머신러닝이란?

머신러닝이란 무엇인가?
"명시적인 프로그래밍없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구분야이다." - 아서 사무엘(Arthur Samuel)

머신러닝은 주어진 training data 로 학습을 하지만, 또한 새로운 데이터로도 일반화가 가능해야 한다.
주어진 training data 에서 높은 성능을 보인다고 해도 실제 데이터, 새로운 데이터에서 성능이 떨어진다면 그것은 실패한 알고리즘이다.

머신러닝의 진짜 목적은 new data 에서 잘 동작하는, '일반화'이다.

머신러닝의 종류?

1. 지도학습 (Supervised learning)
지도학습에서는 {입력:정답}, 즉 입력 값과 정답(label)이 같이 주어진다.

분류(Classification)와 회귀(Regression)가 지도학습의 가장 전형적인 예이다.
분류는 범주와 관련되고 회귀는 수치의 예측과 관련 있다.

이 외에도,
서포트벡터머신(SVM),
의사결정나무(Decision tree),
랜덤포레스트(Random forest),
k-Nearest Neighbors,
신경망(Neural networks)가 있다.

2. 비지도학습 (Unsupervised learning)

비지도학습의 training data 에는 정답(label)이 없다. 레이블없이 훈련하고 학습한다.
대표적인 예로,
군집(Clustering)인 k-means, 계층군집분석(HCA),
기대 값 최대화(Expectation Maximization)이 있고,
시각화와 차원축소 그리고 연관규칙학습이 있다.

3. 강화학습 (Reinforcement learning)

환경을 관찰하고 행동을 실행하고 그 결과로 보상 혹은 벌을 받으며 학습하는 방식을 말한다.
학습이 반복되면서 시스템은 가장 큰 보상을 위한 최상의 방법을 스스로 학습한다.

가장 대표적인 예로 우리가 잘 알고 있는, 이세돌을 이긴, 구글 딥마인드의 '알파고'가 강화학습으로 구현되었다.

이 외에도
배치학습,
온라인학습,
준지도학습 등이 있다.

어떻게 테스트하고 검증할 것인가?

머신러닝 모델이 실제로 일반화가 가능한지의 여부를 확인하기 위해서는
new data 들에 실제로 적용해보면 된다.

그러나, 실제 데이터를 활용하는 것은
현실적인 제약이 따르므로 보통은 training data 를 훈련용 데이터와 테스트용 데이터로 나눈다.

훈련용 데이터를 이용해 학습을 하고 테스트용 데이터를 통해 모델을 평가한다.
보통 데이터의 80%를 training data 로, 20%를 test data 로 사용한다.

구글 어시스턴트, 미용실/식당 전화예약

1 분 10 초~ 미용실 전화예약
3 분~ 식당 전화예약

좀 지난 영상이긴 한데, 딥러닝에 있어서 굉장히 의미있는 사건이라 간략히 설명하자면
실제로 미용실에 구글 AI 가 전화를 걸고 예약까지 하는 과정을 보여주는 영상이다.

불과 몇 년 전에 비해 음성인식 기술도 굉장히 발전했으며,
잠깐만 기다려달라는 미용실 직원의 말에 정말 사람처럼 "Mm-hmm"이라고 대답하는 모습을 보며
기술혁신에 의한 우리들의 삶의 변화가 정말 머지않았다는 것을 다시 한 번 느낀다.

딥러닝(deep learning)이란? - 딥러닝과 인공신경망 개념

딥러닝이란 무엇인가?

인공지능, 머신러닝, 딥러닝, 다 비슷한 것 같은 용어들이지만 사실 각 용어 간에는 엄연한 차이가 있다.

