머신러닝의 기초

학습 목표

- 머신러닝과 전통적인 프로그래밍의 차이를 이해한다.
- 머신러닝의 과정을 이해한다.
- 붓꽃 데이터를 분류해본다.
- 숫자 이미지를 분류해본다.
- 머신러닝의 성능 측정 척도들을 살펴본다.

머신러닝이란?

- 현재의 컴퓨터는 스스로 학습할 수 없기 때문에 우리가 컴퓨터에게 어떤 작업을 시키려면 반드시 프로그램을 작성하여 작업을 지시하여야 한다.
- 컴퓨터가 스스로 학습할 수 있다면 컴퓨터는 프로그램 없이도 여러 가지 일을 할 수 있을 것이다.

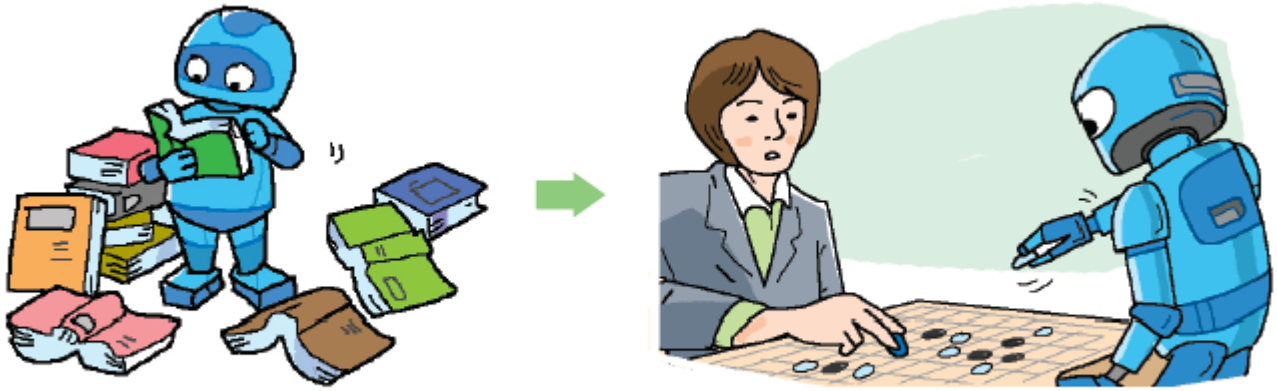


그림 3-1 머신러닝

머신러닝과 전통적인 프로그래밍과의 차이점

- 전통적인 접근 방식은 원하는 절차를 "프로그래밍"하는 것이다. 즉 인간이 컴퓨터에게 적절한 문제 해결 알고리즘을 만들어서 건네주어야 한다.
- 우리는 강아지를 인식하는 프로그램을 작성하지 않는다. 단순히 많은 수의 동물 사진을 머신러닝 시스템에 제공하고 어떤 사진이 강아지인지만 알려주면 된다. 이런 식으로 훈련이 진행되면 머신러닝 시스템이 스스로 사진에서 강아지를 인식할 수 있다

머신러닝은 규칙을 사전에 제공해주지 않는다.

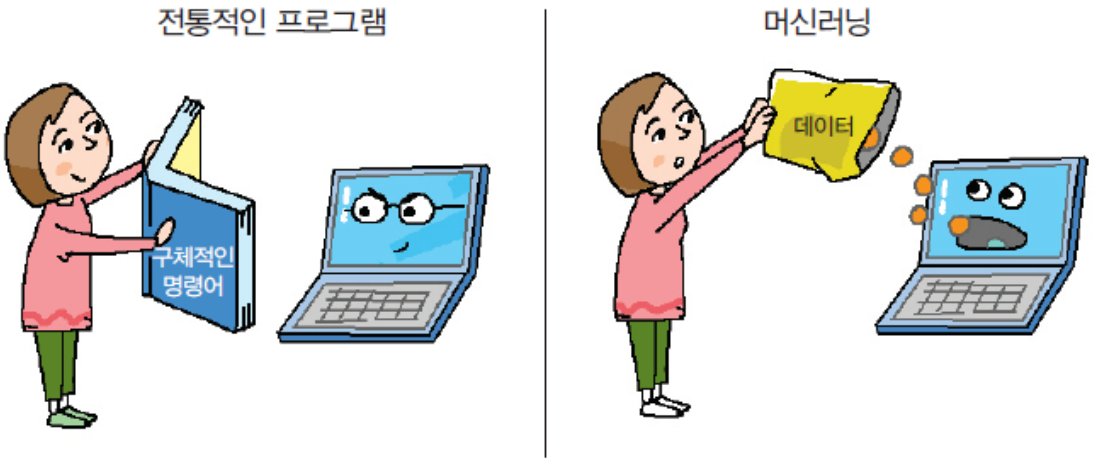
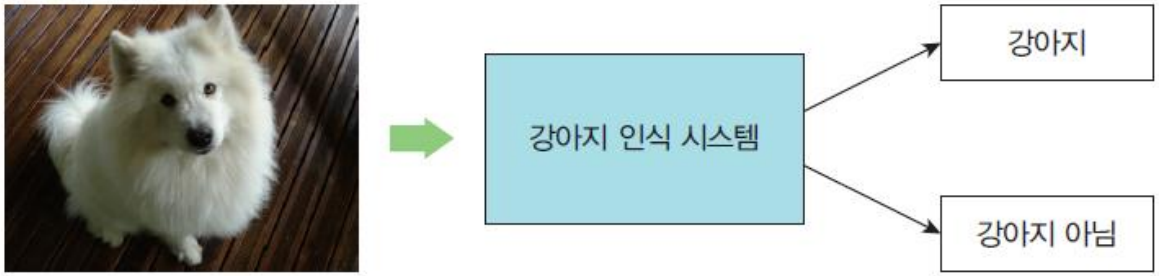


그림 3-2 전통적인 프로그래밍과 머신러닝의 차이점

예를 들어서 강아지와 고양이를 구별하는 문제를 생각해보자.

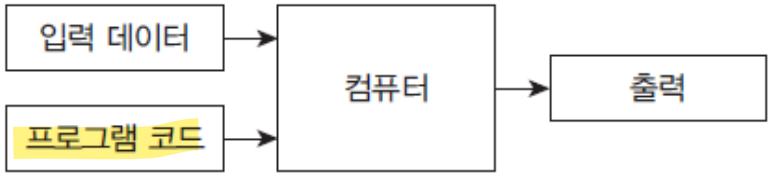
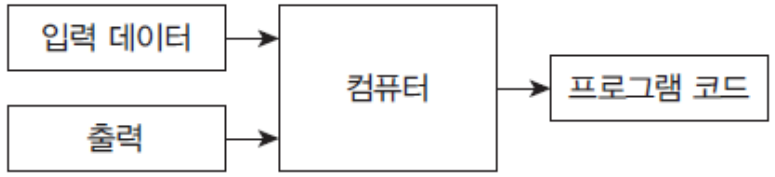
- 인간에게는 너무 쉬운 작업이지만 알고리즘으로 구성하기가 매우 어렵다.



머신러닝 방법

다량의 데이터를 밀어넣고 , 답을 준다. 데이터의 특성을 파악하고 정답을 스스로 찾는다..

전통적인 프로그래밍과 머신러닝을 비교

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

새로운 문제를 푸려면 새로운 규칙을 업데이트 해야하는 단점이 있다.

사전에 규칙을 지정하지 않는다

인공지능, 머신러닝, 딥러닝

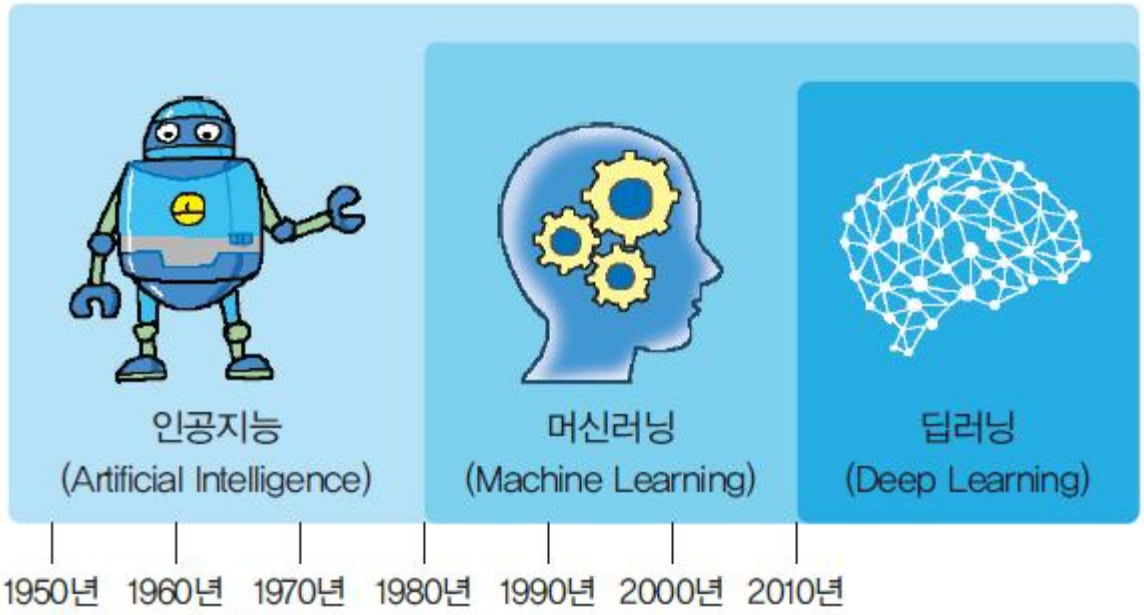


그림 3-4 인공지능, 머신러닝, 딥러닝 간의 관계

머신러닝의 역사

- IBM에서 근무하던 아서 사무엘(Arthur Samuel)이 컴퓨터 게임과 인공지능을 연구하면서 1959년에 "머신러닝"이란 용어를 만들었다.
- 그는 IBM 최초의 상용 컴퓨터인 IBM 701에서 최초의 체커 프로그램을 만들었다.

