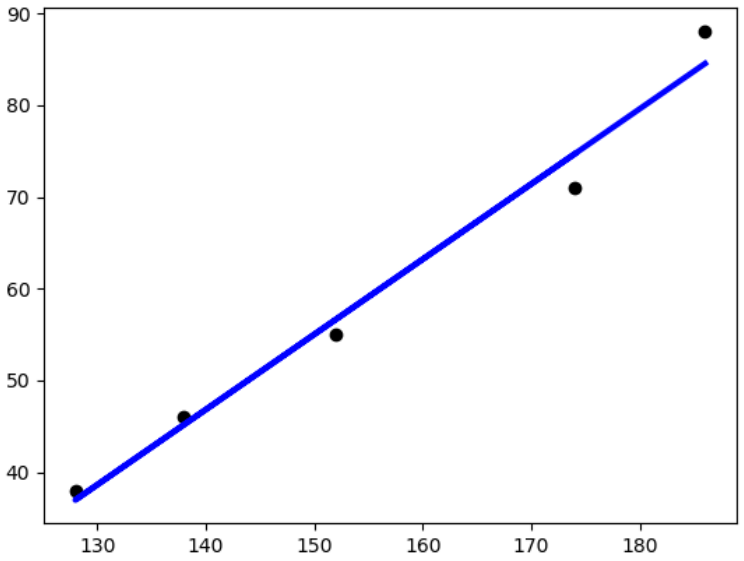
16장 파이썬을 이용한 기계 학습

학습 목표

- 기계 학습의 개념에 대하여 살펴본다.
- 선형 회귀 문제를 sklearn 라이브러리를 이용하여 실습해본다.
- XOR 문제를 케라스 라이브러리를 이용하여 실습해본다.
- 숫자 인식 프로그램을 케라스 라이브러리를 이용하여 실습해본다.

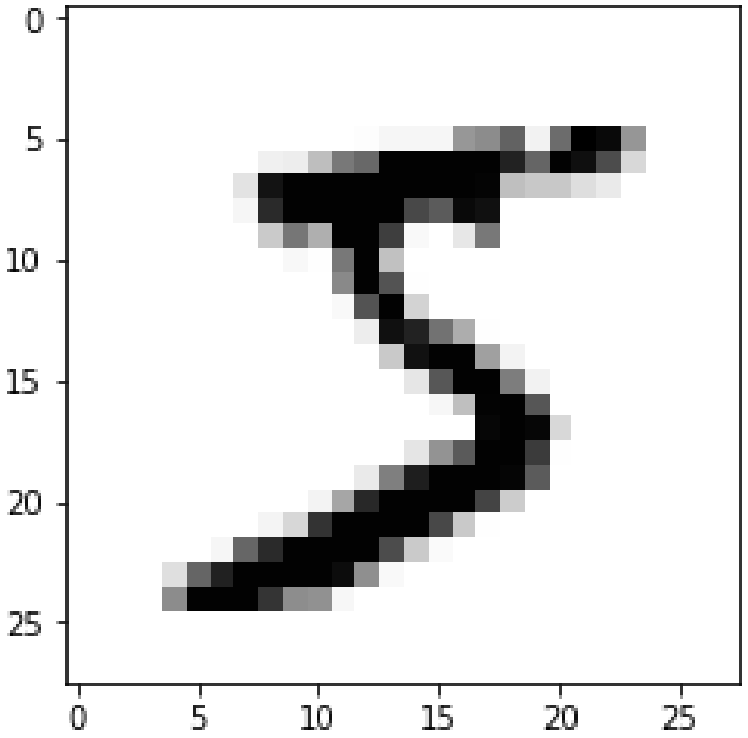
이번 장에서 만들 프로그램

- 선형 회귀 분석 프로그램을 sklearn 라이브러리를 이용하여 작성해 보자.



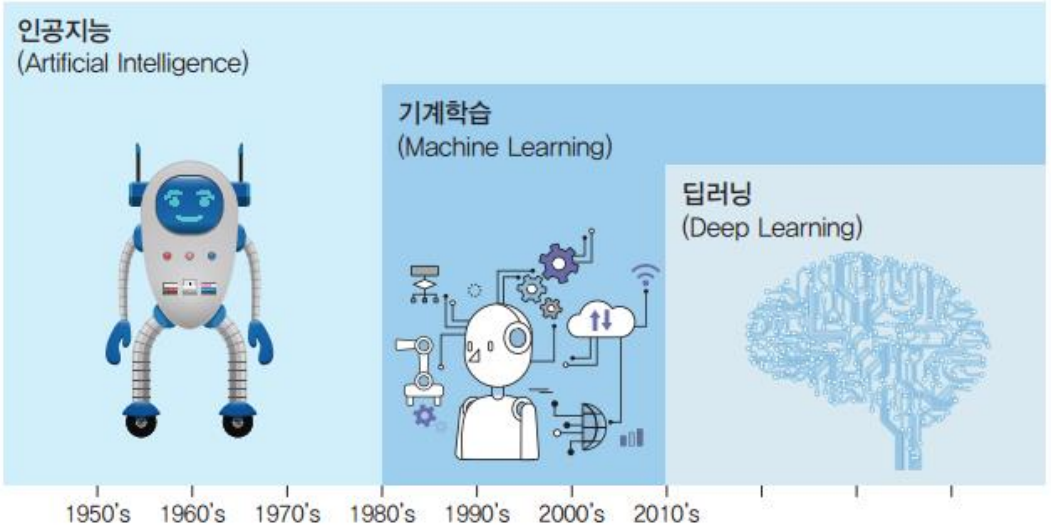
이번 장에서 만들 프로그램

- 필기체 숫자 인식 프로그램을 케라스 라이브러리를 이용하여 작성해 보자.



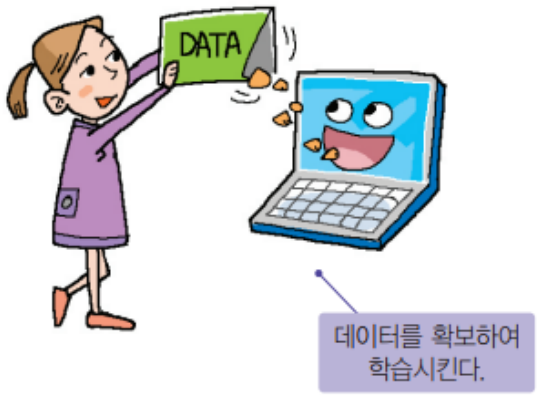
기계학습(machine learning)

- 인공 지능의 한 분야로 컴퓨터에 학습 기능을 부여하기 위한 연구분야
- 컴퓨터가 주어진 데이터를 학습하는 알고리즘을 연구한다. 학습할 수 있는 데이터가 많아지면 알고리즘 성능이 향상된다.
- 기계 학습은 어떤 문제에 대하여 명시적 알고리즘을 설계하고 프로그래밍하는 것이 어렵거나 불가능한 경우에 주로 사용된다.



