하는 알고리즘



분류(classification)

- 앞에 나왔던 식 y = f(x)에서 출력 y가 이산적(discrete)인 경우에 이것을 분류 문제(또는 인식 문제)라고 부른다.
- 입력을 2개 이상의 클래스(부류)로 나누는 것
- 예. 사진을 보고 “강아지” 또는 “고양이”로 분류하는 것. 이메일에서 스팸 메일을 찾아내는 것. 도 분류 문제이다.

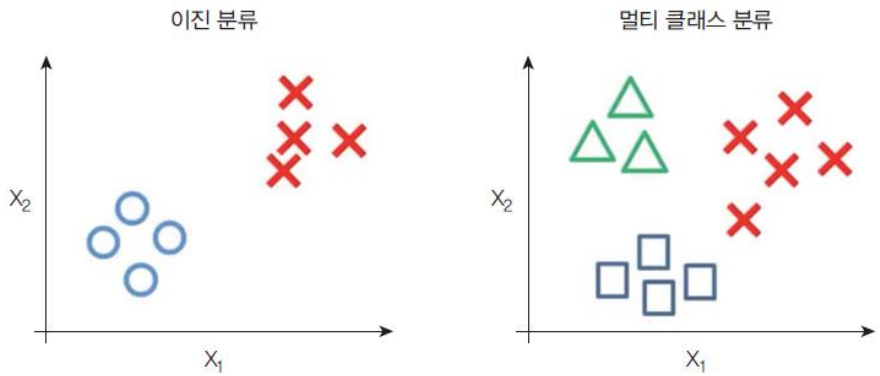


분류(classification)

- 많은 과일로 채워진 과일 바구니를 보고, 프로그램이 바나나, 오렌지와 같은 올바른 레이블을 예측하는 경우. 과일 크기, 색상, 모양 등이 특징이고 과일 이름은 정답(레이블)이다

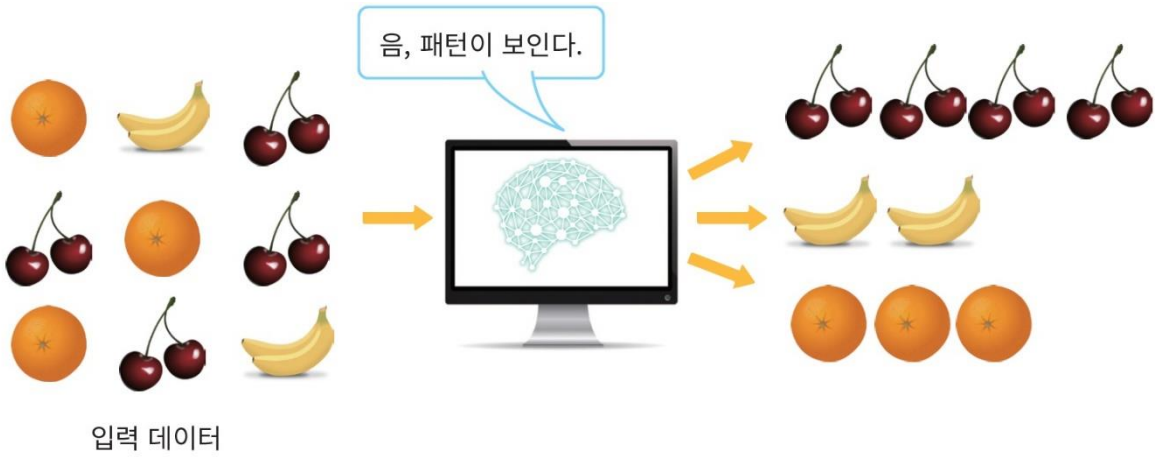
번호	크기	색상	모양	과일 이름
1	크다.	빨강색	동근 모양에 꼭지가 있음	사과
2	작다.	빨강색	심장모양	체리
3	크다.	녹색	길고 곡선 형태의 원통 모양	바나나
4	작다.	녹색	타원형, 다발 형태	포도

- 분류를 수행하기 위한 일반적인 알고리즘에는 신경망, kNN(k-nearest neighbor), SVM(Support Vector Machine), 의사 결정 트리 등이 있다.