머신러닝의 역사

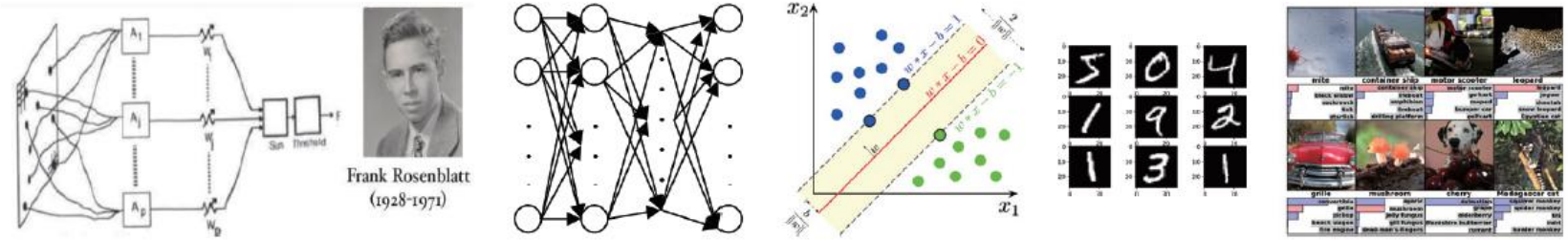
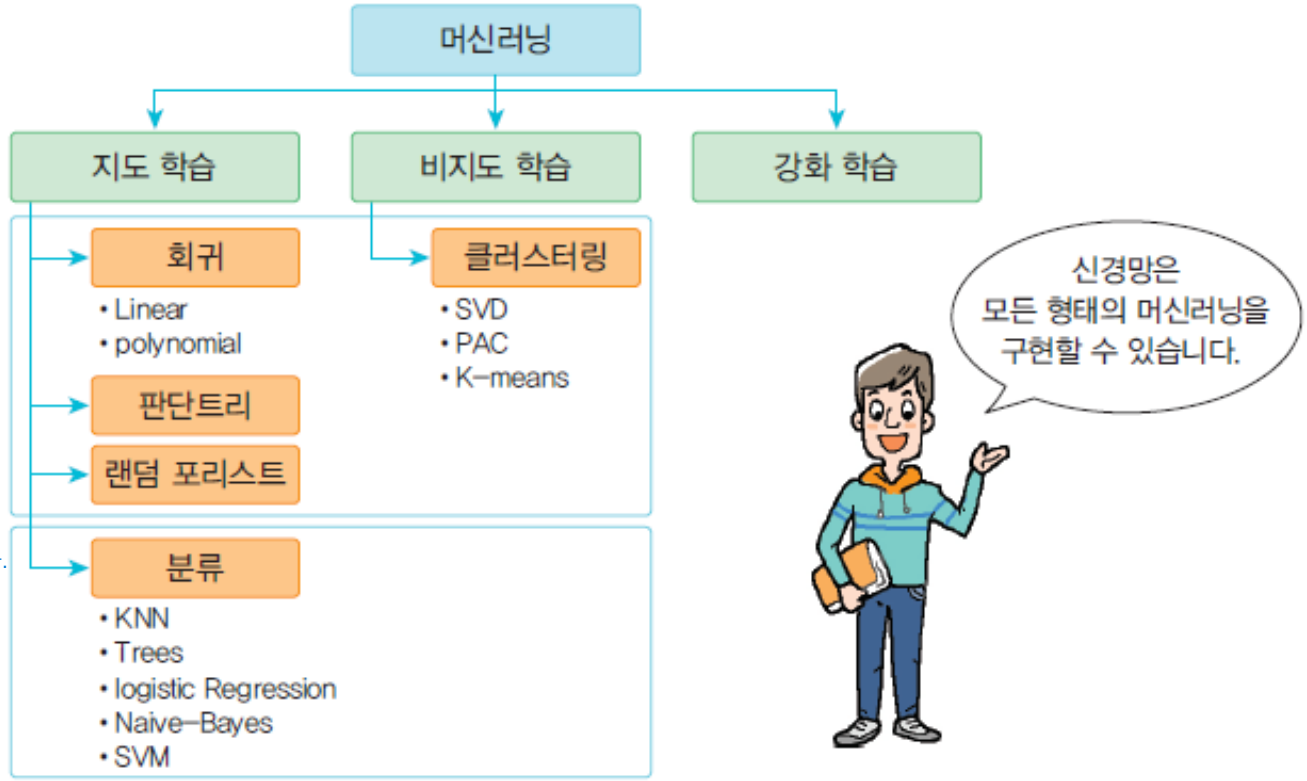


그림 3-5 머신러닝의 역사

머신러닝의 종류



회귀는 예측 할 때 사용

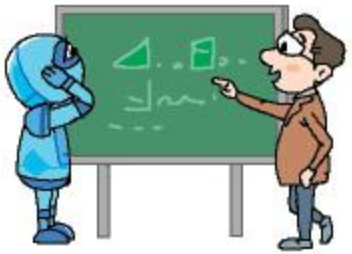
분류는 클래스, 그룹이다.

그림 3-6 머신러닝의 종류

머신러닝의 종류

데이터를 모아줄 때 비지도 학습을 사용

지도 학습



비지도 학습

강화 학습

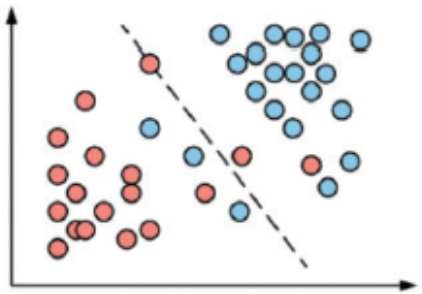
머신러닝의 종류

데이터와 , 정답을 넣을 때

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.

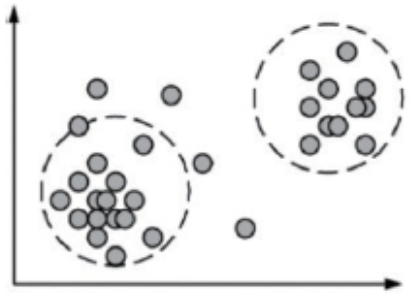
예를 들어서 강아지와 고양이를 구분하는 문제라면 강아지와 고양이에 대한 사진을 제공한 후에, 교사가 어떤 사진이 강아지인지, 어떤 사진이 고양이인지를 알려주는 것이다.



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.

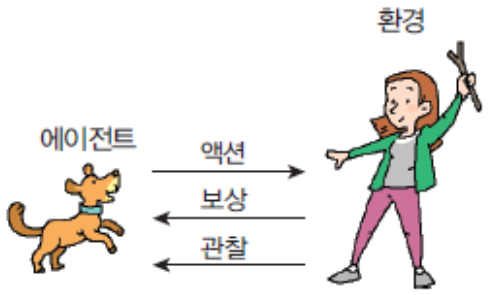


게임에 많이 사용이 된다.

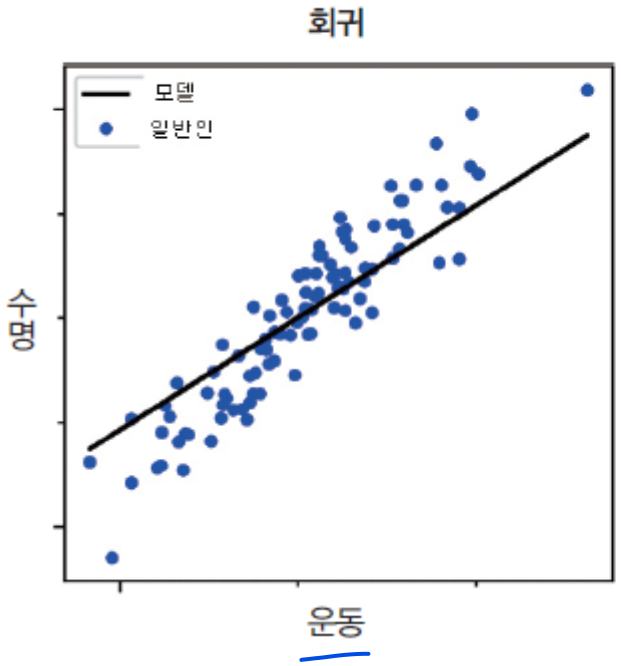
강화 학습(reinforcement Learning)

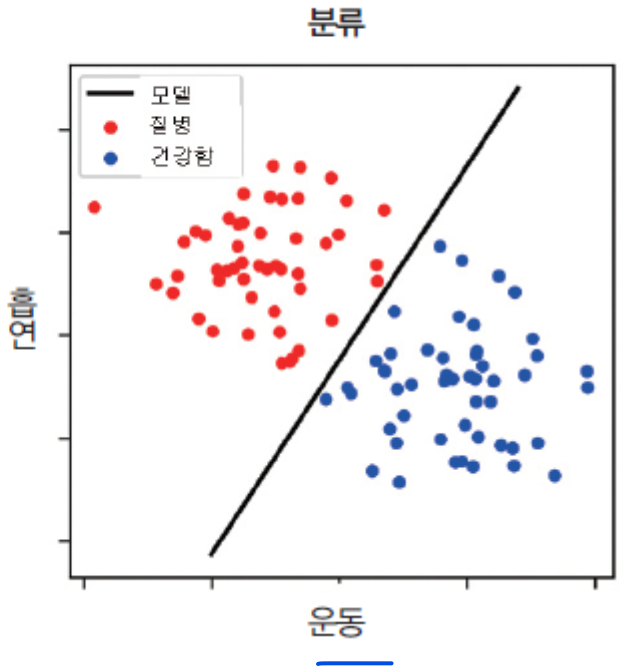
보상 및 처벌의 형태로 학습 데이터가 주어진다. 주로 차량 운전이나 상대방과의 경기 같은 동적인 환경에서 프로그램의 행동에 대한 피드백만 제공되는 경우이다.