기계 학습이 중요하게 사용되는 분야

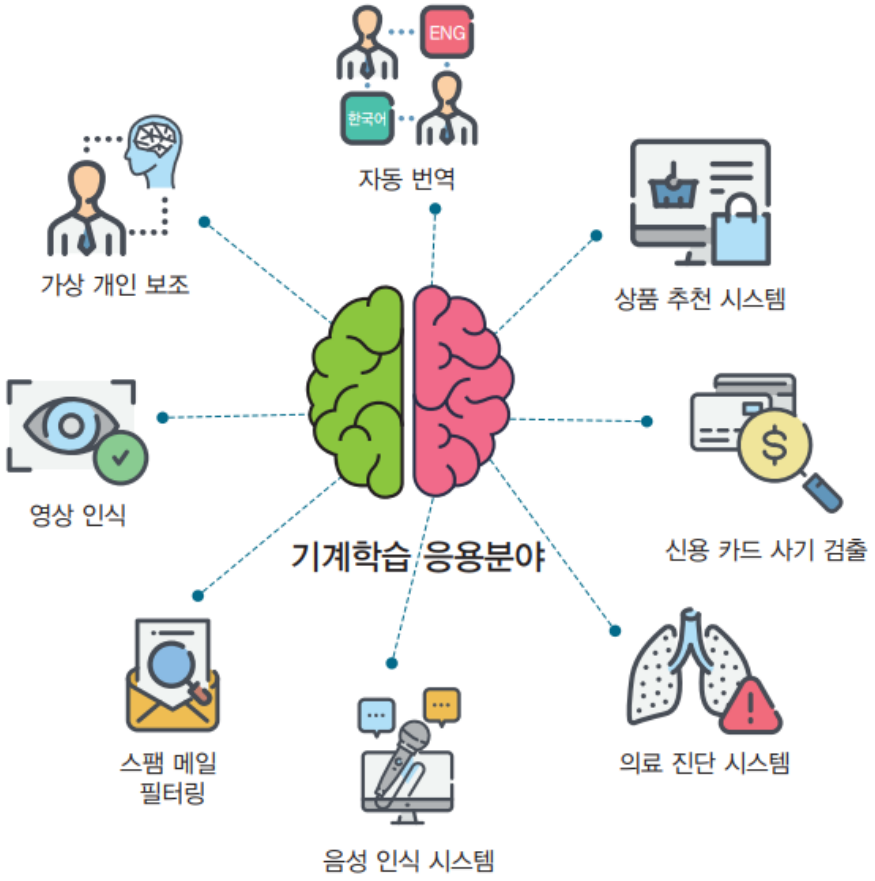
- 기계 학습은 빅데이터(big data)와 밀접한 관련이 있다. 학습을 시키려면 많은 데이터가 필수적이기 때문이다.
- 기계 학습은 문제를 해결하는데, 많은 경우가 있어서 각각의 경우를 정확하게 처리하는 것이 불가능한 경우(예: 바둑, 스팸 이메일, 자율주행 자동차)에 필요
- 이런 경우에 기계 학습이 제격. 많은 데이터를 확보한 후 학습시키면 프로그램이 저절로 어떤 규칙을 만들어 낸다.



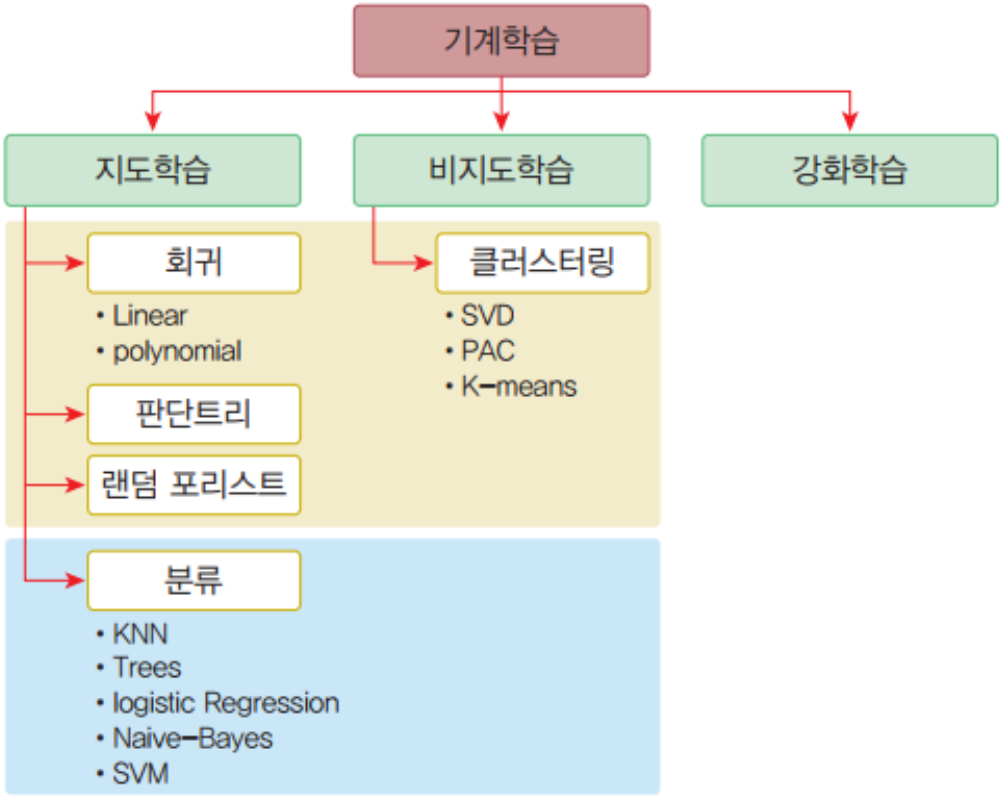
기계 학습이 중요하게 사용되는 분야

- 음성 인식처럼 프로그램으로 작성하기에는 규칙과 공식이 너무 복잡할 때(인간도 잘 모르는 분야이다.)
- 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 여부를 판단할 때
- 구매자가 클릭할 확률이 가장 높은 광고가 무엇인지를 알아내고 싶을 때, 사용자가 선호하는 상품이나 비디오를 자동으로 추천하고 싶을 때
- 영상 인식 : 얼굴 인식, 움직임 인식, 객체 감지, 자율 주행에 나타나는 다양한 상황을 처리하고 싶을 때, 이미지를 자동으로 분류하고 싶을 때, 이미지 데이터베이스 중에서 특정 이미지를 탐색하고 싶을 때

기계 학습이 중요하게 사용되는 분야

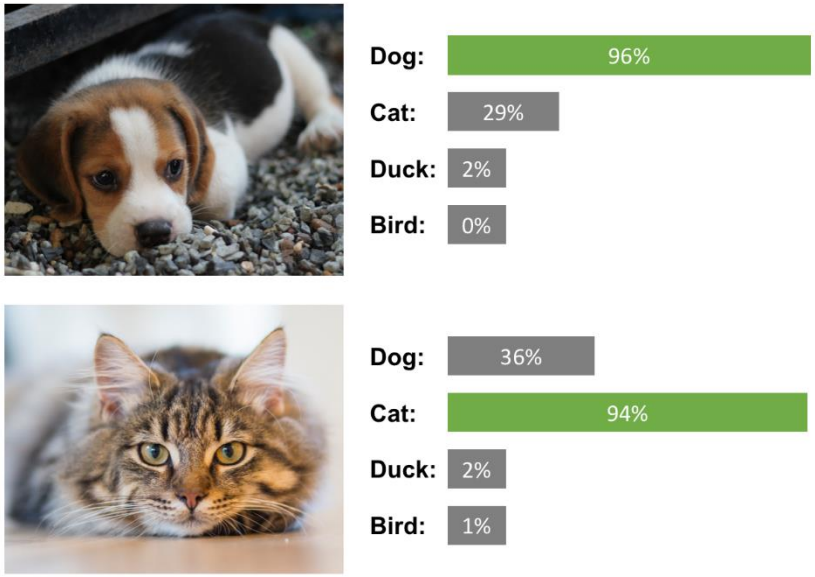


기계학습의 분류


기계학습
지도학습
비지도학습
강화학습
회귀
• Linear
• polynomial
판단트리
랜덤 포리스트
분류
• KNN
• Trees
• logistic Regression
• Naive-Bayes
• SVM
클러스터링
• SVD
• PAC
• K-means

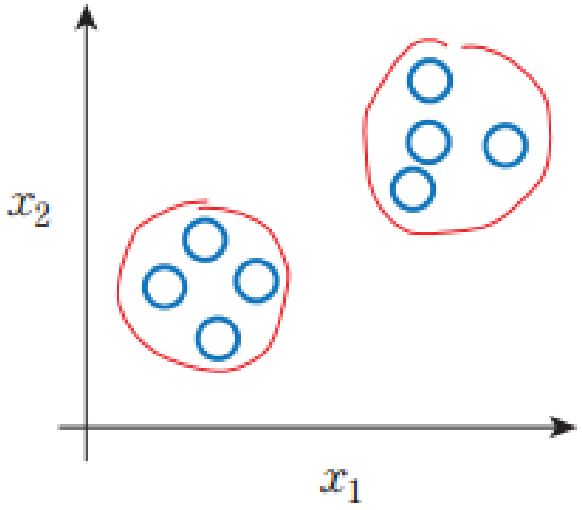
기계학습의 분류

- 지도 학습(Supervised Learning) :
  - 컴퓨터는 '교사'에 의해 주어진 예제와 정답(레이블)을 제공받는다.
  - 입력을 출력에 매핑하는 일반적인 규칙을 학습하는 것.
  - 예: 개와 고양이 사진의 구별 학습