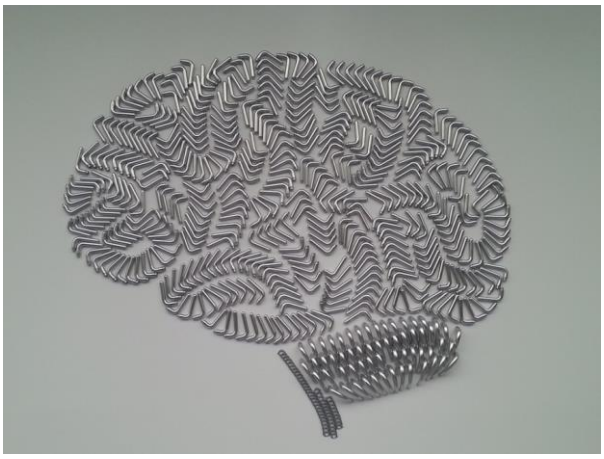
먼저,
인공지능(AI) : 인간의 지능을 본따 인간을 닮은 인공적인 지능의 총체를 의미한다.

우리 말로 기계학습으로 번역되는 **머신러닝(machine learning)** :
기존의 데이터를 이용해 앞으로의 일을 예측하는 방법/기술을 의미한다.

그리고 머신러닝에는 다양한 모델들이 존재하는데 그러한 모델 중 하나가 바로 '**딥러닝**'이다.

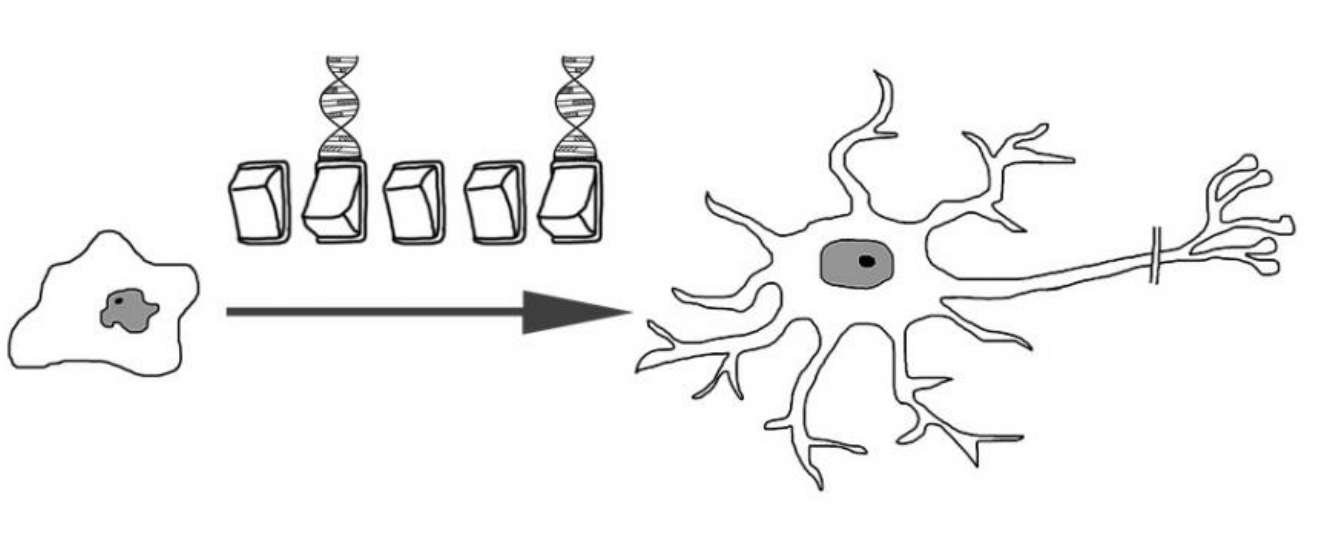
인공지능과 머신러닝 그리고 딥러닝의 관계는 단절된 것이 아니라 포함관계이다.
딥러닝은 머신러닝에 포함되고, 머신러닝은 인공지능이란 거대한 범주 안에 속한다

인공지능 > 머신러닝 > 딥러닝



AI 는 '인간의 뇌 모방'을 base 로 한다.
그렇다면 **인간의 뇌는 어떤 방식으로 작동할까?**

우리는 죽을 때까지 '생각'이란 것으로부터 자유로울 수 없다.
눈을 뜬 순간부터 눈을 감기 직전까지 우리의 뇌는 쉴 틈 없이 생각하고 생각하고, 또 생각한다.
생각을 하지 않으려고 마음먹더라도 그 마음을 먹는 순간마저 '생각'이라는 벗어날 수 없는 족쇄에 얽매인다.
우리의 의지와는 상관없이 결코 해방될 수 없는 이러한 생각은 일련의 과정을 거쳐서 일어난다.