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



지도학습





회귀

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

주식 , 일기예보에 많이 사용

$$y = f(x)$$

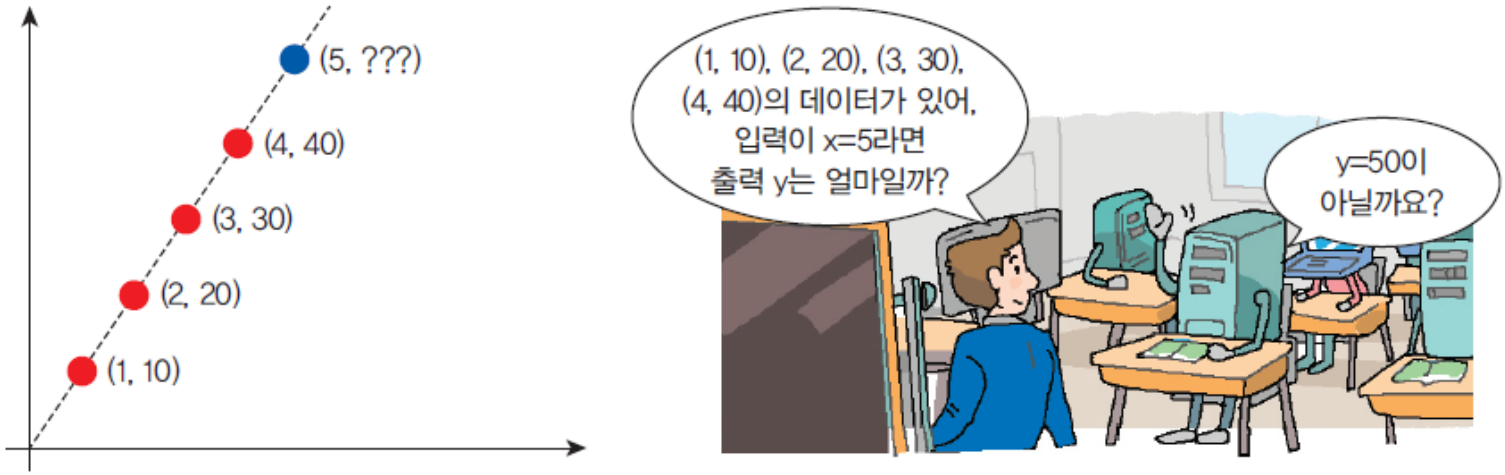
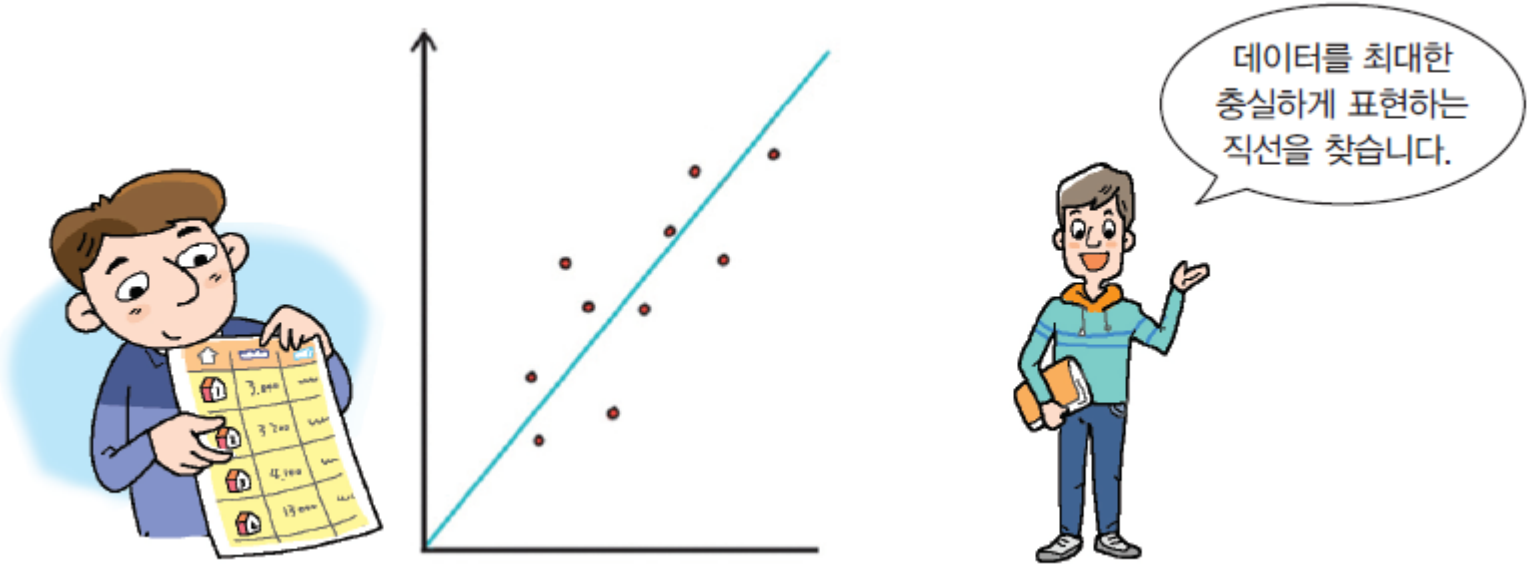


그림 3-7 회귀

회귀

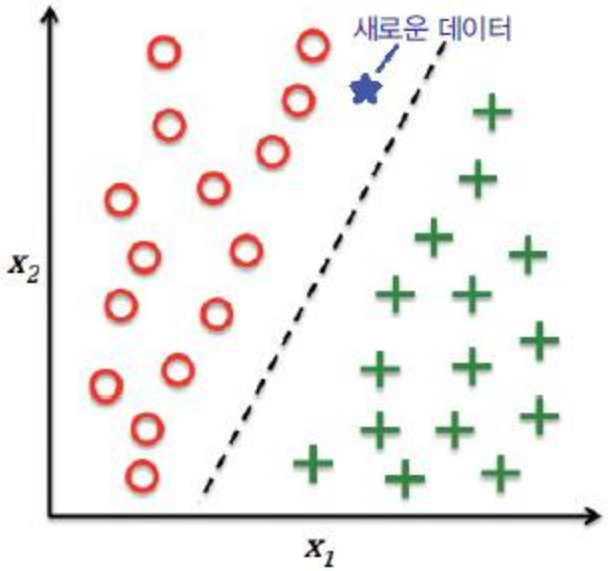
- 회귀(regression) : 회귀에서는 입력과 출력이 모두 실수이다.
  - “사용자가 이 광고를 클릭할 확률이 얼마인가요?”
  - “면적에 따른 각 아파트의 가격은 어떻게 되나요?”



분류

- 식 y = f(x)에서 출력 y가 이산적(discrete)인 경우에 이것을 분류 문제(또는 인식 문제)라고 부른다.
- 분류에서는 입력을 2개 이상의 클래스(부류)로 나누는 것이다.
- 예를 들어서 사진을 보고 “강아지”, 또는 “고양이”로 분류하는 것도 분류 문제이다.

지도는 supervised



분류의 예

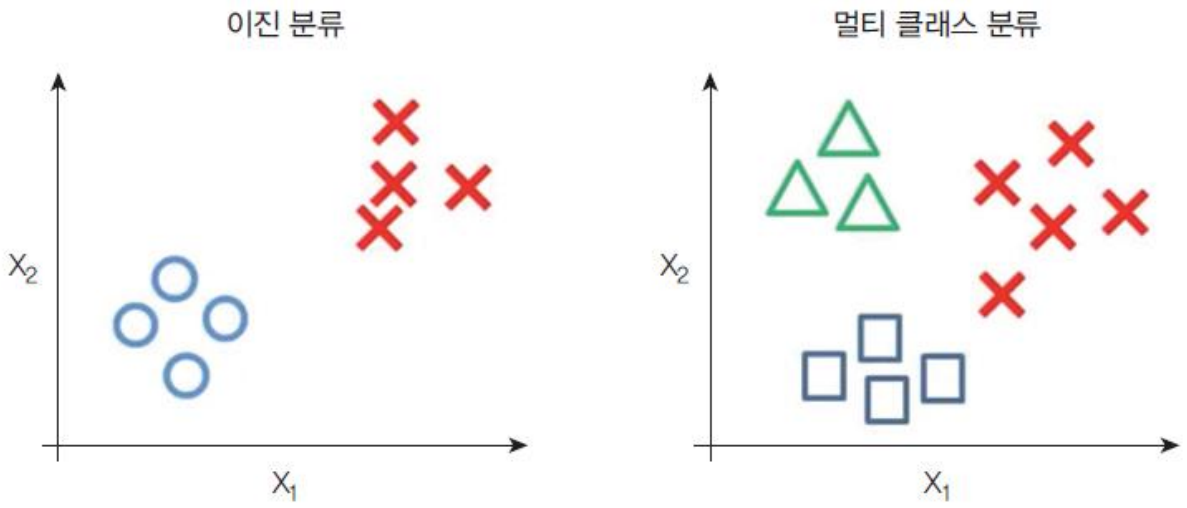
- 많은 과일로 채워진 과일 바구니를 보고, 프로그램이 바나나, 오렌지와 같은 올바른 레이블을 예측

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

노란표를 특징이라고 부르고 빨간 부분을 레이블이라고 부른다.