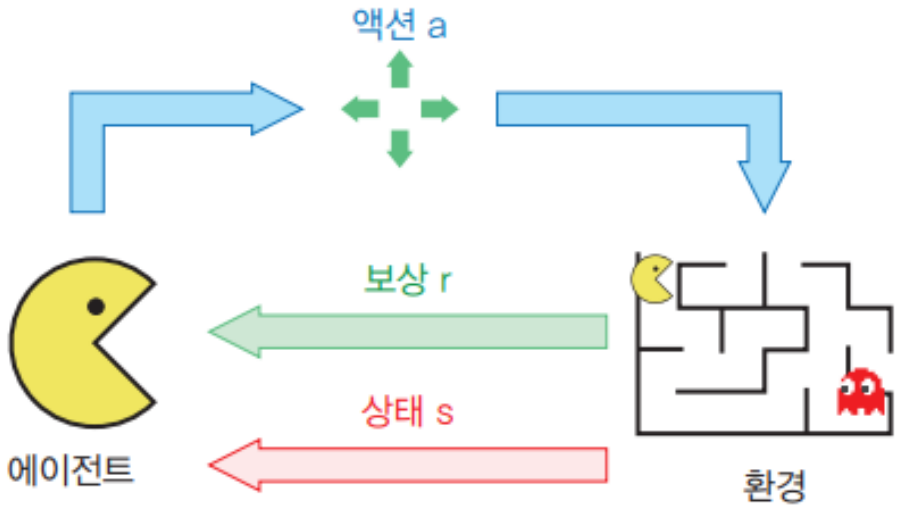
기계학습의 분류

- 비지도 학습(Unsupervised Learning)
  - 외부에서 정답(레이블)이 주어지지 않고 학습 알고리즘이 스스로 입력에서 어떤 구조를 발견하는 학습
  - 데이터의 숨겨진 패턴을 발견할 수 있다.
  - 예: 클러스터링(clustering) – 구글에서 비슷한 뉴스를 자동 그룹핑



기계학습의 분류

- 강화 학습(Reinforcement Learning)
  - 보상이나 처벌 형태의 피드백으로 학습이 이루어지는 기계 학습
  - 주로 차량 운전이나 상대방과의 경기 같은 동적인 환경에서 프로그램의 행동에 대한 피드백만 제공
  - 예: 바둑에서 어떤 수를 두어서 승리하였다면 보상이 주어진다.

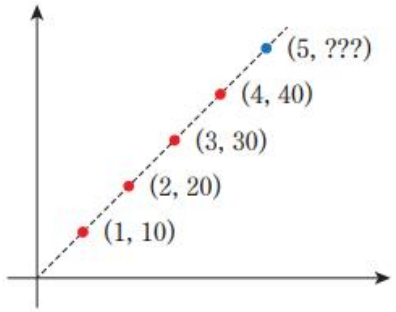


지도학습

- 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 출력값을 합리적으로 예측하는 것
- 입력(x)과 출력(y)이 주어질 때, 입력에서 출력으로의 매핑 함수를 학습하는 것

$$y = f(x)$$

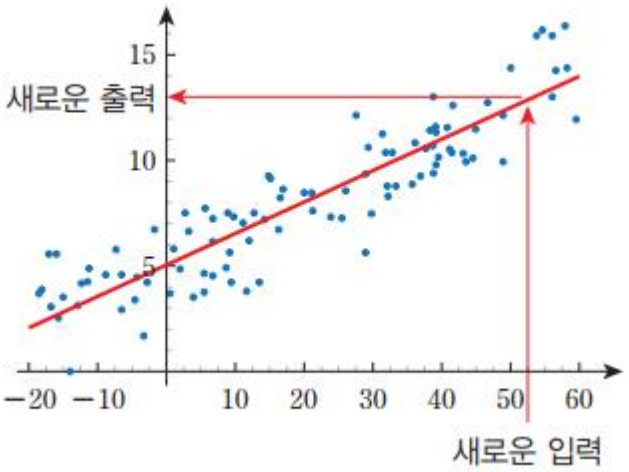
- 예 : 입력 데이터로 직선 y=10x 위에 있는 점 (1, 10), (2, 20), (3, 30), (4, 40)들이 주어져 있다고 하자. 학습이 끝난 후에 x=5를 입력하면 컴퓨터가 y=50이라는 답을 할 수 있도록 만드는 것이 지도 학습이다.



- 선형 회귀와 분류로 나눌 수 있다.

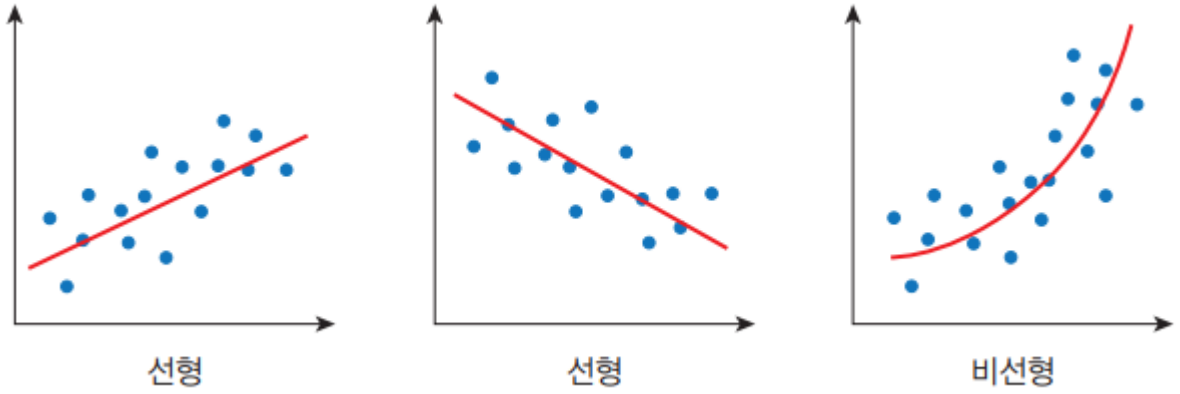
지도 학습: 회귀(regression)

- 예제 데이터들을 2차원 공간에 찍은 후에, 이들 데이터들을 가장 잘 설명하는 직선이나 곡선을 찾는 문제
- 학습이 끝난 후에 새로운 데이터가 들어오면 이 직선이나 곡선을 이용하여 출력값을 예측하게 된다.
- y=f(x)에서 입력과 출력을 보면서 함수 f(x)를 예측하게 된다.



지도 학습: 회귀(regression)

- 주식의 추세선을 예측하거나, 흡연과 사망률 사이의 관계, 온도 변화나 전력 수요 변동 등의 연속적인 응답을 예측하는데 사용
- 선형 회귀도 있고 비선형 회귀도 있다.



지도 학습: 분류(classification)

- 식 y = f(x)에서 출력 y가 이산적(discrete)인 경우에 이것을 분류 문제(또는 인식 문제)라고 부른다.
- 분류 - 입력을 2개 이상의 클래스로 나누는 것. 예 : 강아지 또는 고양이 분류, 이메일에서 스팸 메일을 찾는 것.