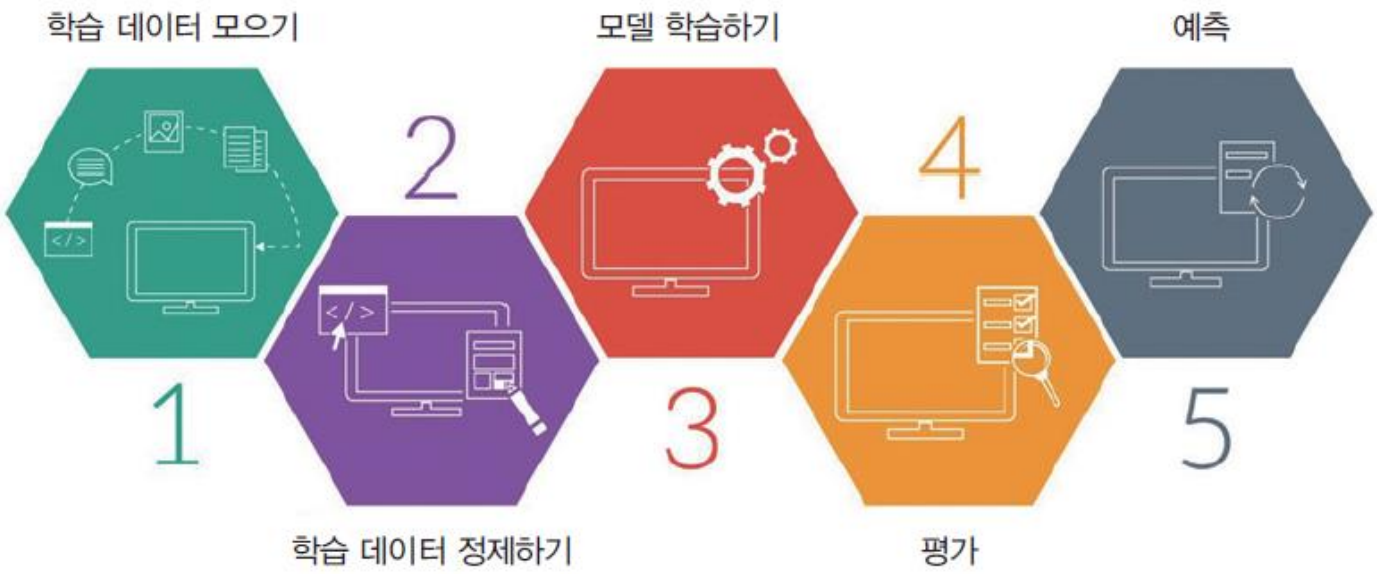


비지도 학습

- 비지도 학습(unsupervised Learning)은 “교사” 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다. 식 y = f(x)에서 레이블 y가 주어지지 않는 것이다.
- 데이터들의 상관도를 분석하여 유사한 데이터들을 모을 수는 있다.

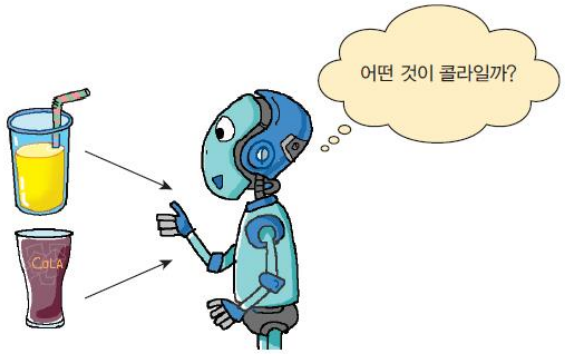


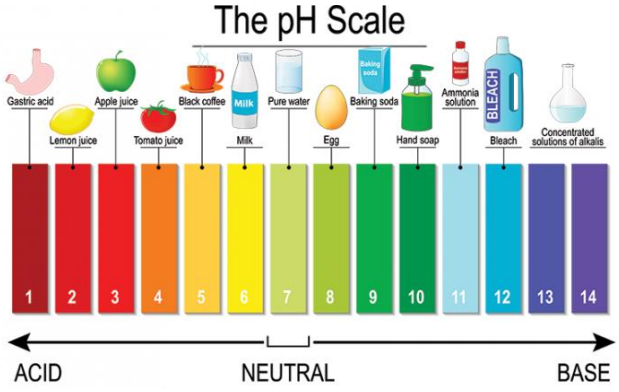
머신 러닝의 과정



머신러닝의 예

- 우리가 음료수가 콜라인지 주스인지를 판별하는 시스템을 만들어 달라는 요청을 받았다고 가정하자. 머신러닝에서는 이러한 시스템을 "모델(model)"이라고 하며, 이 모델은 "학습(train)"이라는 과정을 통해 생성된다.





- 어떤 데이터가 필요할까? 색상(파장)과 산성도(실수)의 두 가지를 선택하도록 하자. 우리는 이 두 가지 요소만으로 음료수를 구별할 수 있기를 희망한다. 이것을 "특징(feature)"라고 부른다.

데이터 수집

- 수집되는 데이터의 품질과 양이 예측 모델이 얼마나 좋은지를 결정하기 때문에, 이 단계는 매우 중요하다.
- 마트에 가서 다양한 콜라와 주스를 사고 산성도 측정을 수행할 장비도 필요.
- 음료의 색상, 산성도, 콜라 또는 주스 여부에 대한 표를 생성한다. 이것이 훈련 데이터가 된다.

색상(nm)	산성도(pH)	라벨
610	3.8	오렌지주스
380	2.5	콜라
390	2.6	콜라
...	...	...

훈련 데이터와 테스트 데이터

- 데이터를 두 부분으로 분할한다.
- 일부는 모델 학습에 사용 -> 훈련(학습) 데이터
- 나머지 부분은 훈련된 모델의 성능 평가에 사용 -> 테스트 데이터

훈련 데이터와 테스트 데이터

- 예. '콜라'와 '주스'레이블이 붙어있는 훈련 데이터로 시스템을 학습시킨다. 학습 알고리즘은 입력 데이터의 특징에 따라 '콜라'와 '주스'로 분류할 수 있는 모델을 내부적으로 생성한다. 특징 공간 위에 경계선이 만들어 진다.