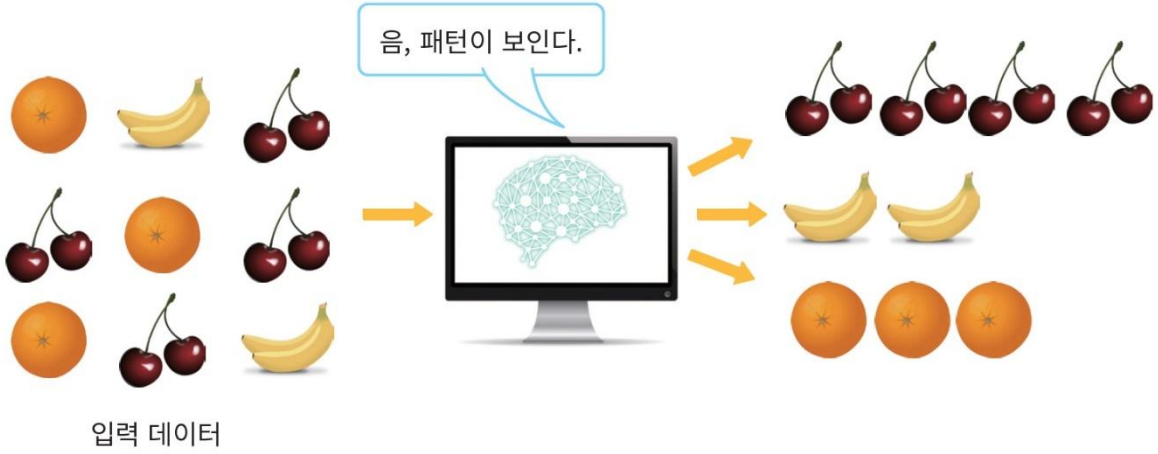
분류

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

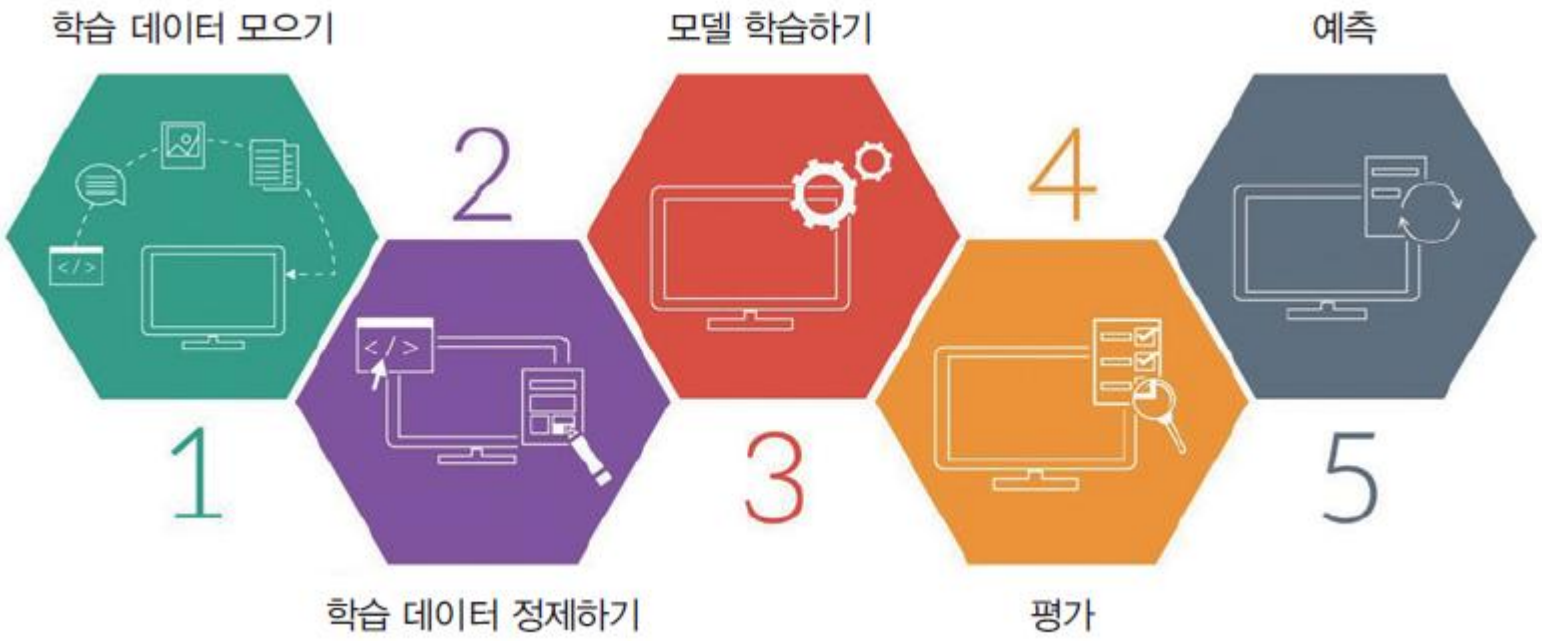


비지도 학습

- 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

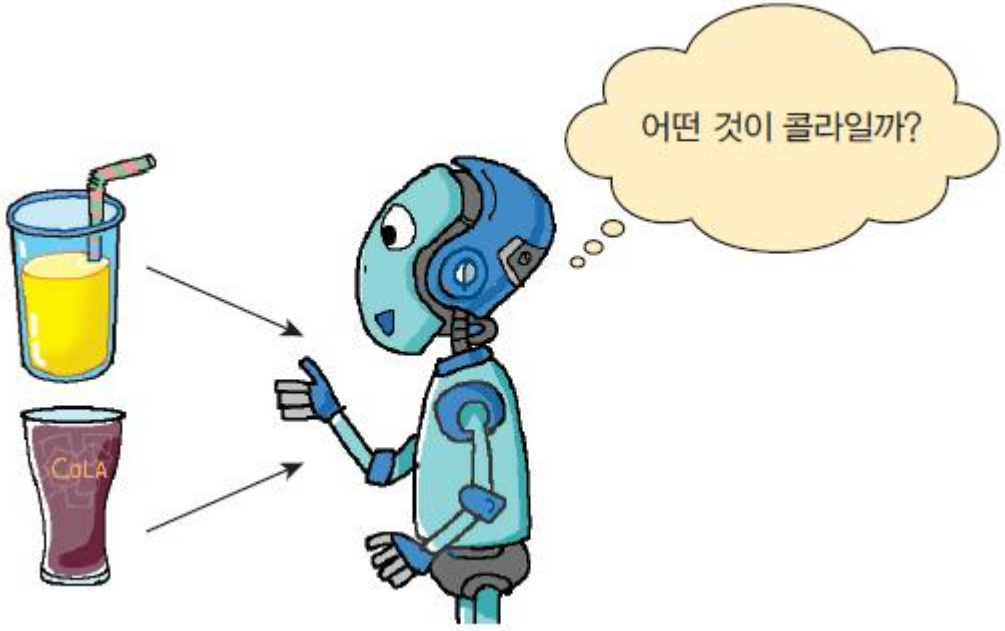


데이터에 있는 불순물을 제거 하기 위함
특징을 추출해준다.
최적화덴 픽처를 사용하기 위해서 selcetion을 해준다,

1.데이터의 잡음 제거
2. fecture extraction
3. fecture selection

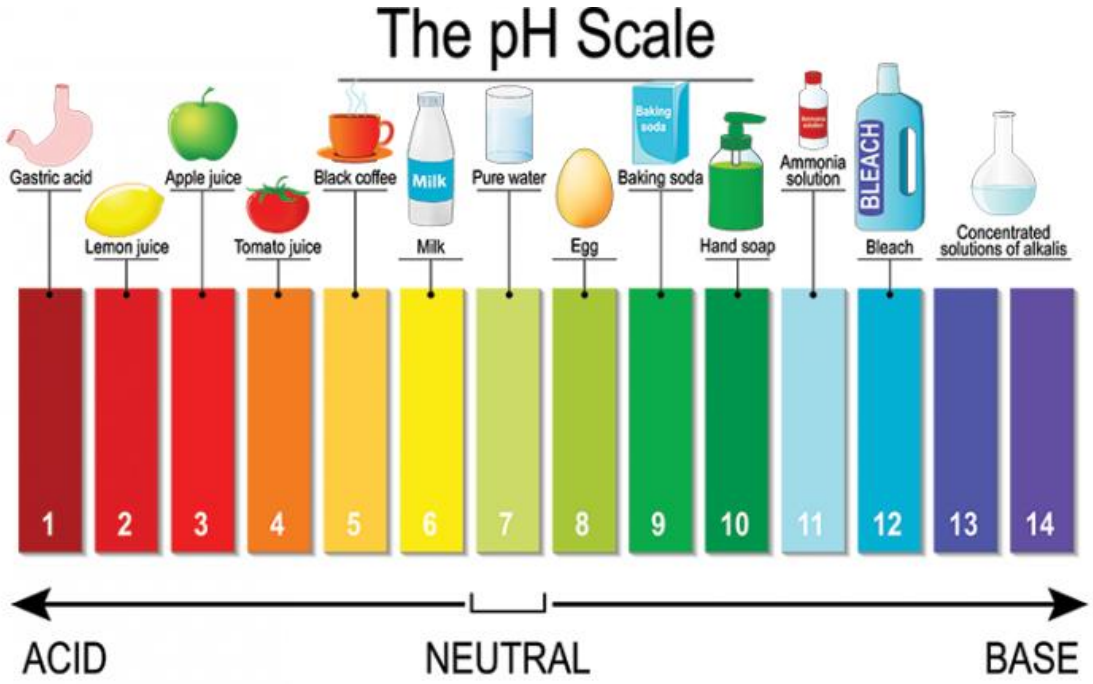
머신러닝의 예

- 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 특징?

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



데이터 수집

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

훈련 데이터의 구조

특징 #1	특징 #2	특징 #3	…	특징 #n	
x_1	x_2	x_3	…	x_n	샘플 #1
x_1	x_2	x_3	…	x_n	샘플 #2
x_1	x_2	x_3	…	x_n	샘플 #3
…					
x_1	x_2	x_3	…	x_n	샘플 #k

fecture 와 레이블을 정확히 나누어 줘야한다.

학습 데이터와 테스트 데이터

- 머신러닝에는 항상 훈련 데이터와 테스트 데이터가 있어야 한다.

2개로 포션을 나눈다.
훈련 데이터는 학습을 하는 데 사용이 된다.

