라즈베리파이와 텐서플로로 배우는 딥러닝

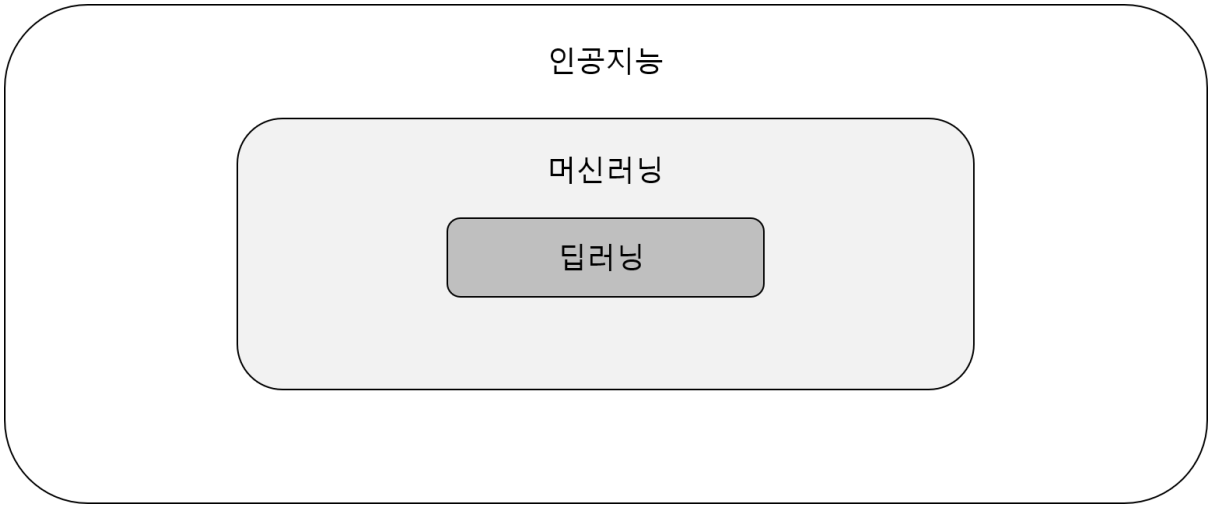
메카솔루션

www.mechasolution.com

1. 인공지능 / 머신러닝 / 딥러닝

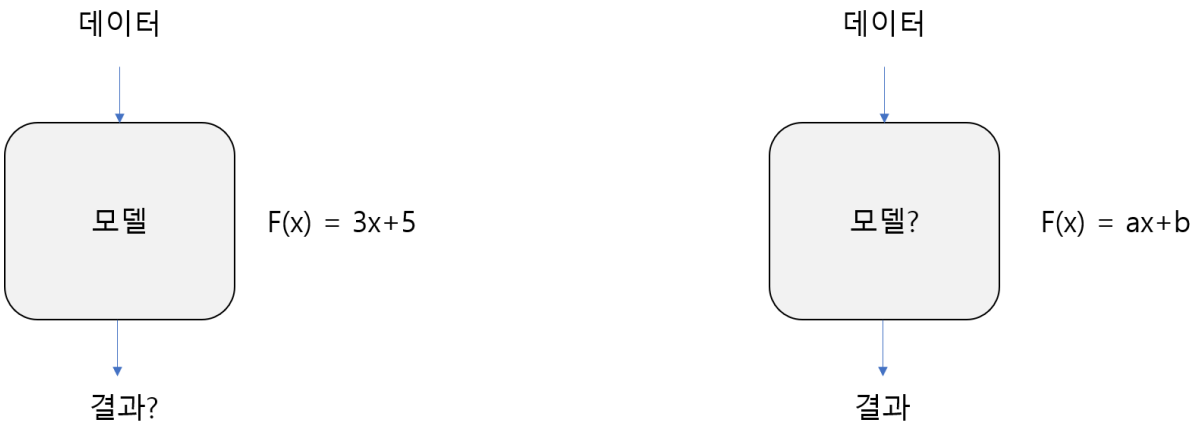
인공지능, 머신러닝, 그리고 딥러닝은 일반인들이 생각했을 때, 혼란스러운 개념일 수 있습니다. 뉴스와 언론 등에서 인공지능을 이야기할 때도 있고, 딥러닝을 이야기할 때도 있기 때문이죠. 예를 들어, 알파고가 인공지능인가 딥러닝인가에 대해서 설명할 때도 비전공자들에게는 헷갈릴 수 있습니다 (심지어 전공자들에게까지도).