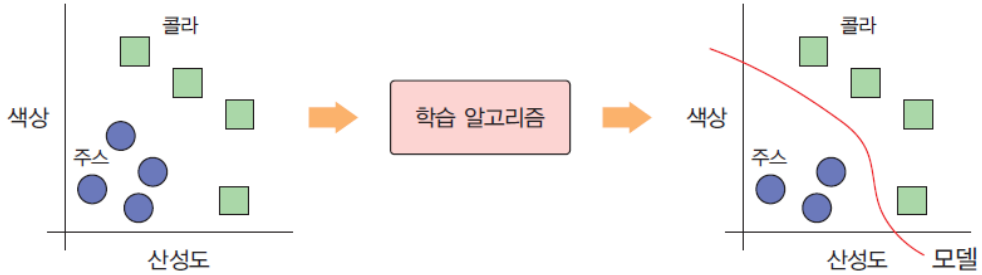


그림 3-10 훈련 단계

- 학습이 끝나면 한 번도 본 적이 없는 새로운 데이터(테스트 데이터)로 시스템을 테스트한다.

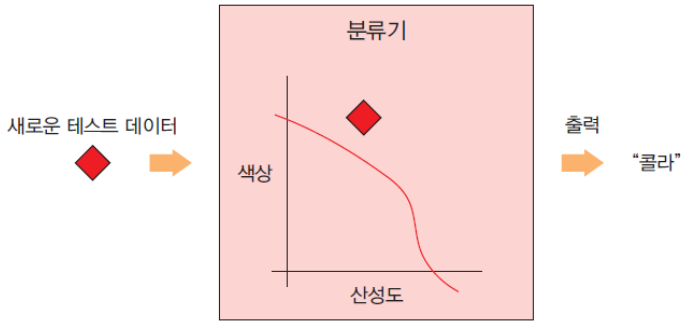


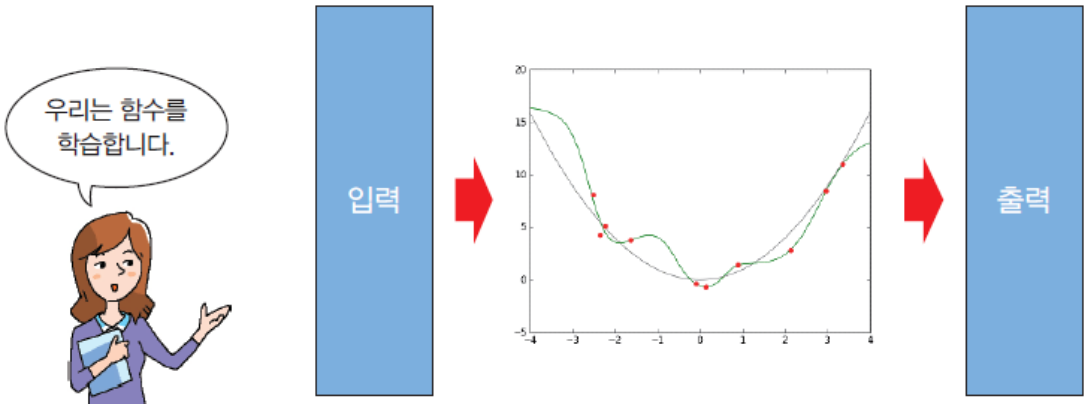
그림 3-11 테스트 단계

데이터 정제

- 때대로 우리가 수집하는 데이터는 다른 형태의 조정 및 조작이 필요하다.
- 중복 제거, 정규화, 오류 수정 등이 필요한 경우가 많다.
- 이는 모두 데이터 준비 단계에서 발생한다.

모델 선택

- 다음 단계는 머신러닝 모델을 선택하는 것이다.
- 많은 연구자들이 수년에 걸쳐 만든 많은 모델이 있다. 일부는 이및 데이터에 매우 적합하고 일부는 소리 데이터, 숫자 데이터, 텍스트 기반 데이터에 적합하다.
- 우리의 경우에는 색상과 산성도라는 두 가지 특성만 있으므로 매우 간단한 선형 모델을 사용하자.



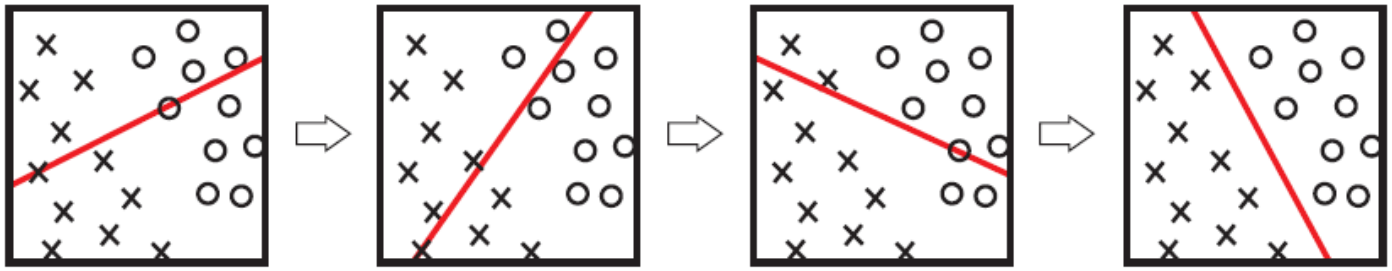
학습

- 이 단계에서는 데이터를 사용하여 주어진 음료가 어떤 것인지를 예측하는 모델의 능력을 점진적으로 개선할 것이다.

그림 3-12 학습 과정

- 우리가 선택한 머신러닝 모델은 선행모델이다. 선형모델은 직선을 이용하여 특징 공간에 입력을 분리한다.
- 직선의 식 y=m*x+b 를 잘 만드는 것이 학습(훈련)의 과정
- 훈련에 변경할 수 있는 값은 m과 b이다. m을 가중치(weight), b를 바이어스(bias)라고 한다.