훈련 단계

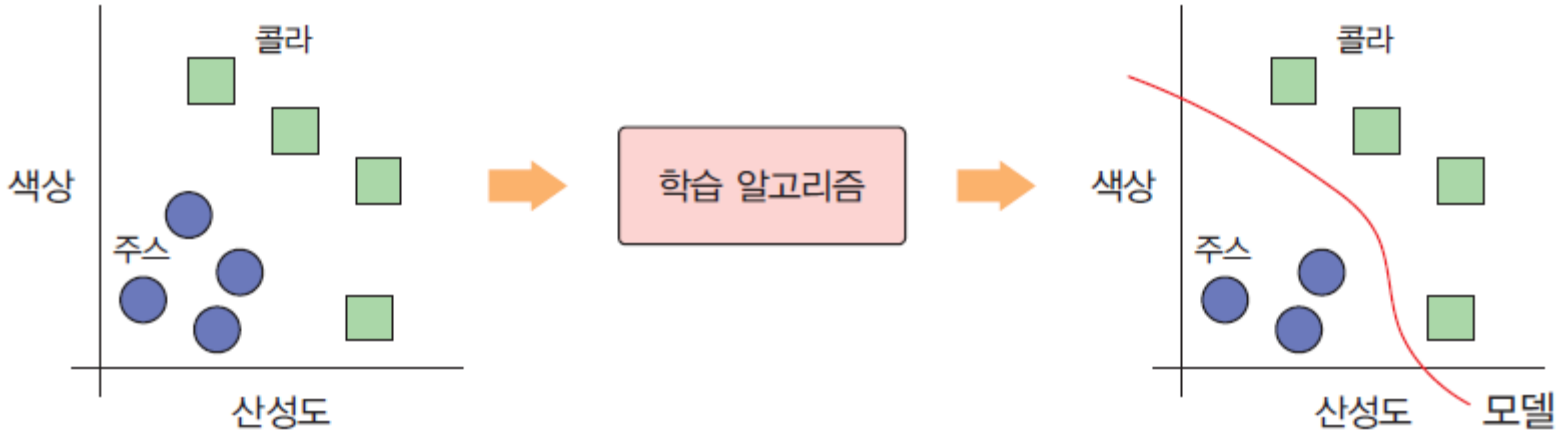


그림 3-10 훈련 단계

테스트 단계

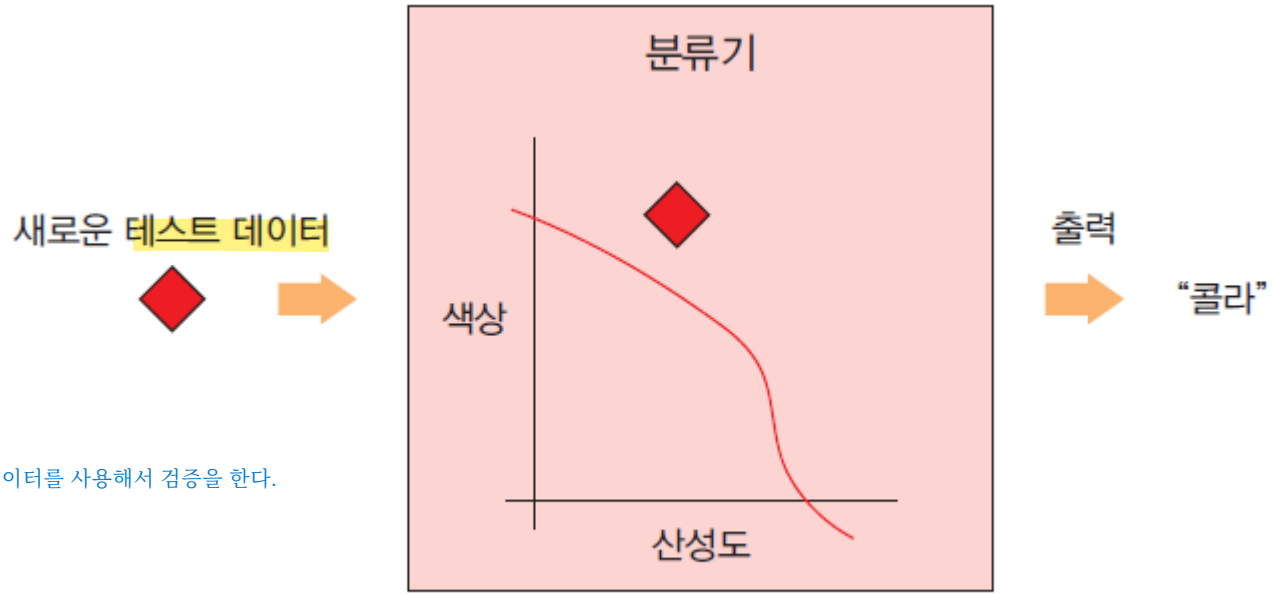
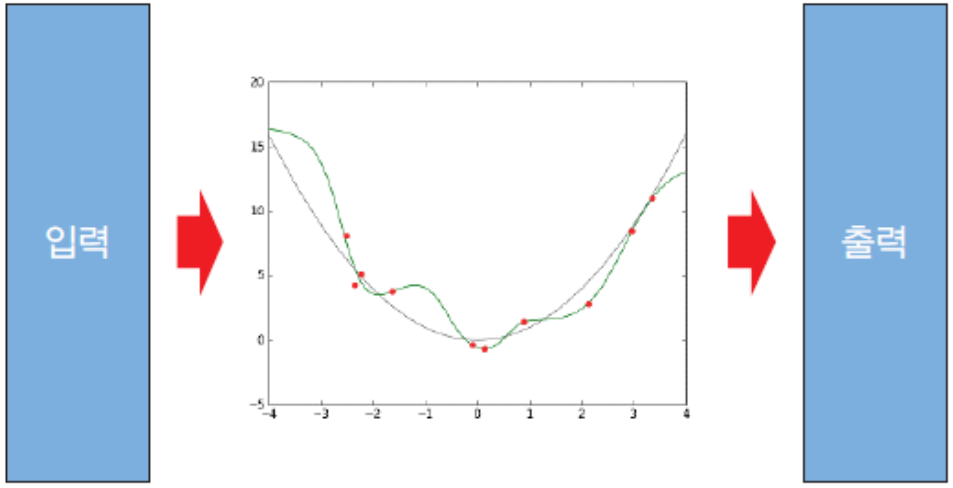


그림 3-11 테스트 단계

테스트 데이터를 사용해서 검증을 한다.

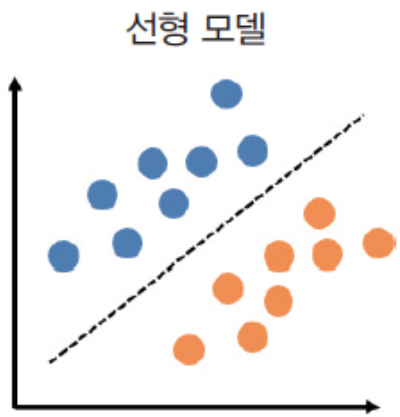
모델 선택

- 다음 단계는 머신러닝 모델을 선택하는 것이다.
- 많은 연구자들이 수년에 걸쳐 만든 많은 모델이 있다.
- 우리의 경우에는 색상과 산성도라는 두 가지 특성만 있으므로 매우 간단한 선형 모델을 사용하자.



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

선형 모델



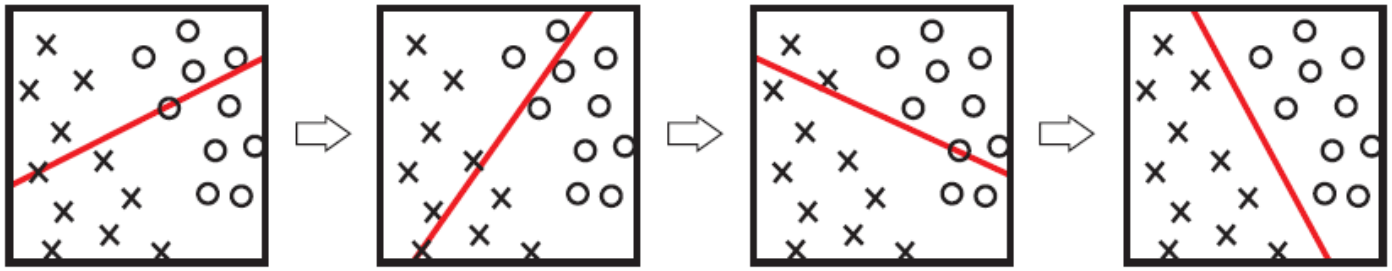
$$y = m * x + b$$

출력 기울기 입력 y-절편

그림 3-12 학습 과정

학습

- 맨 처음에는 W와 b를 임의의 값을 초기화하고 입력값으로 출력을 예측해 본다.
- 이 출력값을 정확한 값과 비교하여 더 정확한 예측을 갖도록 W 및 b의 값을 조정



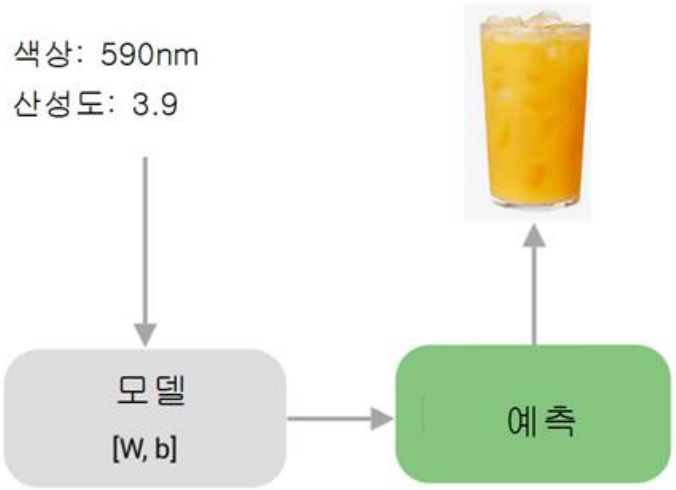
평가

- 학습이 완료되면 모델을 평가하여 모델이 좋은지 나쁜지를 확인해야 한다.
- 테스트 데이터 사용
- 일반적인 훈련 데이터와 테스트 데이터의 비율은 80:20 또는 70:30이다.

standard 가 아니다.