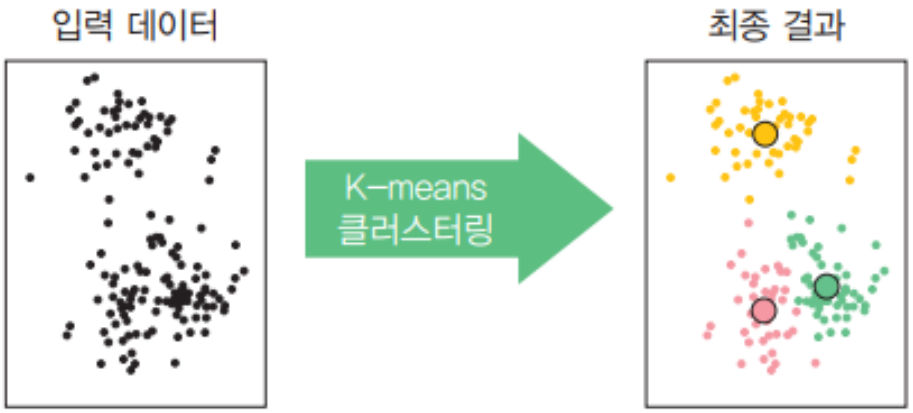
- 신경망(딥러닝), kNN(k-nearest neighbor), SVM(Support Vector Machine), 의사 결정 트리 등

비지도학습

- ‘교사’ 없이 컴퓨터가 스스로 입력들을 분류하는 것을 의미한다. 식 y = f(x)에서 정답인 레이블 y가 주어지지 않는 것이다.
- 비지도 학습은 정답이 없는 문제를 푸는 것과 같으므로 학습이 맞게 되었는지 확인할 수 없다. 하지만 데이터들의 상관도를 분석하여 유사한 데이터들을 모을 수는 있다.

비지도학습

- 가장 대표적인 비지도 학습이 클러스터링(군집화, clustering).
- 클러스터링이란 입력 데이터 간의 거리를 계산하여서 입력을 몇 개의 군집으로 나누는 방법이다. K-means 클러스터링이 가장 고전적인 클러스터링 방법.



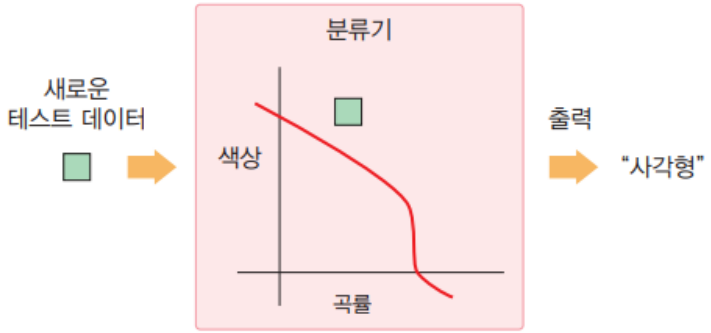
기계 학습의 요소들

- 특징(feature)
  - 학습 모델에게 공급하는 입력이다. 가장 간단한 경우에는 입력 자체가 특징이 된다.
  - 예 : 이메일이 스팸인지 아닌지를 결정할 때의 특징(feature)
    - 이메일에 “검찰”이라는 문자 포함 여부(yes 또는 no)
    - 이메일에 “광고”, “선물 교환권”이나 “이벤트 당첨” 문자열 포함 여부(yes 또는 no)
    - 이메일의 제목이나 본문에 있는 ‘★’과 같은 특수 기호의 개수 (정수)

기계 학습의 요소들

- 학습 데이터와 테스트 데이터
  - '원'과 '사각형' 레이블이 붙어 있는 학습 데이터로 시스템을 학습시킨다.

  - 학습이 끝나고 나면 한 번도 본 적이 없는 새로운 데이터로 시스템을 테스트(예측)한다.



선형 회귀 분석

- 직선의 방정식

  $f(x) = mx + b.$ m은 기울기, b 는 절편

- 선형 회귀는 입력 데이터를 가장 잘 설명하는 직선의 기울기와 절편을 찾는 문제이다
- 기계 학습에서는 기울기 대신에 가중치(weight), 절편 대신에 바이어스(bias)라고 한다.

  $f(x) = Wx + b$

- 선형 회귀 알고리즘은 데이터 요소에 여러 직선을 맞추어 본 후에 가장 적은 오류를 발생시키는 직선을 찾는 것이다.

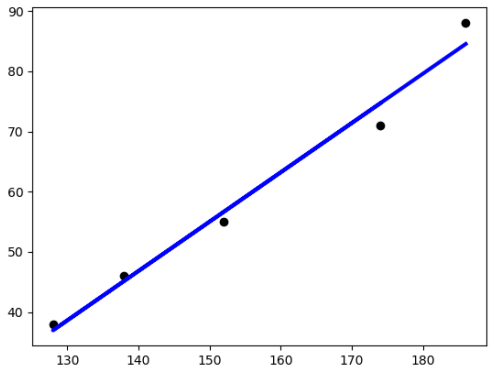
선형 회귀 분석 예제

- 표에 있는 키와 몸무게 데이터를 이용하여 선형 회귀시스템을 학습시킨 후에 새로운 키를 넣었을 때 몸무게를 예측할 수 있는 시스템을 만들자.
- 사이킷런(sklearn) 라이브러리를 사용하여 선형 회귀 분석을 구현.

선형 회귀 분석 예제

p697_linear_reg.py

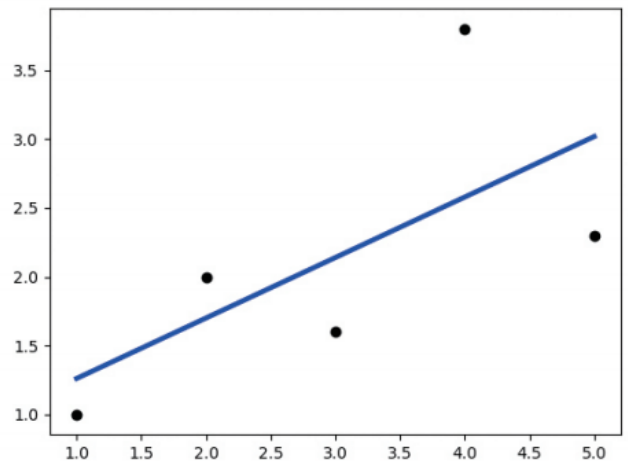
```
import matplotlib.pylab as plt

# sklearn 라이브러리에서 linear_medel 모듈을 가져오기
from sklearn import linear_model

# linear_medel 모듈에 포함되어 있는 선형회귀 객체인 LinearRegression() 생성자 호출
reg = linear_model.LinearRegression()          #  생성

X = [[174], [152], [138], [128], [186]]        # 학습 예제
y = [71, 55, 46, 38, 88]                       # 정답

# X와 y 값으로 학습시키기
reg.fit(X, y)

# 학습한 결과로 구한 기울기, 절편을 확인
print(reg.coef_)                 # 직선의 기울기
print(reg.intercept_)            # 직선의 절편
print(reg.score(X, y))           # 학습 점수

…
…
```



Lab: 선형 회귀 실습

- 다음과 같은 데이터를 가장 잘 설명하는 직선을 찾기,

X	y
1.0	1.0
2.0	2.0
3.0	1.6
4.0	3.8
5.0	2.3

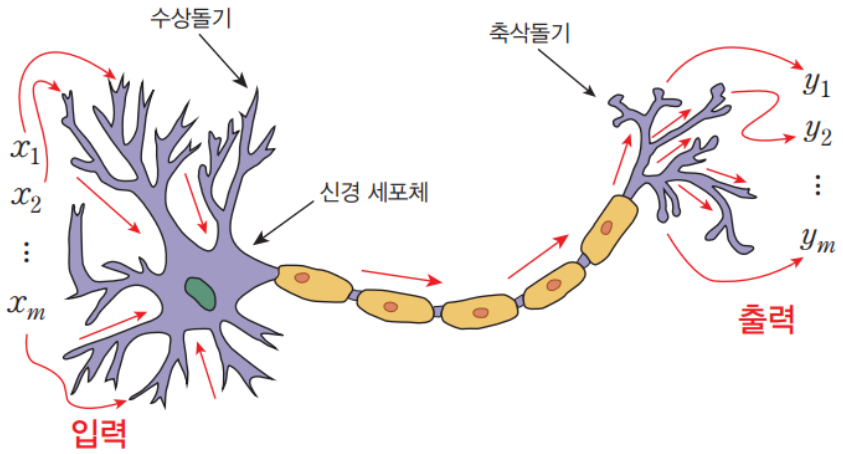


p700_linear_reg.py

```
import matplotlib.pylab as plt
from sklearn import linear_model

reg = linear_model.LinearRegression()

X = [[1.0], [2.0], [3.0], [4.0], [5.0]]
y = [1.0, 2.0, 1.6, 3.8, 2.3 ]
reg.fit(X, y)                                    # 학습
…
```