대학 및 대학원에서 배우는 인공지능 과목, 머신러닝 과목, 그리고 딥러닝 과목의 커리큘럼을 보면, 어쩌면 이해하기 수월할 지 모르겠습니다. 일반적으로 인공지능은 4학년 혹은 대학원 1학년 때의 과목으로 개설되면서 Introduction to Artificial Intelligence로 되어 있고, 머신러닝과 딥러닝은 인공지능을 선행 이수 과목 (Prerequisite)으로 정하는 경우가 많습니다. 즉, 인공지능을 배운 다음, 머신러닝을 배우고, 그 다음에 딥러닝을 배우는 것이 뭔가 더 체계적이라고 할까요? 때문에, 많은 사람들이 인공지능, 머신러닝, 딥러닝의 관계를 다음과 같이 표현하곤 합니다.



함수란 입력과 출력 사이에서의 모델링을 직접 하고, 해당 입력을 넣었을 때 출력을 얻는 것과 같습니다. 예를 들어, F(x) = 3x + 5를 만들고, x에 10을 넣으면 결과로 35를 얻는 과정입니다.

머신러닝이란 사람이 직접 모델링하기 어려운 부분들에 대해서 전체적인 틀을 잡아주고, 많은 수의 데이터를 통해 컴퓨터가 모델을 만들어가는 과정으로 기계(머신)가 학습(러닝)하는 것을 의미합니다. 단순히 2개의 점을 가지고 직선의 방정식을 구하는 것을 넘어 사진, 음성 등을 통해 분류를 하려면 많은 수의 데이터가 필요하고, 이를 처리할 수 있는 좋은 성능의 기계 및 컴퓨터도 필요합니다.



즉, 머신러닝이란 우리가 직접 많은 데이터를 보고, 분석해서 모델을 구체화하는 것이 아니라, Y = ax + b과 같이 일종의 틀만 만들고, a와 b에 해당하는 최적의 값을 찾는 것이라고 볼 수 있습니다.

2. 학습과 실제, 그리고 과적합

머신러닝은 많은 데이터를 통해 학습을 하고, 전혀 다른 데이터를 실제로 모델에 적용하면서 분류를 합니다. 학습을 했을 때 정확도가 95%인데, 실제로 정확도는 어느 정도일까요? 얼마면 될까요?

일반적으로 학습을 했을 때보다 실제로 정확도가 높지는 않을 것입니다. 하지만, 학습을 너무 잘해도 문제입니다. 이는 오버피팅 (Overfitting)이라는 용어로 설명이 가능한데, 너무 과하게 분류를 했다고 이해하면 될 것 같습니다. 오버피팅을 설명하기 위해, 언더피팅 (Underfitting)과 바람직한 피팅, 그리고 오버피팅을 그림으로 설명하면 다음과 같습니다.

3. 지도학습, 비지도학습, 그리고 강화학습

머신러닝은 학습하는 방법에 따라서 크게 1) 지도학습 (Supervised learning), 2) 비지도학습 (Unsupervised learning), 3) 강화학습 (Reinforcement learning)으로 나눌 수 있습니다.

각각의 특징으로는 다음과 같습니다.

1) 지도학습: 정답이 있는 문제를 풀게 됩니다. 예를 들어, 여러 장의 사진을 통해 여자와 남자를 구별해야한다면, 각각의 사진마다 여자인지 남자인지 알고 학습을 합니다. 많은 수의 데이터(사진과 라벨)를 통해 학습을 하다 보면, 새로운 데이터가 들어왔을 때 더 정답에 가까워집니다. 이러한 지도학습으로는 선택을 하는 분류, 예측하는 회귀, 그리고 추천 알고리즘이 해당됩니다.

2) 비지도학습: 정답이 없기 때문에 데이터의 특성들을 파악하거나 유형을 나누는데 도움이 됩니다. 예를 들어, 곤충 사진과 물고기 사진들을 섞어 놓고 구분을 짓는 것은 비지도학습일 수 있습니다. 그리고, 이렇게 특징을 파악한 다음에 라벨을 붙인 다음에 지도학습을 통한 분류를 하기도 합니다. 비지도학습으로는 데이터의 분포를 예측하거나 차원을 축소하는 것을 포함하기도 합니다.