- Y = m * x + b
- 맨 처음에는 m와 b를 임의의 값을 초기화하고 입력값으로 출력을 예측해본다. 이 출력값을 정확한 결과값과 비교하여 더 정확한 예측을 갖도록 m 및 b의 값을 조정할 수 있다. 이 과정이 반복된다.
- 훈련을 시작할 때 임의의 직선을 그린 것과 같다. 그런 다음 학습의 각 단계가 진행됨에 따라 직선이 단계적으로 이동하여 콜라와 주스의 이상적인 분리에 가까워진다.

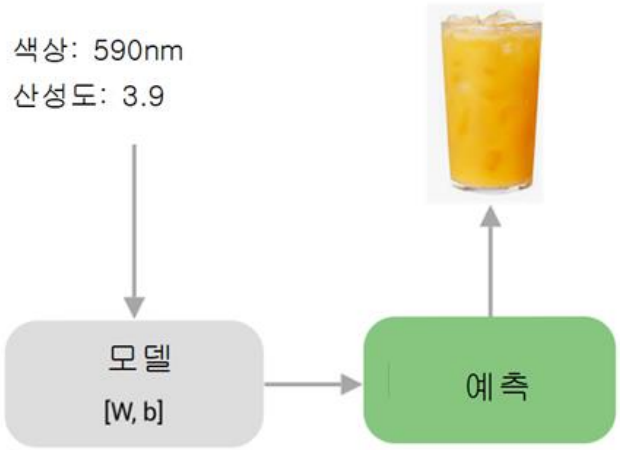


평가

- 학습이 완료되면 모델을 평가하여 모델이 좋은지 나쁜지를 확인해야 한다.
- 테스트 데이터 사용한다. 학습에 사용된 적 없는 데이터에 대해 모델의 성능을 평가할 수 있다.
- 일반적인 훈련 데이터와 테스트 데이터의 비율은 80:20 또는 70:30 이다.

예측

- 이 단계에서 머신러닝 시스템은 우리의 질문에 답한다.
- 예. 색상이 600nm이고 산성도가 1.5인 음료가 무엇인지를 머신러닝 시스템에 물어볼 수 있고, 머신러닝 시스템은 훈련된 대로 색상과 산성도를 고려하여 주어진 음료가 콜라인지 주스인지 예측할 수 있다.



붓꽃을 머신러닝으로 분류해보자

- 데이터 세트 읽어들이기

데이터 세트 이름	함수
보스톤 지역의 집값	load_boston(*[, return_X_y])
붓꽃 데이터	load_iris(*[, return_X_y, as_frame])
당뇨병 데이터	load_diabetes(*[, return_X_y, as_frame])
숫자 이미지 데이터	load_digits(*[, n_class, return_X_y, as_frame])
운동 데이터	load_linnerud(*[, return_X_y, as_frame])
와인 데이터	load_wine(*[, return_X_y, as_frame])
유방암 데이터	load_breast_cancer(*[, return_X_y, as_frame])

붓꽃 데이터 세트

- 1936년에 생물학자인 로날드 피셔의 논문에 등장한 데이터 세트
- 세가지 붓꽃 종류(iris setona, iris virginica, iris versicolor)의 150개 샘플로 구성
- 각 샘플에 4가지 특징의 조합으로 종을 구별하는 선형 판별 모델을 개발
- 꽃받침 길이와 너비, 꽃잎의 길이와 너비

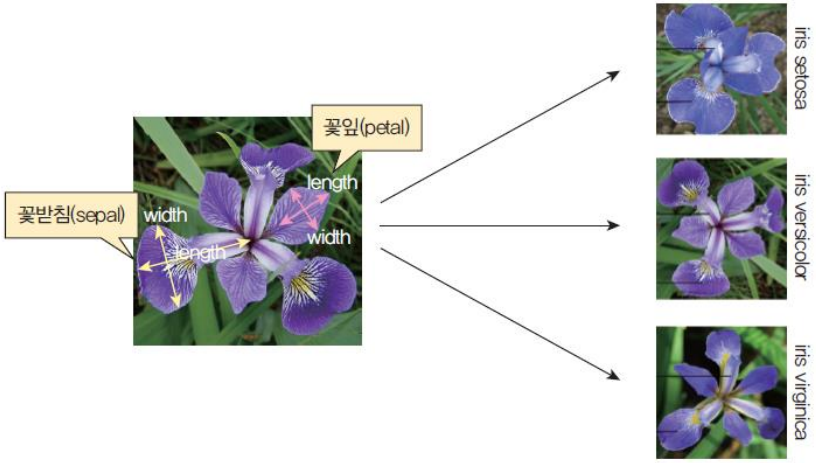
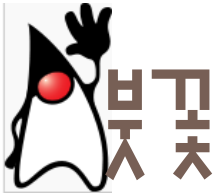


그림 3-13 붓꽃 데이터 세트

붓꽃 데이터 세트

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris)
```

```
{'data': array([[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],
...
[5.9, 3. , 5.1, 1.8]]),
'target': array([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,
...
```



붓꽃 데이터 세트

순번	sepal length (꽃받침 길이)	sepal width (꽃받침 너비)	petal length (꽃잎 길이)	petal width (꽃잎 너비)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.3	0.2	0
2	4.7	3.2	1.3	0.2	0
...					
					1
					2
149					

data target

그림 3-14 특징과 레이블의 구조

훈련 데이터와 테스트 데이터 분리