예측

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

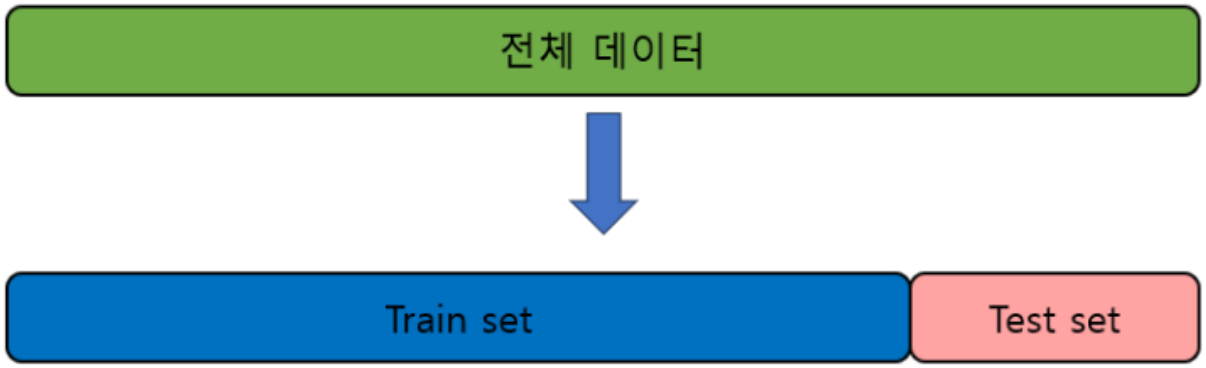


교차검증

- 학습 데이터를 기반으로 테스트 데이터로 해당 모델의 성능을 측정할 경우 문제점
- Problem 1. 테스트의 수가 적으면 성능 평가의 신뢰성이 떨어짐
- Problem 2. 학습 데이터를 줄이고 테스트 데이터를 늘리면 정상적인 학습이 되지 않음

교차검증cross-validation

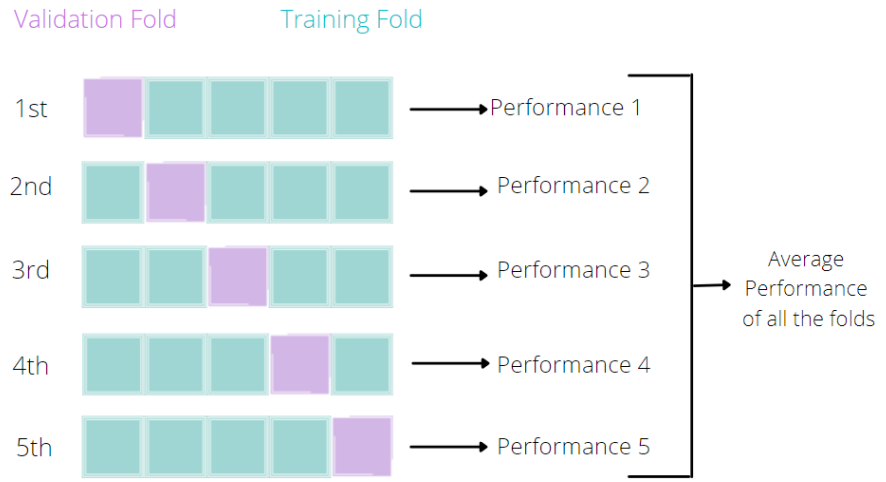
- 교차 검증은 데이터를 여러 부분으로 나누고, 각 부분을 훈련과 테스트 용도로 번갈아 사용하여 모델을 평가하는 방법



교차검증

- K-fold cross-validation
  - 데이터를 K (K는 실수)개의 동일한 크기로 나누어 하위 데이터 세트로 나눔
  - 이후 K번 반복하여 ML 알고리즘을 활용한 학습 및 테스트를 수행함
  - 이때 [K-1]개의 하위 데이터 세트가 학습에 활용, 나머지 1개 데이터 세트가 테스트에 활용됨
  - K번 반복한 성능을 평균하여 -〉 최종 결과로 보고

    [중요] K-fold cross-validation 수행 시, 학습과 테스트 데이터 세트는 완벽히 분리 되어야 함

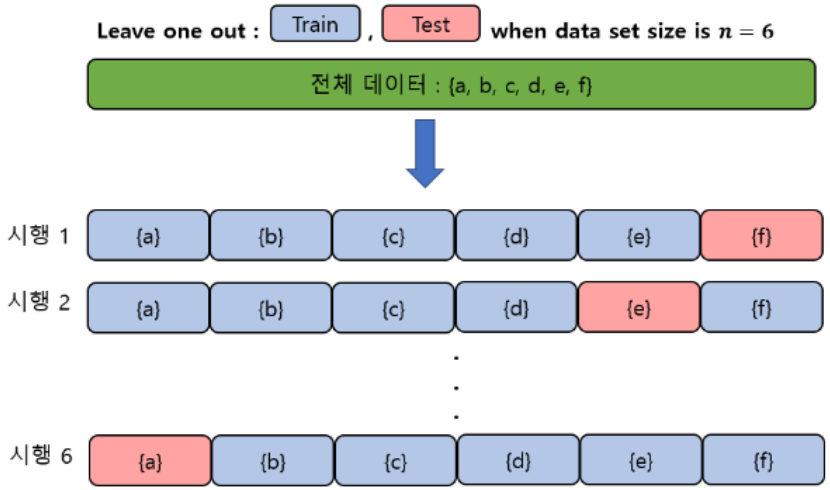


교차검증

- **Leave-one-out cross-validation** 하나 샘플마다 다 테스트로 사용을 한다.
  - 데이터의 샘플의 개수가 n일 경우, n 번 반복하여 ML 알고리즘을 학습함
  - 학습 시 [n-1]개의 하위 데이터 세트가 활용, 나머지 1개 데이터 세트가 테스트에 활용됨
  - n번 반복한 성능을 평균하여 -〉 최종 결과로 보고

  [K-fold와의 차이점] 샘플의 개수가 적을 때(데이터 양이 부족할 때) 주로 사용함 훈련 데이터 양이 충분하지 않으므로 사용을 한다.

  [중요] K-fold cross-validation 와 마찬가지로, 학습과 테스트 데이터 세트는 완벽히 분리 되어야 함



잘 알려진 데이터 세트

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

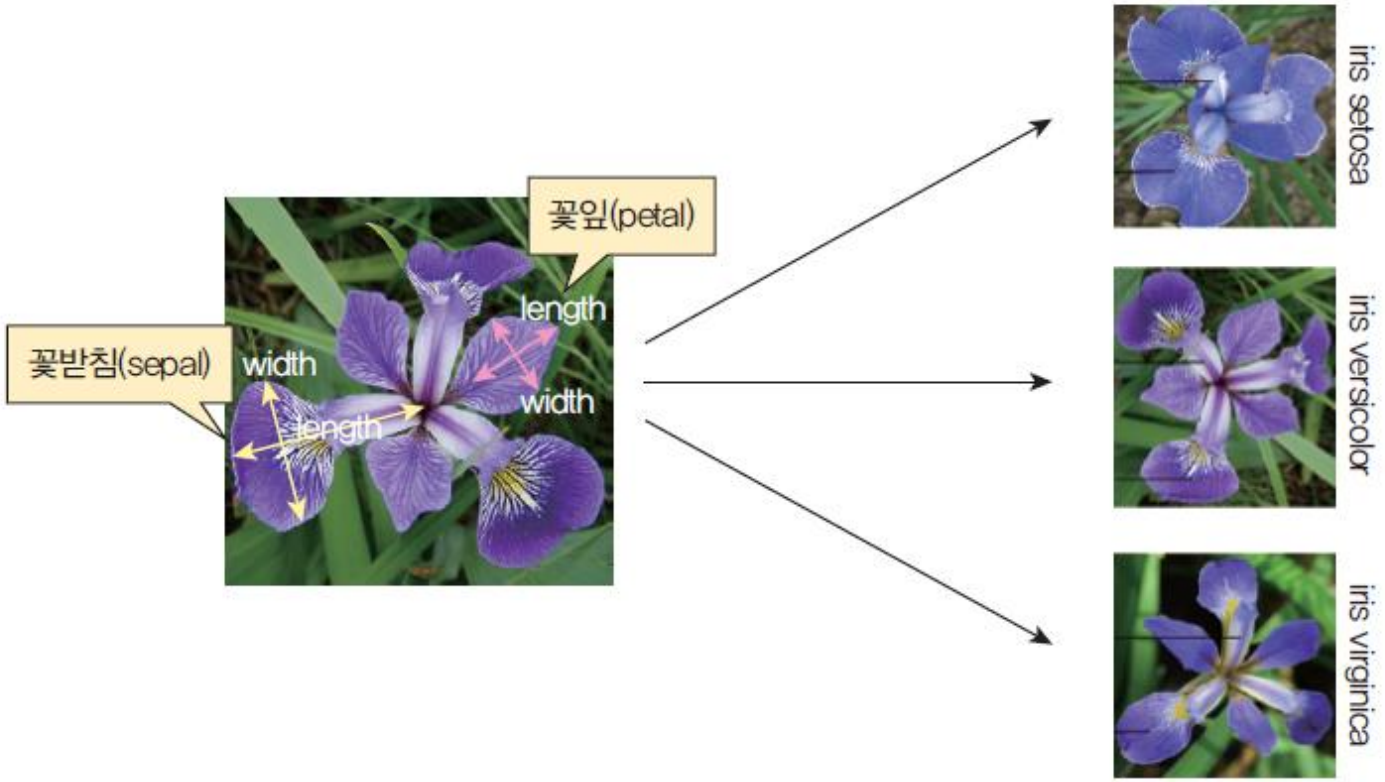


그림 3-13 붓꽃 데이터 세트

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,
...
```

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.

k값에 따라서 결과가
변하는 단점이 있다.
k값을 정해주는 standard가 없다.

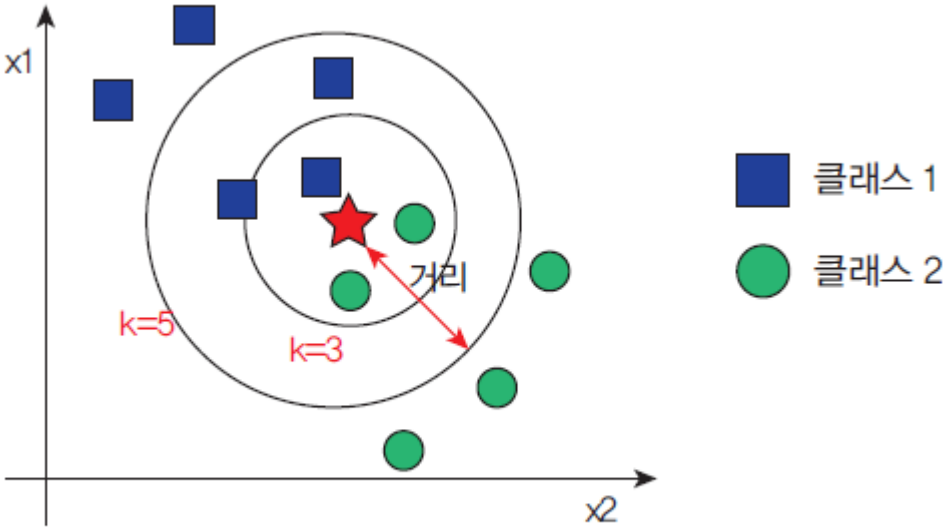


그림 3-15 kNN 알고리즘에서의 분류

학습