3) 강화학습: 강화학습은 로봇제어에서도 많이 사용되고, 게임에서도 종종 사용하는 학습 방법으로 최적화된 값을 찾기 위해서 보상이라는 당근과 채찍을 사용합니다. 미로를 찾는데, 잘 찾으면 +1을 반대로 가면 -1을 주는 방법이랄까요?

4. 분류 (Classification), 군집화 (Clustering), 회기 (Regression)

머신러닝을 하게 되면 가장 많이 사용하는 것이 분류입니다. 그리고, 군집화는 비지도학습의 주요 특징이기도 하며, 그룹화하는 것을 의미합니다. 끝으로 회기는 데이터를 기반으로 예측하는 것을 말합니다. 분류와 군집화, 그리고 회기는 각각의 예를 통해서 이해할 수 있겠습니다.

1) 분류

   A. 수상한 사람과 수상하지 않은 사람 구별하기

   B. 자동차와 사람 구별하기

   C. 남자와 여자 구별하기

2) 군집화

   A. 회사의 매출, 트래픽, 성장성 등을 토대로 회사의 군을 나눠보기

   B. 구매 데이터를 통해서 고객의 성향 및 구매 패턴 파악하기

3) 회기

A. 월별 상품 구매 수량과 구매 횟수를 통해서, 다음 주 매입 수량 예측하기

B. 상품 판매량에 영향이 있는 요소들을 통해 가중치가 높은 요소 찾기

5. 확률통계와 딥러닝

딥러닝이라는 용어가 나오기 전에는 인공지능이 없었을까요? 그렇지 않습니다. 사실, 인공지능 과목에서 배우는 상당수는 확률통계에 기초하는 학문입니다. 때문에, 인공지능, 머신러닝, 그리고 딥러닝의 키워드를 선택하라고 하면, 다음과 같이 말할 수 있습니다.

- 인공지능: 확률통계
- 머신러닝: 학습
- 딥러닝: 신경망

6. 머신러닝 소프트웨어 소개

머신러닝하면, 생각하는 라이브러리가 텐서플로로 알고 계시는 분들이 많지만, 텐서플로가 개발되기 전에도 머신러닝은 있었고, 많은 사람들이 연구를 해 왔습니다. 머신러닝 라이브러리와 딥러닝 라이브러리, 그리고 최근에 딥러닝의 수치 연산을 위한 라이브러리를 살펴보면 다음과 같습니다.

1) 머신러닝 라이브러리

A. 넘파이 (Numpy): 파이썬에서 고속으로 행렬 및 수치 연산을 지원하는 라이브러리

B. 싸이파이 (scipy): 넘파이와 더불어 다양한 수학 함수를 지원하는 라이브러리

C. 사이킷런 (scikit-learn): 파이썬용 머신러닝 라이브러리

D. Matplotlib: 파이썬에서 그래프를 그리기 위한 라이브러리

E. OpenCV: 파이썬에서 이미지 데이터를 처리하는데 필요한 라이브러리

F. Pandas: 데이터 관련 다양하고 편리한 함수들을 제공하는 라이브러리

2) 딥러닝 라이브러리

A. 텐서플로: 구글에서 개발한 딥러닝 라이브러리로 최적화 함수 및 최신의 딥러닝 모델을 제공합니다.

B. Caffe: 버클리 주립대에서 개발되었으며, 이미지를 사용하는 딥러닝에 강합니다.

C. Theano: 몬트리올 대학에서 파이썬으로 개발되었습니다

3) 그래픽 카드 관련 라이브러리

A. CUDA: 그래픽카드로 유명한 엔비디아에서 만들어졌습니다

B. OpenCL: 애플에서 개발했으며, 애플, AMD, 인텔에서 사용할 수 있습니다.

7. 라즈베리파이와 OpenCV, TensorFlow

많은 사람들이 딥러닝을 할 때는 좋은 PC와 그래픽 카드를 사용합니다. 물론, 성능이 좋은 시스템이라면, 보다 짧은 시간에 학습이 가능합니다. 하지만, 이동성 및 휴대성을 생각하면 라즈베리파이와 같은 임베디드 시스템 혹은 싱글보드컴퓨터(Single Board Computer)가 적합할 수 있습니다. 라즈베리파이에서 텐서플로를 사용하여 딥러닝을 하는 것이 가능하며, 어쩌면 교육을 목적으로 한다면 최적의 시스템일 수 있습니다.