```
from sklearn.model_selection import train_test_split

X = iris.data
y = iris.target

# (80:20)으로 분할한다.
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=4)

print(X_train.shape)
print(X_test.shape)
```

```
(120, 4)
(30, 4)
```

모델 선택

- k-Nearest Neighbor(kNN) 알고리즘은 모든 머신러닝 알고리즘 중에서도 가장 간단하고 이해하기 쉬운 분류 알고리즘이다.
- kNN은 학습 시에 교사가 존재하는 “지도 학습”이다.

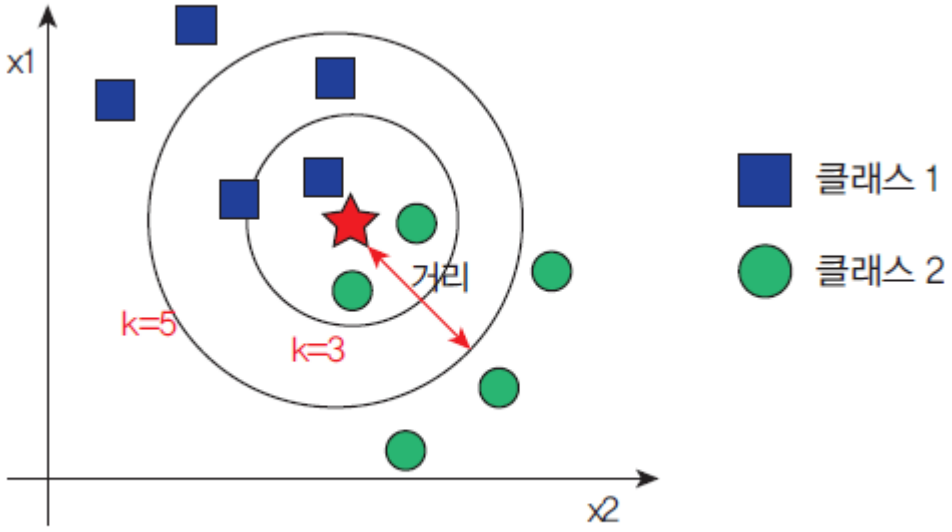


그림 3-15 kNN 알고리즘에서의 분류

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
```

평가

```
y_pred = knn.predict(X_test)
from sklearn import metrics

scores = metrics.accuracy_score(y_test, y_pred)
```

```
0.9666666666666667
```


```
classes = {0:'setosa',1:'versicolor',2:'virginica’}

# 전혀 보지 못한 새로운 데이터를 제시해보자.
x_new = [[3,4,5,2],  [5,4,2,2]]

y_predict = knn.predict(x_new)

print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

```
versicolor
setosa
```

필기체 숫자를 분류해보자.

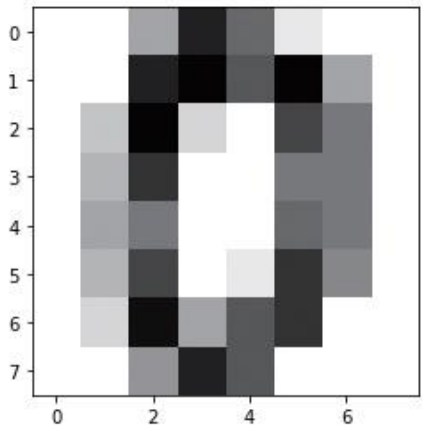
- MNIST가 배포하는 필기체 숫자 이미지
- 각 숫자들은 28x28의 2차원 이미지로 표현. 784픽셀

그림 3-16 MNIST 데이터

- sklearn 라이브러리 안에 있는 간략한 버전. 8x8 이미지(64픽셀)로 실습해보자

데이터 세트 읽기

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split


digits = datasets.load_digits()
plt.imshow(digits.images[0], cmap=plt.cm.gray_r, interpolation='nearest')
```



이미지 평탄화

```
n_samples = len(digits.images)

data = digits.images.reshape((n_samples, -1))
```