우리의 뇌는 약 천 억 개의 뉴런으로 이루어져 있다.
무수히 많은 뉴런과 뉴런 사이에는 시냅스라는 것이 존재하는데
이 시냅스는 뉴런과 뉴런을 연결하는 중요한 역할을 한다.

신경말단이 자극을 받으면 시냅스에서 화학물질이 분비되는데
이것이 전위변화를 일으키고,
이 전위가 임계치를 넘어서면 다음 뉴런으로 신호를 전달한다.
만약 임계치를 넘지 못한다면 아무 일도 일어나지 않는다.

생각은 이러한 과정을 통해 발생한다.
자극에 의한 화학물질의 분비로 인해서 말이다. 그
리고 이러한 메커니즘을 모방하여 인공적으로 생각하는 무언가를 만들려는 시도가 있었고,
그것이 바로 Artificial Neural Network, **인공신경망**이다.

인공신경망은 뇌의 작동과정과 유사하게 동작한다.

기본단위인 퍼셉트론(Perceptron)은 입력 값을 다음으로 넘기는 역할을 한다.

퍼셉트론의 개념은 다음과 같다.

y = wx + b
(w 는 가중치, x 는 입력값, b 는 바이어스)

입력 값 x 가 들어오면 입력값 x 에 가중치 w 를 곱한다.
입력 값인 x1, x2, x3, x4, x5... 를 가중치인 w1, w2, w3, w4, w5... 와 곱하고 모두 더한다.
거기에 바이어스 b 를 더하면, 그 결과가 **가중합(Weighted sum)**이다.

시그모이드 함수나 RELU 함수, 소프트플러스(Softplus)와 같은 활성화 함수(Activation function)를 통해 가중합을 놓고 0 or 1 을 판단하여 출력한다.

scikit-learn iris 데이터 셋 - knn 모델

Scikit-learn 의 datasets 에 있는 iris 데이터셋을 통해
KNN(k-nearest neighbors)모델을 구현해본다. (정말 정말 단순하고 간단하게)

iris data set 이란, 붓꽃의 품종과 꽃잎, 꽃받침, 폭과 길이를 담은 데이터 세트이다

필요한 패키지와 모듈을 불러온다

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
```

load_iris() 함수를 통해 데이터를 iris 변수에 저장하고 출력해봤더니
sklearn 의 Bunch 클래스 형식이다

```
iris = load_iris()
```

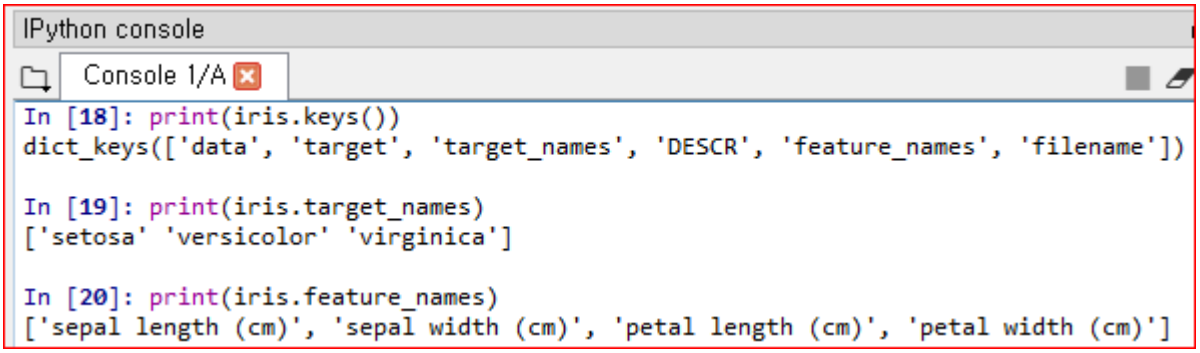
Variable explorer

Name	Type	Size	Value
iris	utils.Bunch	6	Bunch object of sklearn.utils module