```
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
```

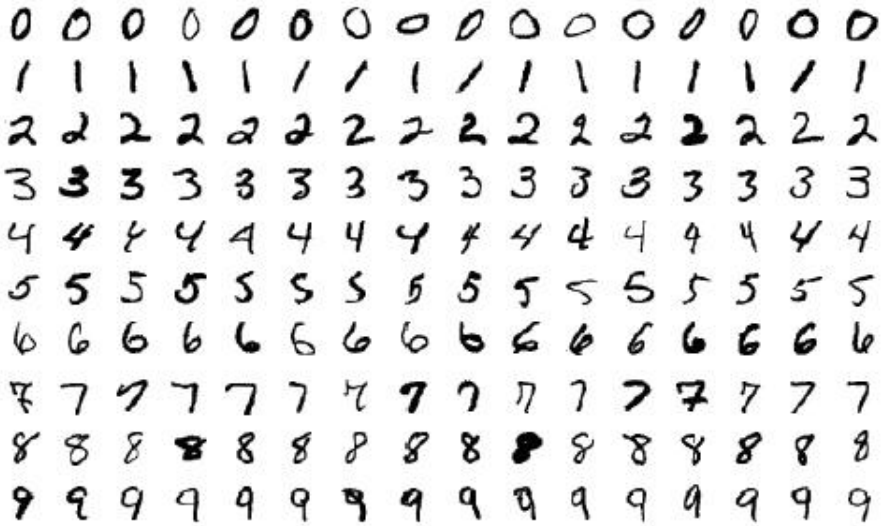
예측

```
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
- 우리는 sklearn을 사용하여 필기체 숫자 이미지를 인식하는 프로그램을 작성해보자.



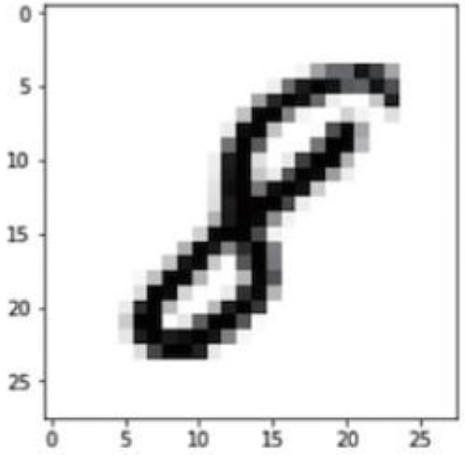
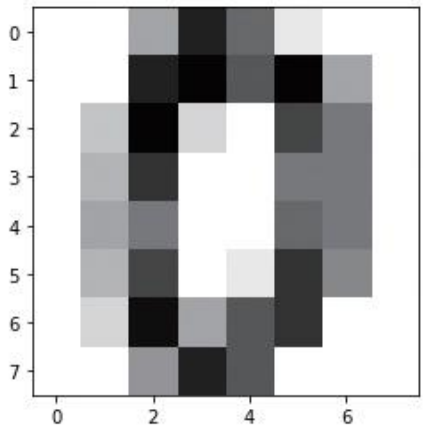


그림 3-16 MNIST 데이터

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))
```

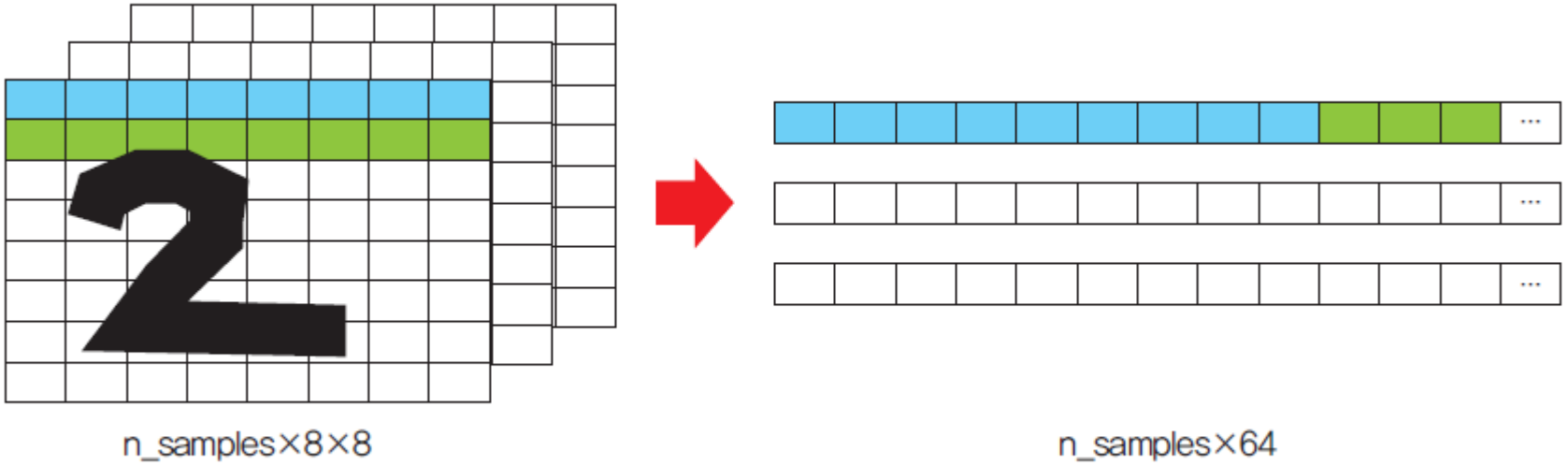


그림 3-17 평탄화

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)
```

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

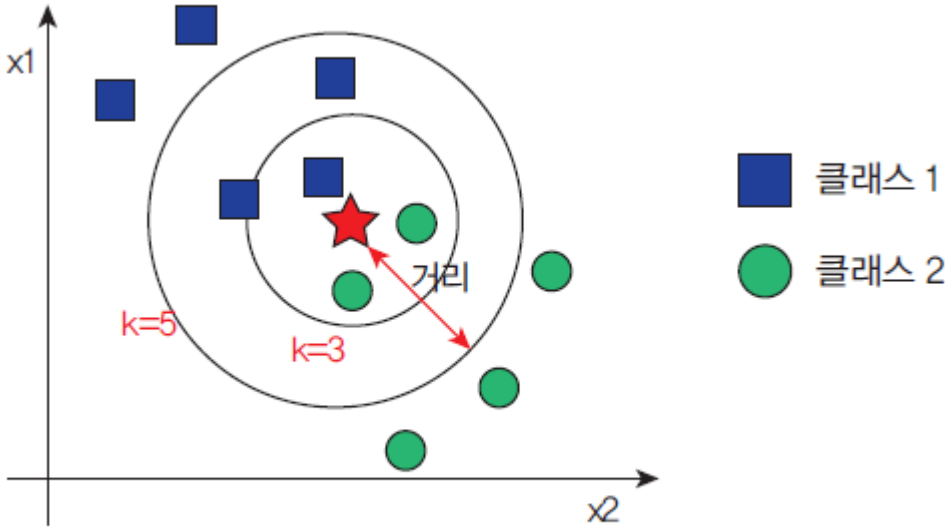


그림 3-15 kNN 알고리즘에서의 분류

예측

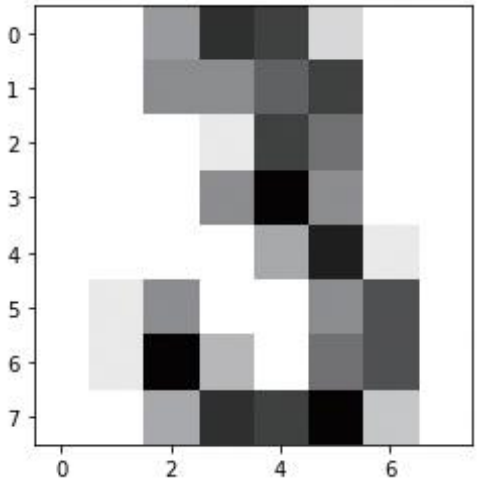
```
# 테스트 데이터로 예측해본다.
y_pred = clf.predict(X_test)
# 정확도를 계산한다.

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]
```