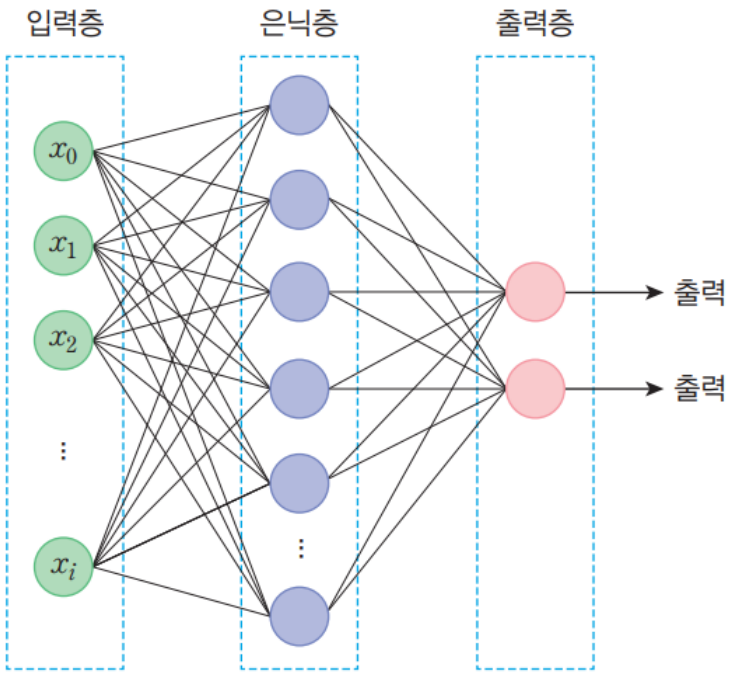
신경망

- 최근에 많은 인기를 끌고 있는 딥러닝(deep learning)의 시작은 1950년대부터 연구되어 온 인공 신경망(artificial neural network: ANN)이다. 인공신경망은 생물학적인 신경망에서 영감을 받아서 만들어진 컴퓨팅 구조이다.
- 인간의 두뇌는 뉴론(neuron, 신경세포)으로 이루어져 있다. 뉴론은 수상돌기를 통하여 주위의 뉴론들로부터 신경 자극을 받아서 세포체에서 어떤 처리를 한 후 축삭돌기를 통하여 다른 세포들로 출력을 내보낸다.



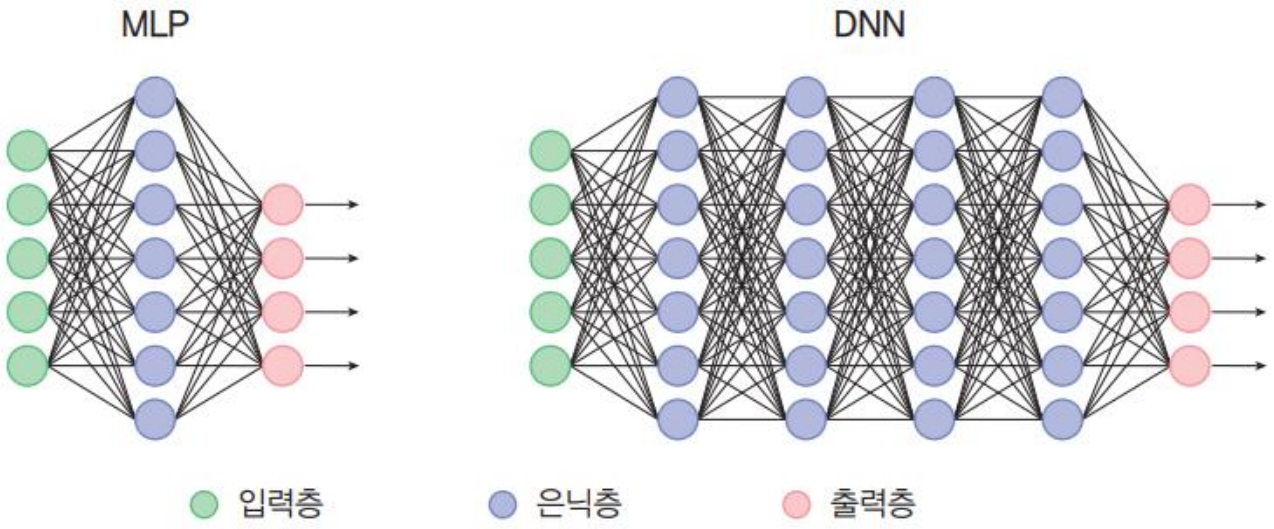
신경망

- 다층 퍼셉트론(multilayer perceptron: MLP) : 입력층과 출력층 사이에 은닉층(hidden layer)을 가지고 있는 신경망
- 역전파(back-propagation) 알고리즘 : 다층 퍼셉트론에 사용할 수 있는 학습 알고리즘



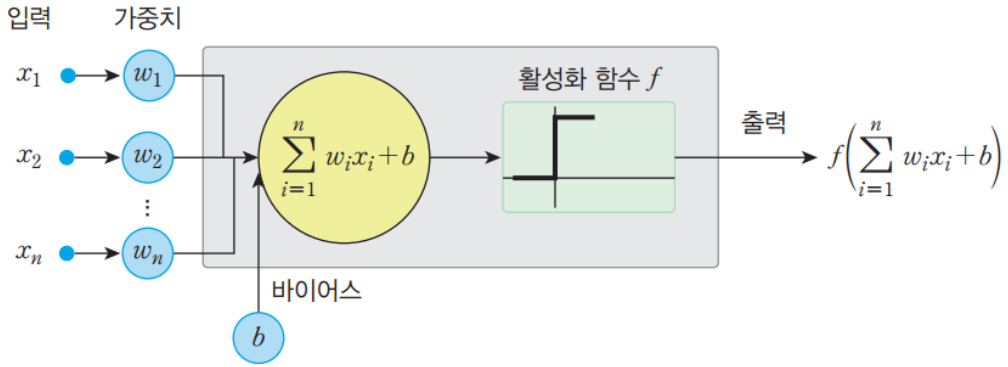
딥러닝

- DNN(deep neural network)에서 사용하는 학습 알고리즘
- MLP에서 은닉층의 개수를 증가시킨 것으로 은닉층을 하나만 사용하는 것이 아니고 여러 개를 사용한다. '딥(deep)'이라는 용어가 은닉층이 깊다는 것을 의미.
- 컴퓨터 시각, 음성 인식, 자연어 처리, 소셜 네트워크 필터링, 기계 번역 등에 적용

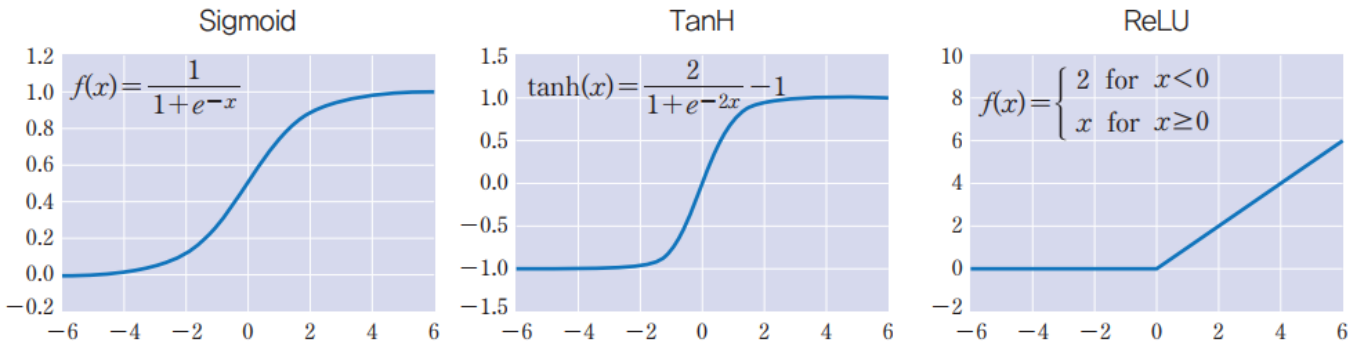


뉴론의 모델

- $\sum_{i=1}^{n} w_i x_i + b$ 가 계산되고 활성화 함수로 입력된다.
- 활성화 함수는 입력값의 가중 합계값을 받아서 출력값을 계산하는 한다.