```
In [17]: 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],
```

중략...

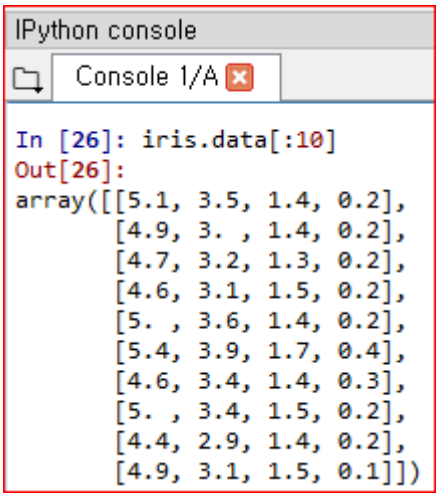
```
Transactions\n     on Information Theory, May 1972, 431-433.\n   - See also: 1988
MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual clustering
system finds 3 classes in the data.\n   - Many, many more ...', 'feature_names':
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'],
'filename': 'C:\\Anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\iris.csv'}
```



iris 에는
setosa (부채붓꽃),
versicolor (흰여로),
virginica(질경이)
라는 붓꽃 품종(iris.target_names)이 있고

sepal length (꽃받침 길이),
sepal width (꽃받침 너비),
petal length (꽃잎 길이),
petal width (꽃잎 너비)
라는 특성(features: iris.feature_names)을 갖고 있다.

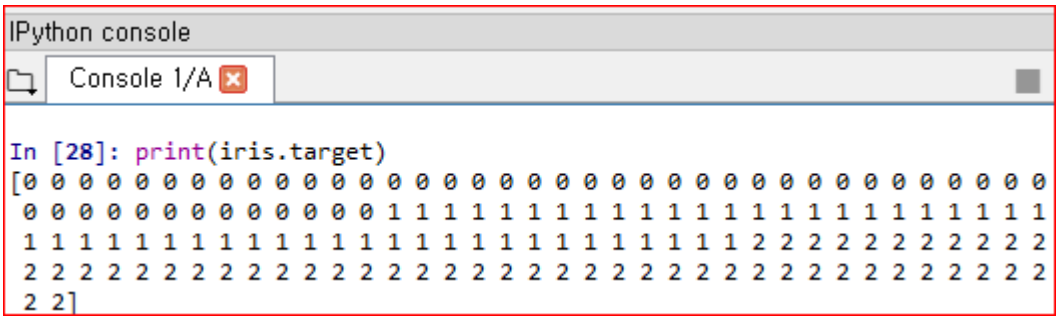
feature 들을 통해
이 데이터 셋에 없는 새로운 데이터가 들어왔을 때,
그 품종을 예측하는 머신러닝 모델을 만드는 것이 목적이다.



데이터의 1 행을 보면
5.1 , 3.5, 1.4, 0.2 라고 되어있는데

이는 각각
sepal length, sepal width, petal length, petal width 를 나타낸다.

우리가 예측하고자 하는 붓꽃의 품종은 다음과 같다
0 은 setosa, 1 은 versicolor, 2 는 virginica 이다.



머신러닝 모델을 만들 때는 데이터를 인풋으로 넣고 학습시키는 것도 중요하지만,
그렇게 학습시킨 모델의 성능을 평가하는 것도 중요하다.

모델의 성능을 평가하지 않으면,
모델이 과적합(Overfitting)되었는지 일반화가 되지 않았는지를 판단할 수 없다.

가령, 데이터를 전부 학습하는 데에만 사용한다면 모델의 정확도는 높을 수 있으나
새로운 데이터가 주어졌을 때 예측도가 현저히 떨어지는, 과적합이 발생할 수 있다.
(즉, 일반화에 실패할 수 있다.)

머신러닝의 목적은 성능을 높이고 일반화하는 데에 있다.

이를 위해 주어진 데이터를
  학습데이터(training data)와
  테스트데이터(test data, hold-out set)
으로 나누어야 한다.

scikit-learn 에서는
데이터 셋을 훈련셋과 테스트셋으로 분할하는 데에 용이한 train_test_split 이란 함수를 제공한다.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(iris.data,
                                                    iris.target,
                                                    test_size=0.3,
                                                    random_state=2019)
```