1) 라즈베리파이에 운영체제 설치하기

- 16GB이상의 마이크로SD카드를 준비한 후, 포맷을 합니다.
- 라즈비안 운영체제를 다운로드 받고, 압축을 풉니다
- ETCHER라는 소프트웨어를 이용하여 .img 확장자인 운영체제 디스크 이미지를 마이크로SD카드에 굽습니다.

2) 가상환경에 OpenCV와 Tensorflow 설치하기

라즈베리파이를 부팅한 후에, 안정적인 네트워크를 연결한 후 (랜선을 이용한 이더넷 및 안정성 있는 와이파이), 터미널에서 다음의 라인을 실행하며 패키지를 설치합니다.

```
$ sudo apt-get install ibus ibus-hangul fonts-unfonts-core
$ sudo reboot
$ sudo apt-get install update
$ sudo apt-get install upgrade
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py

$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm –rf ~/.cache/pip
$ nano ~/.profile
```

위에서 nano 에디터를 이용해서 profile을 열고, 맨 하단에 다음의 세 줄을 적고 저장합니다.

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

가상환경으로 들어가서 활성화하기 위해서 다음을 입력합니다.

```
$ source ~/.profile
$ mkvirtualenv cv –p python3
$ workon cv
```

텐서플로와 OpenCV를 설치하기 위해 다음의 패키지를 설치합니다. 그리고, pip를 이용하여 opencv와 tensorflow를 설치합니다.

```
(cv) $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
(cv) $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
(cv) $ sudo apt-get install libxvidcore-dev libx264-dev
(cv) $ sudo apt-get install qt4-dev-tools
(cv) $ sudo apt-get install libatlas-base-dev

(cv) $ pip3 install opencv-python
(cv) $ pip3 install --no-cache-dir tensorflow
```

3) 머신러닝 라이브러리 설치하기

```
(cv) $ pip3 install numpy
(cv) $ pip3 install scipy
(cv) $ pip3 install scikit-learn
(cv) $ pip3 install matplotlib
(cv) $ pip3 install pandas
(cv) $ pip3 install pgi
(cv) $ pip3 install cairocffi
(cv) $ pip3 install seaborn
(cv) $ pip3 install Pillow
(cv) $ pip3 install keras
(cv) $ pip3 install cython
(cv) $ sudo apt-get install libhdf5-dev
(cv) $ pip3 install h5py
(cv) $ pip3 install pygame
(cv) $ pip3 install pyaudio
```

8. OpenCV 실습

OpenCV는 1999년 인텔(Intel)의 Gary Bradsky가 개발을 시작하여 2000년에 처음으로 공개되었습니다. 2005년에 다르파 그랜드 챌린지(DARPA Grand Challenge)에서 OpenCV를 사용한 스탠리라는 차량이 우승을 하기도 하였으며, 컴퓨터 비전 라이브러리로 머신러닝과 함께 꾸준히 발전되고 있습니다. 특히, OpenCV는 C++, Python, C# 등 다양한 언어를 지원하며, 윈도우, 리눅스, 맥, 안드로이드 등 다양한 운영체제에서도 사용할 수 있습니다.

1) 도형 그리기 (geometry.py)

Cv2.line(), cv2.circle(), cv2.rectangle(), cv2.putText()를 사용할 수 있습니다.

```
import numpy as np
import cv2

img = np.zeros((512, 512, 3), np.uint8)
img = cv2.line(img, (0, 0), (511, 511), (255, 0, 0), 5)
img = cv2.rectangle(img, (384, 0), (510, 128), (0,255,0), 3)
img = cv2.circle(img, (447,63), 63, (0,0,255), -1)
cv2.putText(img, 'OpenCV', (10,500), cv2.FONT_HERSHEY_SIMPLEX, 4, (255,255,255),2)
```

```
cv2.imshow('image',img)



cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) 이미지를 읽기 (imageread.py)