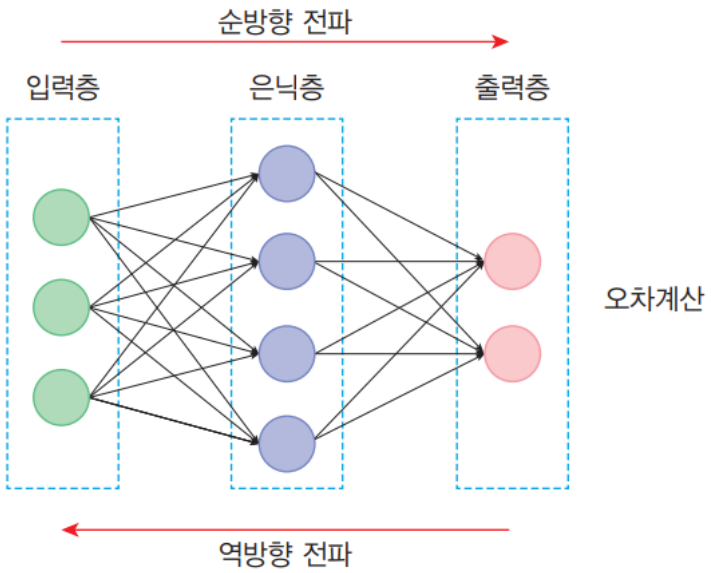


- 활성화 함수의 종류



역전파 학습 알고리즘

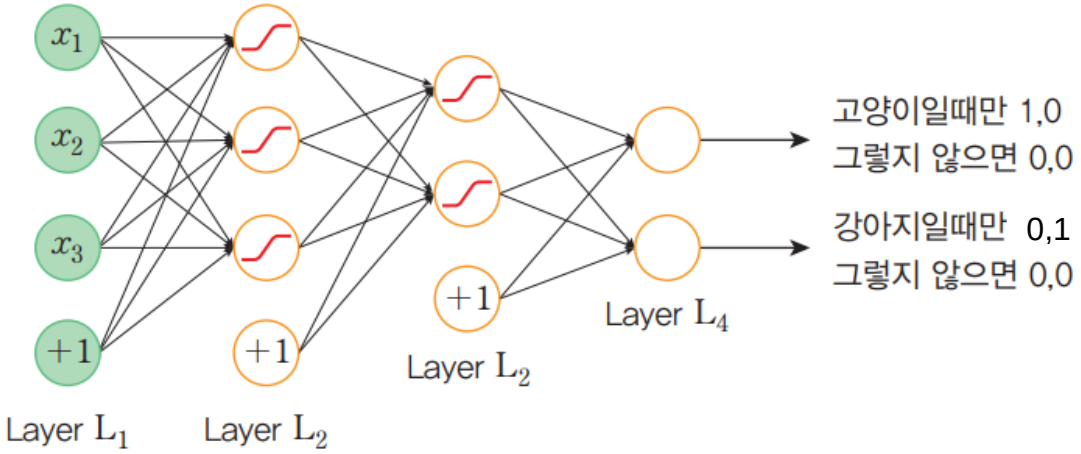
- 역전파 알고리즘은 입력이 주어지면 순방향으로 계산하여 출력을 계산한 후에 실제 출력과 우리가 원하는 출력 간의 오차를 계산한다.
- 이 오차를 역방향으로 전파하면서 오차를 줄이는 방향으로 가중치를 변경한다.
- 이것을 위하여 오차를 계산하는 함수를 정의해야 한다. 이것을 손실 함수(loss function)라고 한다.



손실 함수란 무엇인가

- 예. 강아지와 고양이를 구별하는 신경망에서 강아지 사진을 보여주면 출력 노드는 (0, 1), 고양이 사진을 보여주면 출력 노드는 (1,0)이면 정상. 강아지 사진을 보여주었는데 (1, 0)으로 분류하였다면 잘못 분류한 것. (0,1)로 나와야 하는데 (1,0)으로 나온 것이므로 오차는 (0-1)**2 + (1-0)**2 = 2.0

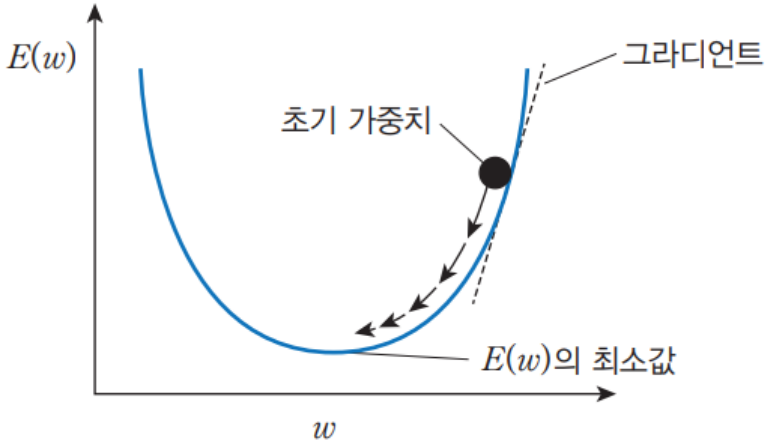
vs.