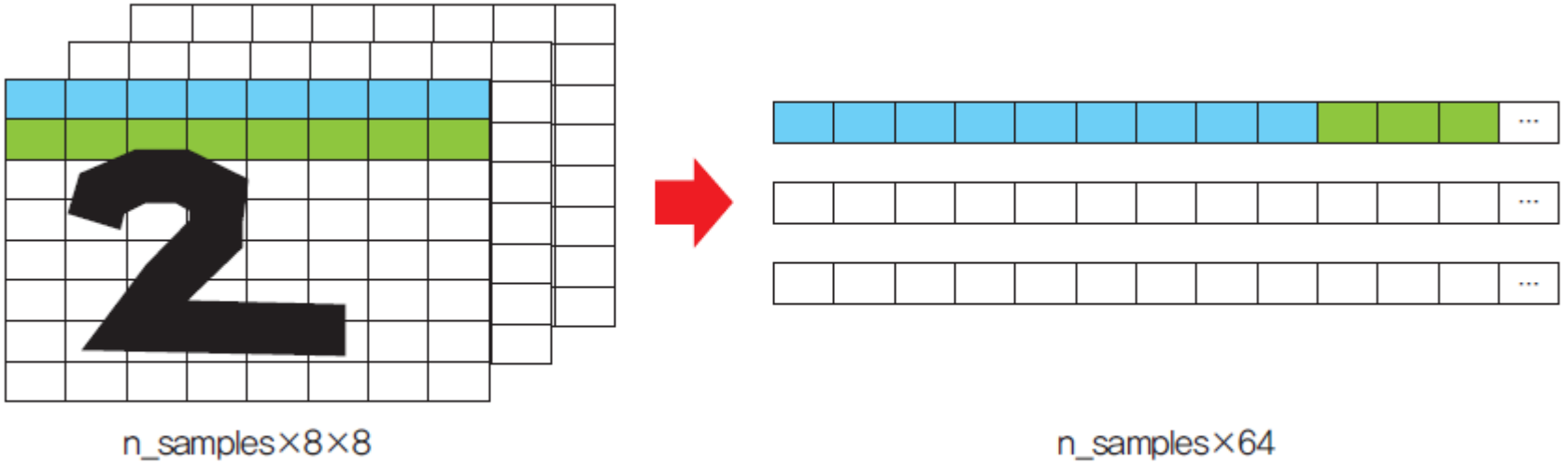


그림 3-17 평탄화

훈련 데이터와 테스트 데이터

```
X_train, X_test, y_train, y_test = train_test_split(
                        data, digits.target, test_size=0.2)
```

모델과 학습

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
```

```
knn.fit(X_train, y_train)
```

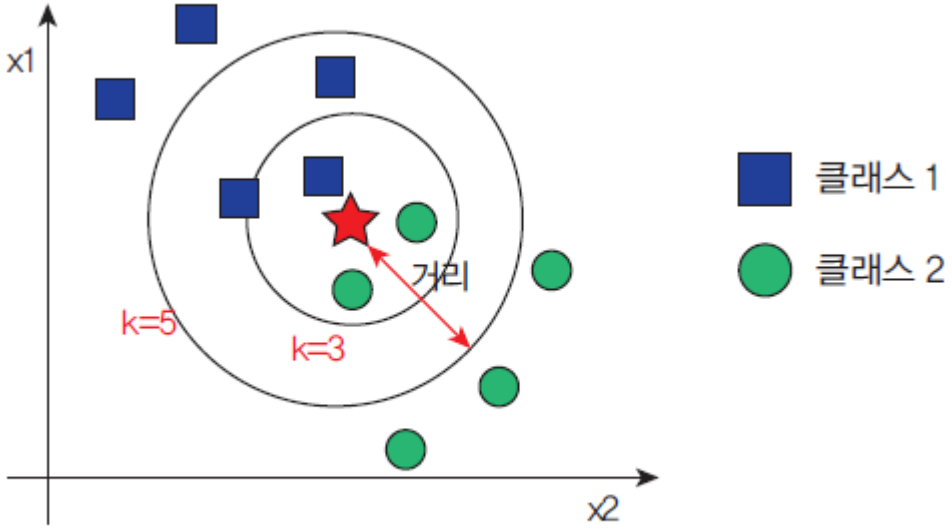


그림 3-15 kNN 알고리즘에서의 분류

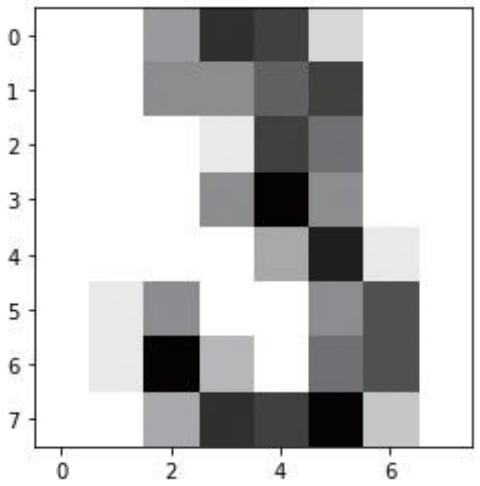

```
# 테스트 데이터로 예측해본다.
y_pred = knn.predict(X_test)
# 정확도를 계산한다.

scores = metrics.accuracy_score(y_test, y_pred)
print(scores)
```

```
0.9532814238042269
```

예측

```
# 이미지를 출력하기 위하여 평탄화된 이미지를 다시 8×8 형상으로 만든다.
plt.imshow(X_test[10].reshape(8,8), cmap=plt.cm.gray_r, interpolation='nearest')