X_train 에는 전체 데이터 셋의 70%를, X_test 에는 나머지 30%의 데이터를 가진다.

원래 머신러닝 모델에 데이터를 학습시킬 때는 feature engineering 을 거쳐야 하는데
iris 셋에선 큰 필요가 없을 것 같아 그냥 있는 그대로의 데이터를 넣고 학습을 시킨다

Variable explorer

Name	Type	Size	Value
X_test	float64	(45, 4)	[[4.4 3.2 1.3 0.2] [5.4 3.4 1.7 0.2]]
X_train	float64	(105, 4)	[[6.1 2.8 4.7 1.2] [4.8 3.1 1.6 0.2]]
Y_test	int32	(45,)	[0 0 2 ... 0 2 1]
Y_train	int32	(105,)	[1 0 1 ... 0 2 1]

각각의 값 확인

IPython console

Console 1/A

```
In [33]: print(X_train)
[[6.1 2.8 4.7 1.2]
 [4.8 3.1 1.6 0.2]
 [5.1 2.5 3.  1.1]
 [5.5 3.5 1.3 0.2]
```

IPython console

Console 1/A

```
In [34]: print(X_test)
[[4.4 3.2 1.3 0.2]
 [5.4 3.4 1.7 0.2]
 [6.7 3.3 5.7 2.1]
 [6.3 2.3 4.4 1.3]
```

IPython console

Console 1/A

```
In [35]: print(Y_train)
[1 0 1 0 0 2 0 2 1 2 0 0 2 0 1 2 2 0 1 0 2 2 0 2 1 1 0 2 1 1 1 2 1 2 2 1 2
 1 0 0 2 1 2 2 0 0 2 1 1 1 1 2 2 2 2 1 1 2 1 0 2 0 0 0 2 0 1 1 0 1 1 1 1 1
 1 2 1 1 1 0 1 1 2 1 0 1 1 1 2 0 0 2 2 2 0 1 0 2 2 0 0 0 0 2 1]
```

IPython console

Console 1/A

```
In [36]: print(Y_test)
[0 0 2 1 2 0 2 0 1 2 2 2 2 1 0 1 2 1 0 2 0 2 0 1 0 0 1 2 0 0 2 0 0 2 2 0 0
 2 0 1 0 1 0 2 1]
```

KNN 모델의 fit 메서드를 통해 입력(X_train)과 정답(Y_train)을 넣고 학습시킨다.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
```

학습을 시켰으므로 모델의 성능을 평가해본다.

IPython console

Console 1/A

```
In [42]: print('accuracy : {:.2f}'.format(knn.score(X_test, Y_test)))
accuracy : 1.00
```

전체 데이터셋에서 30% 데이터를 분할하여 test 데이터로 만들었었다.
그 test 데이터에는 {입력:정답}을 모두 갖춘 데이터 셋이므로 학습시킨 모델을 평가하기에 적절하다.

knn.score(X_test,Yy_test)는 X_test 데이터를 아까 학습시킨 knn 모델에 넣고
Y_test 와 비교하여 정확도를 출력한다. 100%의 정확도를 얻었으며,
이는 테스트 세트에 있는 샘플 데이터들의 100%를 정확히 맞췄다는 뜻이다.[