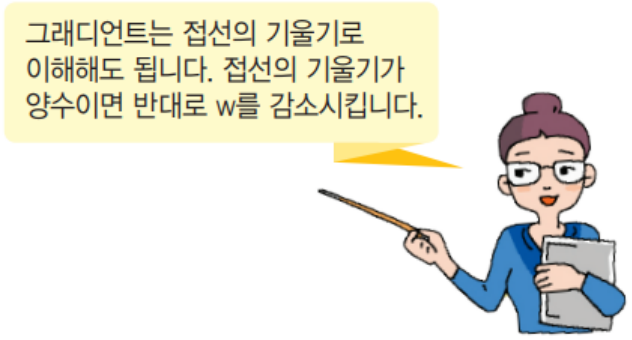


- 손실 함수(loss function) : $E(w)=\frac{1}{2}\sum_i (t_i-o_i)^2$

손실 함수란 무엇인가

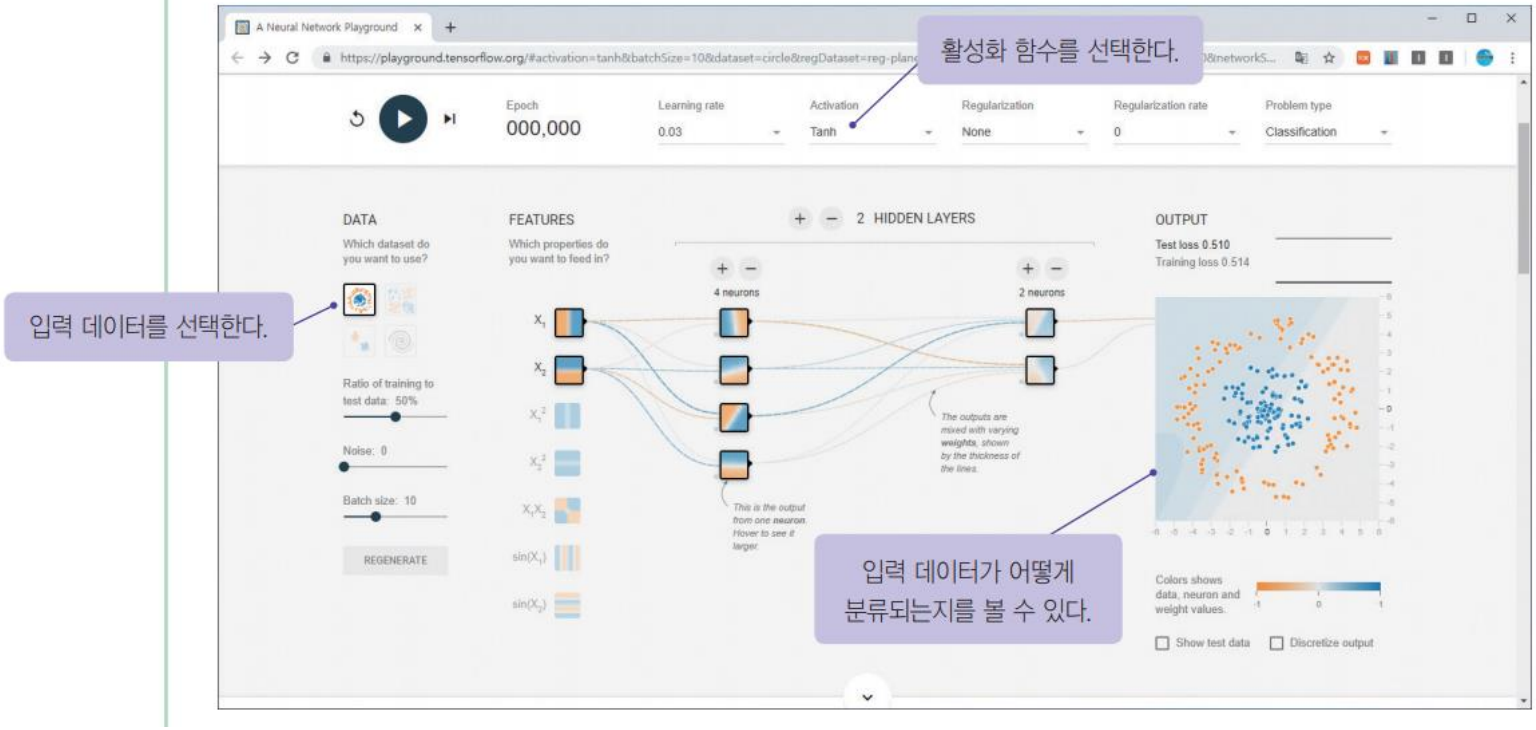
- 손실함수 값을 최소화하는 가중치를 찾는 것이 중요. 만약 손실 함수 값이 0이 되었다면 신경망은 입력을 완벽하게 분류하게 된다.
- 경사 하강법(gradient descent) : 손실 함수의 최소값을 찾기 위해 1차 미분(그라디언트)을 사용하는 반복적인 최적화법. 현재 위치에서 함수의 그라디어트 값을 계산한 후에 그라디언트의 반대 방향으로 움직이는 방법





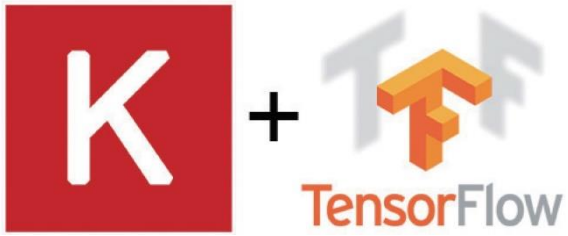
Lab: 활성화 함수 실험

https://playground.tensorflow.org/



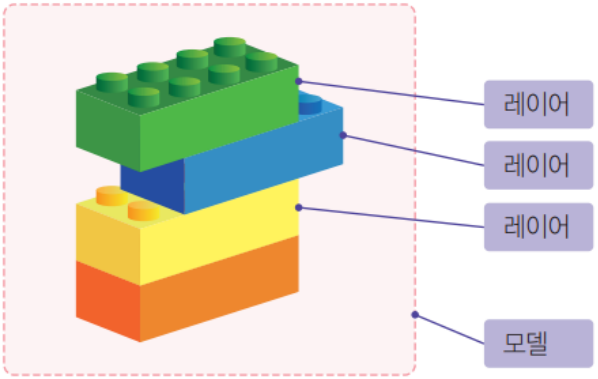
Keras

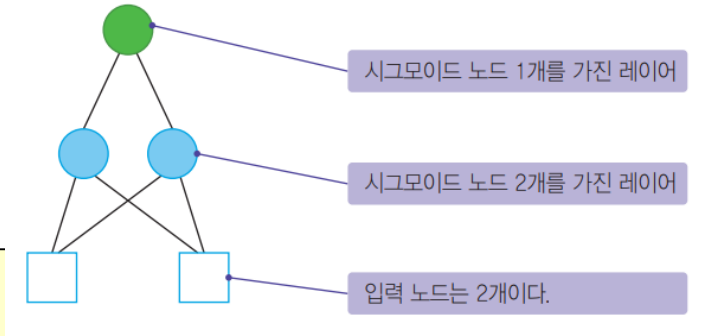
- Keras는 Python으로 작성되었으며 TensorFlow , CNTK 또는 Theano 에서 실행할 수 있는 고수준 딥러닝 API이다.
  - 쉽고 빠른 프로토타이핑이 가능하다.
  - 순방향 신경망, 컨볼루션 신경망과 반복적인 신경망은 물론 물론 여러 가지의 조합도 지원한다.
  - CPU 및 GPU에서 원활하게 실행된다.



Keras

- model : Keras의 핵심 데이터 구조로 레이어를 구성하는 방법. 가장 간단한 모델 유형은 Sequential 선형 스택 모델(순방향으로 전파되는 모델)





- Sequential 모델 생성

```
import tensorflow as tf

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(units=2, input_dim=2, activation='sigmoid'))   # ①
model.add(tf.keras.layers.Dense(units=1,  activation='sigmoid'))                # ②
```

Keras

p707_keras.py

```
# 최적화 방법으로 경사하강법을 사용(학습률 lr = 0.1)
sgd = tf.keras.optimizers.SGD(lr=0.1)

# 학습과정 구성(손실함수로 평균 제곱 오차를 지정)
model.compile(loss='mean_squared_error', optimizer=sgd)

# 입력 데이타셋, 출력 데이타 셋을 주고 XOR 학습을 진행
X = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
y = np.array([[0], [1], [1], [0]])

model.fit(X, y, batch_size=1, epochs=10000)

# 예측값 테스트
print( model.predict(X) )
```

```
[[0.02771977]
 [0.970044 ]
 [0.9701259 ]
 [0.03741568]]
```

	x1	x2	y
샘플 #1	0	0	0
샘플 #2	0	1	1
샘플 #3	1	0	1
샘플 #4	1	1	0

Lab: 논리적인 OR 학습

p707_keras.py

```
import tensorflow as tf
import numpy as np

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units=2, input_dim=2, activation='sigmoid'))   # ①
model.add(tf.keras.layers.Dense(units=1,  activation='sigmoid'))                # ②

sgd = tf.keras.optimizers.SGD(lr=0.1)
model.compile(loss='mean_squared_error', optimizer=sgd)

X = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
y = np.array([[0], [1], [1], [1]])

model.fit(X, y, batch_size=1, epochs=10000)
print( model.predict(X) )
```

```
[[0.02099779]
 [0.9857609 ]
 [0.9858792 ]
 [0.9968916 ]]
```