y_pred = knn.predict([X_test[10]]) # 입력은 항상 2차원 행렬이어야 한다.
print(y_pred)
```

```
[3]
```



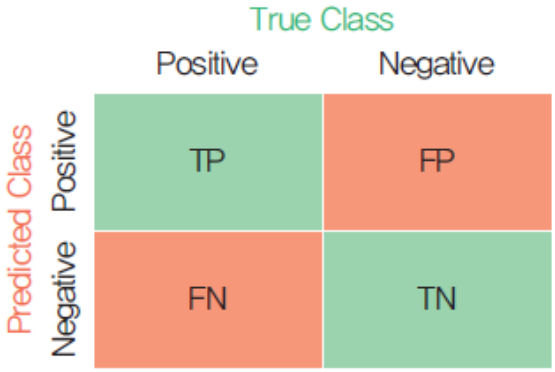
머신러닝 알고리즘의 성능평가

- 정확도

$$\text{정확도}(accuarcy) = \frac{\text{올바르게 분류한 샘플 수}}{\text{전체 샘플 수}}$$

머신러닝 알고리즘의 성능평가

- 혼동행렬



- 긍정을 긍정으로 올바르게 예측하면 TP(True Positive)라고 한다. 앞에 True가 붙으면 예측과 실제가 같다는 의미이다.
- 긍정을 부정으로 잘못 예측하면 FN(False Nagative)라고 한다. 앞에 False가 붙으면 예측과 실제가 틀리다는 의미이다.
- 부정을 긍정으로 잘못 예측하면 FP(False Positive)라고 한다.
- 부정을 부정으로 올바르게 예측하면 TN(True Nagative)라고 한다.

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

머신러닝 알고리즘의 성능평가

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

- 민감도 : 맞는 것을 맞다고 판정하는 비율
- 특이도 : 틀린 것을 틀리다고 판정하는 비율

- 예.

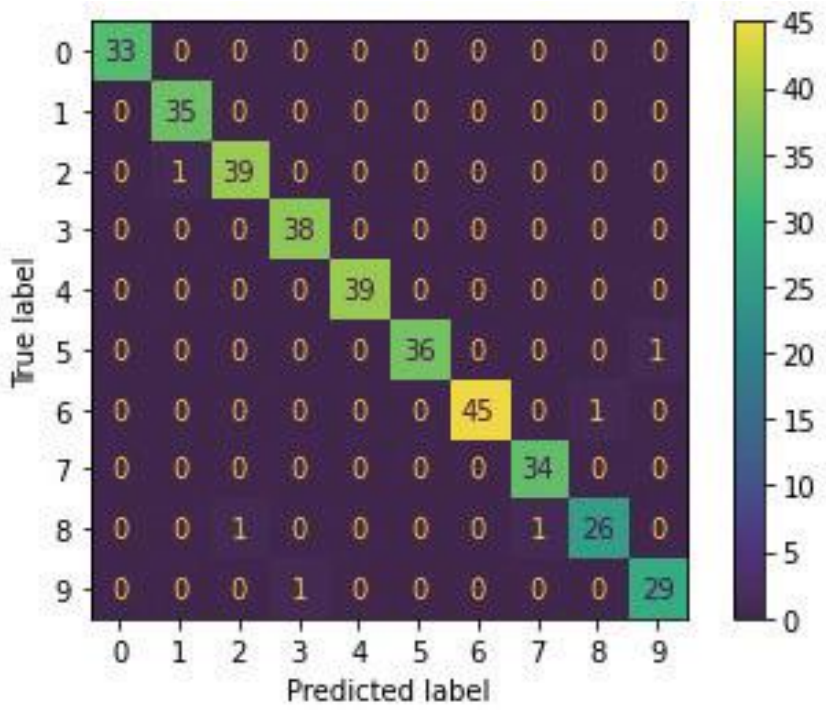
	폐암 O	폐암 X
X-ray 양성	90	100
X-ray 음성	10	800

- 민감도 = 90 / 100 = 0.9
- 특이도 = 800 / 900 = 0.89

혼동 행렬 출력

```
import matplotlib.pyplot as plt

from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split

digits = datasets.load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)

X_train, X_test, y_train, y_test = train_test_split(
            data, digits.target, test_size=0.2)

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

disp = metrics.plot_confusion_matrix(knn, X_test, y_test)
plt.show()
```

혼동 행렬 출력


True label
Predicted label
0 1 2 3 4 5 6 7 8 9
33 0 0 0 0 0 0 0 0 0
0 35 0 0 0 0 0 0 0 0
0 1 39 0 0 0 0 0 0 0
0 0 0 38 0 0 0 0 0 0
0 0 0 0 39 0 0 0 0 0
0 0 0 0 0 36 0 0 0 1
0 0 0 0 0 0 45 0 1 0
0 0 0 0 0 0 0 34 0 0
0 0 1 0 0 0 0 1 26 0
0 0 0 1 0 0 0 0 0 29
45 40 35 30 25 20 15 10 5 0

분류 리포트

- 사이킷런에서는 분류 리포트를 생성하는 기능이 있다. 대표적인 성능 척도들을 계산해준다