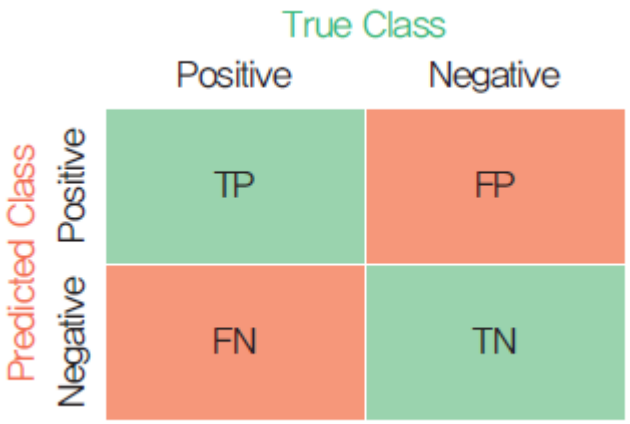


머신러닝 알고리즘의 성능평가

- 정확도

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

- 혼동행렬



혼동 행렬

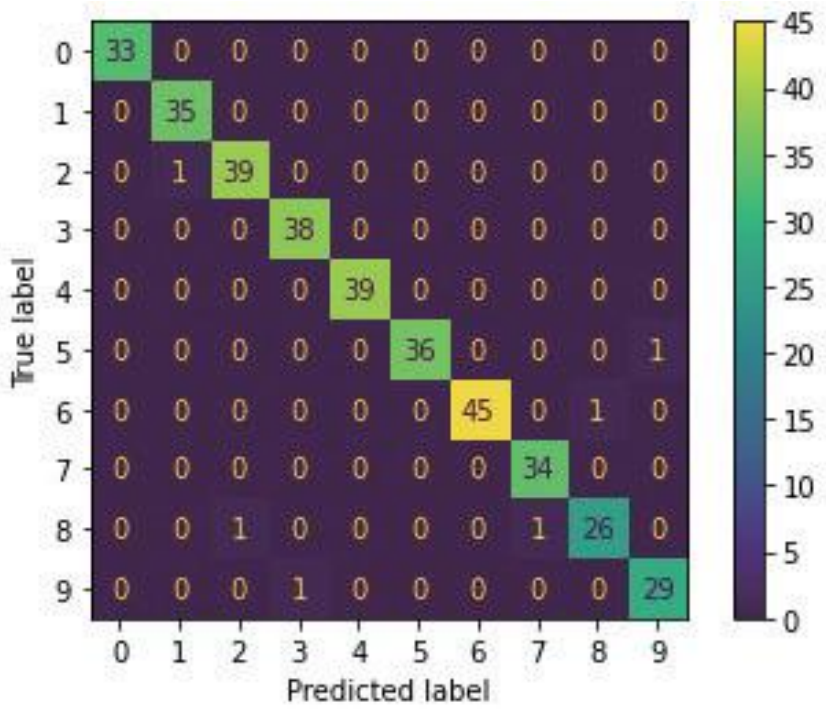
- 긍정을 긍정으로 올바르게 예측하면 TP(True Positive)라고 한다. 앞에 True가 붙으면 예측과 실제가 같다는 의미이다.
- 긍정을 부정으로 잘못 예측하면 FN(False Nagative)라고 한다. 앞에 False가 붙으면 예측과 실제가 틀리다는 의미이다.
- 부정을 긍정으로 잘못 예측하면 FP(False Positive)라고 한다.
- 부정을 부정으로 올바르게 예측하면 TN(True Nagative)라고 한다.

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

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")
```

	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?

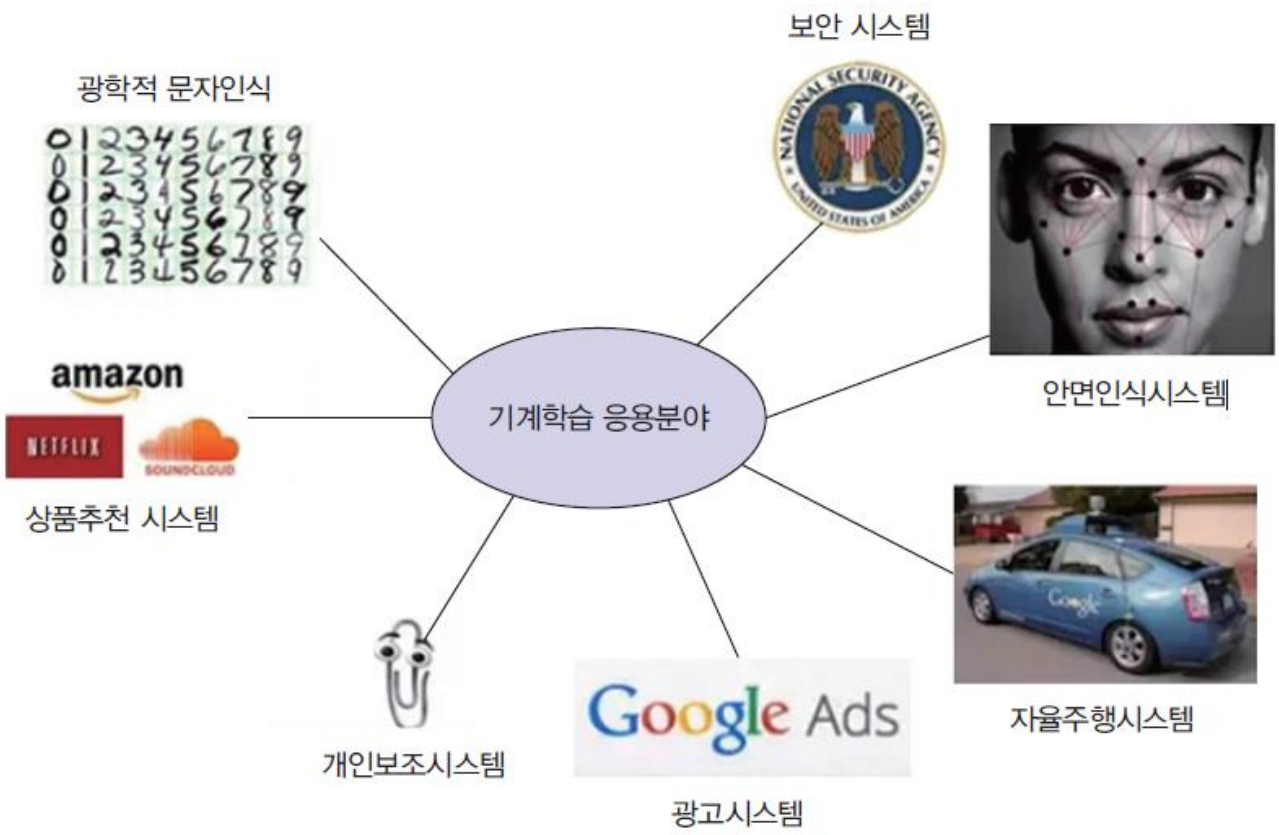


그림 3-18 머신러닝의 응용 분야

머신러닝은 어디에 이용되는가?

- 이들 분야들은 살펴보면 복잡한 데이터들이 있고, 이들 데이터에 기반하여 결정을 내려야 하는 분야이다.
  - 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

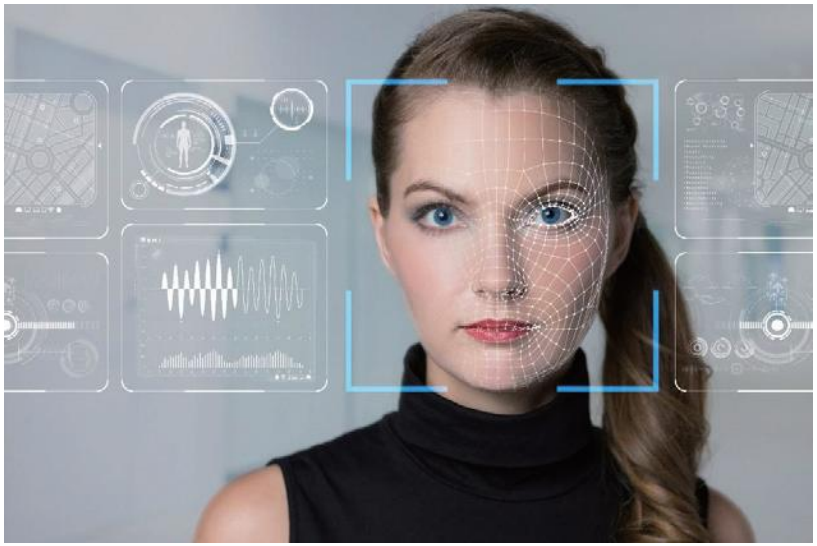
- 첫 번째는 프로그래밍 시간을 줄일 수 있다는 점이다.
- 예를 들어서 맞춤법 오류를 수정하는 프로그램을 개발한다고 하자.
  - 전통적인 방법: 많은 맞춤법 규칙을 이용하여 작성할 수 있다. -> 상당한 시간이 필요
  - 머신러닝 이용: 많은 예제만 있다면 학습시켜서 빠른 시간 안에 신뢰성 있는 프로그램을 완성할 수 있다.

프로그래머로서 머신러닝의 실용적인 가치

- 두 번째로 맞춤형 제품을 쉽게 개발할 수 있다.
- 예를 들어서 여러분이 한국어 맞춤법 수정 프로그램이 작성하여 가지고 있다고 하자. 제품이 성공적이어서 30개국 언어 버전으로 확장하려고 한다.
  - 전통적인 방법: 각 언어마다 새로 작성하려면 수년 이상의 엄청난 시간이 필요하다.
  - 머신러닝 이용: 너무나도 쉽다 예제만 있으면 된다.

프로그래머로서 머신러닝의 실용적인 가치

- 세 번째로 머신러닝은 프로그래머로 시도할 알고리즘이 떠오르지 않는 문제들을 해결할 수도 있다.
- 예를 들어서 컴퓨터가 사람의 얼굴을 인식하는 프로그램을 작성
  - 전통적인 방법: 이런 문제를 작성하려면 컴퓨터 시각 분야의 수많은 지식과 경험이 필요한 작업이다.
  - 머신러닝 이용: 프로그램에 수많은 예제만 보여주기만 하면 문제가 해결된다. 편리하지 않은가?



데이터 라벨링

- 머신러닝 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.

Summary

- 머신러닝(machine learning)은 인공지능의 한 분야로, 컴퓨터에 학습 기능을 부여하기 위한 연구 분야이다.

- 머신러닝은 “교사”의 존재 여부에 따라 크게 지도 학습과 비지도 학습으로 나누어진다. 또 강화학습도 있다.
- 지도 학습은 크게 회귀와 분류로 나눌 수 있다. 회귀(regression)는 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것이다. 분류(classification)는 입력을 두 개 이상의 유형으로 분할하는 것이다.
- 비지도 학습은 “교사” 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다. 비지도 학습에서는 데이터들의 상관도를 분석하여 유사한 데이터들을 모으게 된다.
- 강화 학습에서는 컴퓨터가 어떤 행동을 취할 때마다 외부에서 처벌이나 보상이 주어진다.