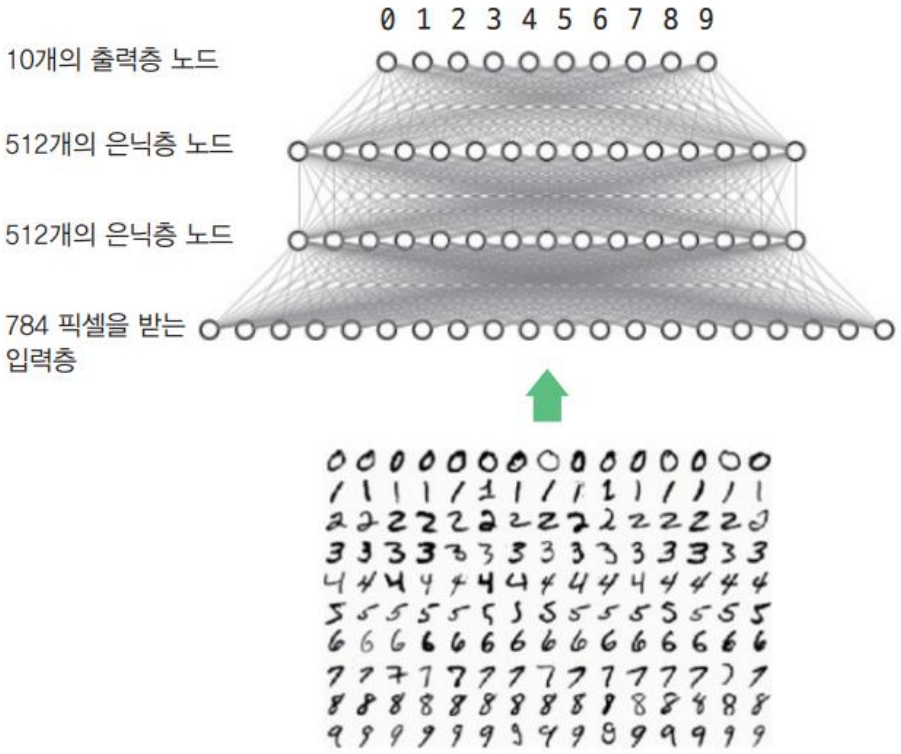
	x1	x2		y
샘플 #1	0	0		0
샘플 #2	0	1		1
샘플 #3	1	0		1
샘플 #4	1	1		1

케라스를 이용한 MNIST 숫자 인식

- MNIST : 필기체 숫자 이미지를 모아둔 데이터 셋(미 표준 연구소)

  http://yann.lecun.com/exdb/mnist/

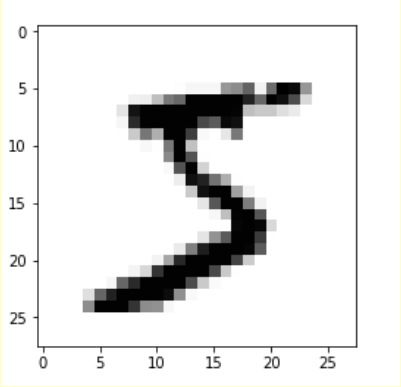
- MLP 구성



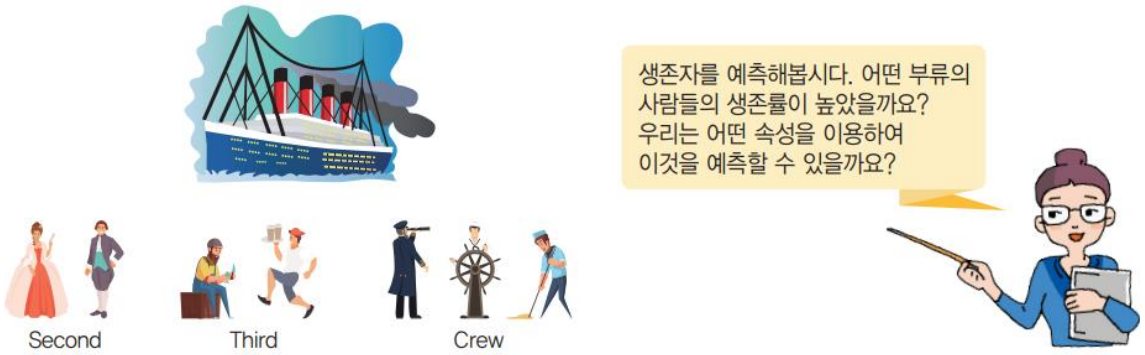
케라스를 이용한 MNIST 숫자 인식

p711_keras_mnist.py

```
import matplotlib.pyplot as plt
import tensorflow as tf

mnist = tf.keras.datasets.mnist

# 훈련 데이터와 테스트 데이터를 가져온다.
(x_train, y_train),(x_test, y_test) = mnist.load_data()

# 넘파이를 사용하여 0에서 255사이의 값을 0.0에서 1.0 사이로 변환.
# 신경망 입력은 0.0에서 1.0으로 정규화되어야 함
x_train, x_test = x_train / 255.0, x_test / 255.0

plt.imshow(x_train[0], cmap="Greys")


# 레이어를 쌓아서 모델 생성
model = tf.keras.models.Sequential()

...
...
```



타이타닉 생존자 예측하기



- 라이브러리 적재

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
```

- Kaggle 사이트에서 학습 데이터, 테스트 데이터 다운로드 (https://www.kaggle.com/c/titanic/data)

```
train = pd.read_csv("train.csv", sep=',')
test = pd.read_csv("test.csv", sep=',')
```

타이타닉 생존자 예측하기

- 학습 데이터 정제

  -원하는 특징만을 남기고 필요없는 컬럼들을 삭제 : drop()

  -원래 데이터 프레임을 변경 : inplace=True

  -1번 컬럼 삭제 : axis=1

  -결손치 있는 데이터 행을 삭제 : dropna()

```
>>> train.drop(['SibSp', 'Parch', 'Ticket', 'Embarked', 'Name',\
        'Cabin', 'PassengerId', 'Fare', 'Age'], inplace=True, axis=1)

>>> train.dropna(inplace=True)

>>> train.head()
   Survived  Pclass     Sex
0         0       3    male
1         1       1  female
2         1       3  female
3         1       1  female
4         0       3    male
```

타이타닉 생존자 예측하기

-성별을 숫자로 변경

```
for ix in train.index:
    if train.loc[ix, 'Sex']=="male":
        train.loc[ix, 'Sex']=1
    else:
        train.loc[ix, 'Sex']=0
```

-생존률을 추출하여 1차원 배열로 변경한 후 학습 데이터에서 삭제

```
target = np.ravel(train.Survived)

train.drop(['Survived'], inplace=True, axis=1)
```

타이타닉 생존자 예측하기

- Keras 모델 만들기 - 순차적 신경망(Sequential)

  -출력값 : '생존', '생존하지 못함' – 이진 분류 문제 – 시그모이드(sigmoid) 출력층 사용

  -2개의 은닉층. Relu 활성화 함수 사용

```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

- 학습을 위해 옵티마이저(경사하강법, adam), 손실 함수(이진 교차 엔트로피), 평가기준(정확도)를 지정

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(train, target, epochs=30, batch_size=1, verbose=1)
```

타이타닉 생존자 예측하기

p719_titanic_deep.py

```
import numpy as np
import pandas as pd
import tensorflow as tf

train = pd.read_csv("train.csv", sep=',')
test = pd.read_csv("test.csv", sep=',')

train.drop(['SibSp', 'Parch', 'Ticket', 'Embarked', 'Name',\
        'Cabin', 'PassengerId', 'Fare', 'Age'], inplace=True, axis=1)

train.dropna(inplace=True)

for ix in train.index:
    if train.loc[ix, 'Sex']=="male":
       train.loc[ix, 'Sex']=1
    else:
       train.loc[ix, 'Sex']=0


…
…
```

이번 장에서 배운 것

- 기계 학습(machine learning)은 인공지능의 한 분야로, 컴퓨터에 학습 기능을 부여하기 위한 연구 분야이다. 인공지능의 한 분야가 기계 학습이라고 할 수 있고 기계학습 중에서 하나의 알고리즘이 딥러닝이라고 할 수 있다.
- 기계 학습은 “교사”의 존재 여부에 따라 크게 지도 학습과 비지도 학습으로 나누어진다. 또 강화학습도 있다. 지도 학습은 "교사"에 의해 주어진 예제(샘플)와 정답(레이블)을 제공받는다. 비지도 학습은 외부에서 정답(레이블)이 주어지지 않고 학습 알고리즘이 스스로 입력에서 어떤 구조를 발견하는 학습이다.
- 지도 학습은 크게 회귀와 분류로 나눌 수 있다. 회귀는 주어진 입력-출력 쌍을 학습한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 예측하는 것이다. 분류(classification): 입력을 두 개 이상의 유형으로 분할하는 것이다. 학습 시에는 교사가 있어서 입력의 올바른 유형을 알려준다.