```
print(f"{metrics.classification_report(y_test, y_pred)}\n")
```

변수나 수식의 값을 문자열로 통합

	precision	recall	f1-score	support
0	1.00	1.00	1.00	28
1	0.95	1.00	0.97	39
2	1.00	1.00	1.00	33
3	0.95	1.00	0.97	36
4	1.00	1.00	1.00	35
5	1.00	1.00	1.00	35
6	1.00	1.00	1.00	45
7	1.00	1.00	1.00	40
8	0.97	0.93	0.95	40
9	1.00	0.93	0.96	29
accuracy			0.99	360
macro avg	0.99	0.99	0.99	360
weighted avg	0.99	0.99	0.99	360

머신러닝은 어디에 이용되는가?

- 경우의 수가 많아 명시적 알고리즘을 설계하고 프로그래밍하는 것이 어렵거나 불가능한 경우 -> if.. else 문으로 만드는 것이 불가능

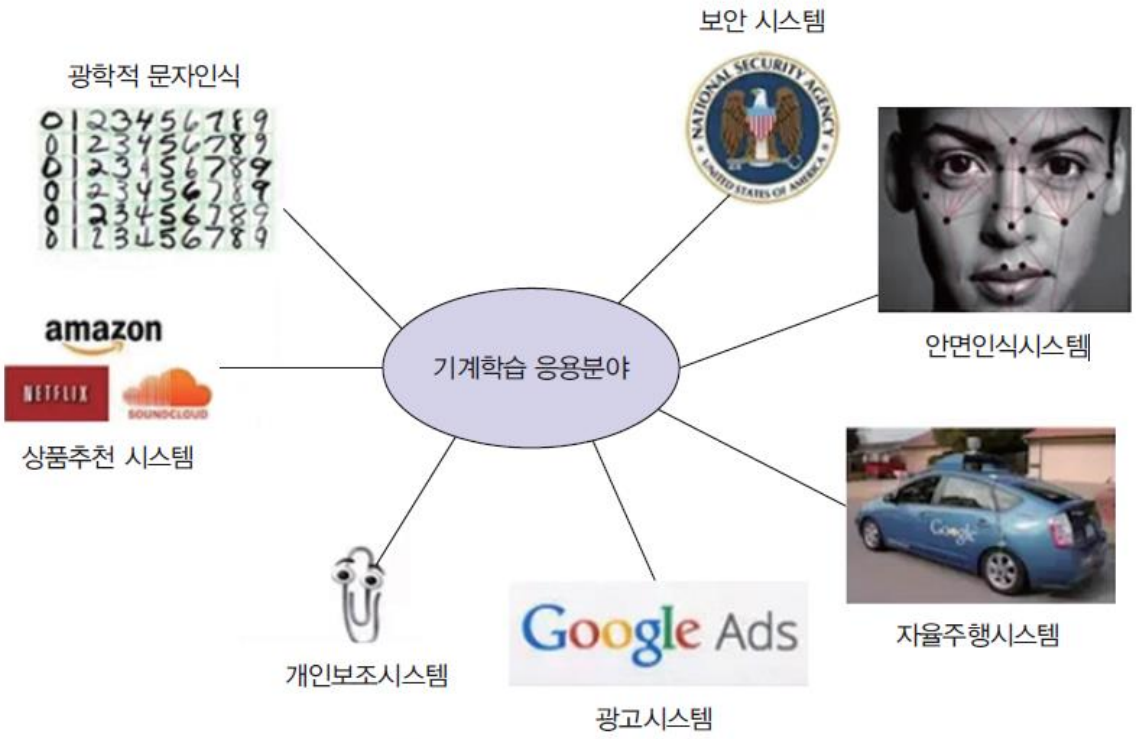


그림 3-18 머신러닝의 응용 분야

머신러닝은 어디에 이용되는가?

- 복잡한 데이터들이 있고, 이들 데이터에 기반하여 결정을 내려야 하는 분야 – 빅데이터(big data)
  - 영상 인식, 음성 인식처럼 프로그램으로 작성하기에는 규칙과 공식이 너무 복잡할 때
  - 보안 시스템에서 침입을 탐지하거나 신용 카드 거래 기록에서 사기를 감지하는 경우처럼 작업 규칙이 지속적으로 바뀌는 상황일 때
  - 주식 거래나 에너지 수요 예측, 쇼핑 추세 예측의 경우처럼 데이터 특징이 계속 바뀌고 프로그램을 계속해서 변경해야 하는 상황일 때
  - 전자 메일 메시지가 스팸인지 아닌지 여부
  - 신용 카드 거래가 허위인지 여부를 판별하는 시스템
  - 구매자가 클릭할 확률이 가장 높은 광고가 무엇인지를 알아내는 시스템

프로그래머로서 머신러닝의 실용적인 가치

- 첫 번째는 프로그래밍 시간을 줄일 수 있다는 점이다.
  - 맞춤법 오류를 수정하는 프로그램. 전통적인 프로그래밍 방법-규칙을 만들어 - 상당한 시간 필요. 많은 훈련 샘플만 있다면 머신러닝으로 학습시켜 빠른 시간 안에 신뢰성있는 프로그램을 완성할 수 있다.
- 두 번째는 맞춤형 제품을 쉽게 개발할 수 있다.
  - 완성된 한국어 맞춤법 수정 프로그램을 이용해서 영어 버전으로 변경하려면. 각 언어마다 새로 작성하려면 엄청난 시간이 필요. 머신 러닝을 사용한다면 훈련 샘플만 있으면 된다.
- 세 번째는 프로그래머로 시도한 알고리즘이 떠오르지 않는 문제들을 해결할 수 있다.
  - 사람의 얼굴을 인식하는 프로그램. 머신러닝 알고리즘에게 수많은 샘플 예제만 보여주기만 하면 문제가 해결된다.

데이터 라벨링

- 머신러닝에서는 데이터를 이용하여 시스템을 학습시킨다. 따라서 많은 데이터가 필요하고 또 데이터에 레이블(label)이 붙어 있어야 한다.
- 시스템이 강아지와 고양이를 구별하게 학습시키려면 수천 장의 사진들이 필요하고 이들은 모두 강아지와 고양이로 구분되어 있어야 한다. 즉 사진에 라벨이 붙어 있어야 한다.
- 이들 구분 작업은 사람이 해줘야 한다. 이렇게 데이터에 라벨을 붙이는 작업을 데이터 라벨링(data labeling, data annotation)이라고 한다.
- ”데이터 라벨러”라고 하는 새로운 직업이 생겨나고 있다고 한다. 21세기의 ‘인형 눈알 붙이기’ 부업에 비유되기도 한다.