```
import cv2

img = cv2.imread('lena.jpg')
cv2.imshow('image',img)
k = cv2.waitKey(0) & 0xFF
if k == 27: # esc key
    cv2.destroyAllWindow()
elif k == ord('s'): # 's' key
    cv2.imwrite('lenagray.png',img)
    cv2.destroyAllWindow()
```

cv2.waitKey(0)

: keyboard입력을 대기하는 함수로 0이면 key입력까지 무한대기이며 특정 시간동안 대기하려면 milisecond값을 넣어주면 됩니다.

k = cv2.waitKey(0) & 0xFF

: 64비트 운영체제에서는 위와 같이 BIT 연산을 해야 합니다.

Cv2.destroyAllWindows()

: 화면에 나타난 윈도우를 종료합니다.

3) 색상 변환하기 (convertcolor.py)

```
import cv2
```

```
img = cv2.imread('lena.jpg')
cv2.imshow('image',img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('gray',gray)

k = cv2.waitKey(0) & 0xFF
if k == 27: # esc key
    cv2.destroyAllWindow()
elif k == ord('s'): # 's' key
    cv2.imwrite('lenagray.png',img)
    cv2.destroyAllWindow()
```

4) 엣지 검출하기 (edgedetect.py)

```
import cv2

img = cv2.imread('lena.jpg')
cv2.imshow('image', img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('gray', gray)
edge = cv2.Canny(gray,100,200)
cv2.imshow('edge', edge)

k = cv2.waitKey(0) & 0xFF
if k == 27: # esc key
    cv2.destroyAllWindow()
elif k == ord('s'): # 's' key
    cv2.imwrite('lenagray.png',img)
    cv2.destroyAllWindow()
```

5) 카메라로부터 영상 재생하기 (webcam.py)

```
import cv2

cap = cv2.VideoCapture(-1)
ret, frame = cap.read()

while(True):
    ret, frame = cap.read()
```

```
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

6) 회색 변환 이미지 재생하기 (graywebcam.py)

```
import cv2

cap = cv2.VideoCapture(-1)
ret, frame = cap.read()

while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

7) 동영상 파일 재생하기 (playvideo.py)

```
import cv2
cap = cv2.VideoCapture('tst.avi')

while(cap.isOpened()):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame',gray)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

8) 영상에서 파란색 찾기 (detectblue.py)

```
import cv2
import numpy as np

cap = cv2.VideoCapture(-1)
cap.set(3,320)
cap.set(4,240)

while(1):
    # camera에서 frame capture.
    ret, frame = cap.read()

    if ret:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        lower_blue = np.array([110, 50, 50])
        upper_blue = np.array([130, 255, 255])

        mask = cv2.inRange(hsv, lower_blue, upper_blue)
        res = cv2.bitwise_and(frame, frame, mask = mask)

        cv2.imshow('frame', frame)
        cv2.imshow('mask', mask)
        cv2.imshow('res', res)

    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

9) 사진에서 얼굴 찾기 (facedetect.py)

wget https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_frontalface_alt.xml

```
import cv2
```

```
face_cascade = cv2.CascadeClassifier('/home/pi/code/haarcascade_frontalface_alt.xml')
img = cv2.imread('/home/pi/code/G20.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rects = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in rects:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 3)
cv2.imshow('Input image', img)
cv2.waitKey()
```

10) 사진에서 상체 찾기 (upperbodydetect.py)

wget https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_upperbody.xml

```
import cv2
body_cascade = cv2.CascadeClassifier('/home/pi/code/haarcascade_upperbody.xml')
img = cv2.imread('/home/pi/code/G20.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rects = body_cascade.detectMultiScale(gray, 1.1, 5)
for (x, y, w, h) in rects:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 3)
cv2.imshow('Input image', img)
cv2.waitKey()
```

11) 영상에서 얼굴 찾기 (facedetectwebcam.py)

```
import cv2

face_cascade = cv2.CascadeClassifier('/home/pi/code/haarcascade_frontalface_alt.xml')
cap = cv2.VideoCapture(-1)
while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in rects:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 3)
    cv2.imshow('Face', frame)
```

```
    k = cv2.waitKey(1)
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

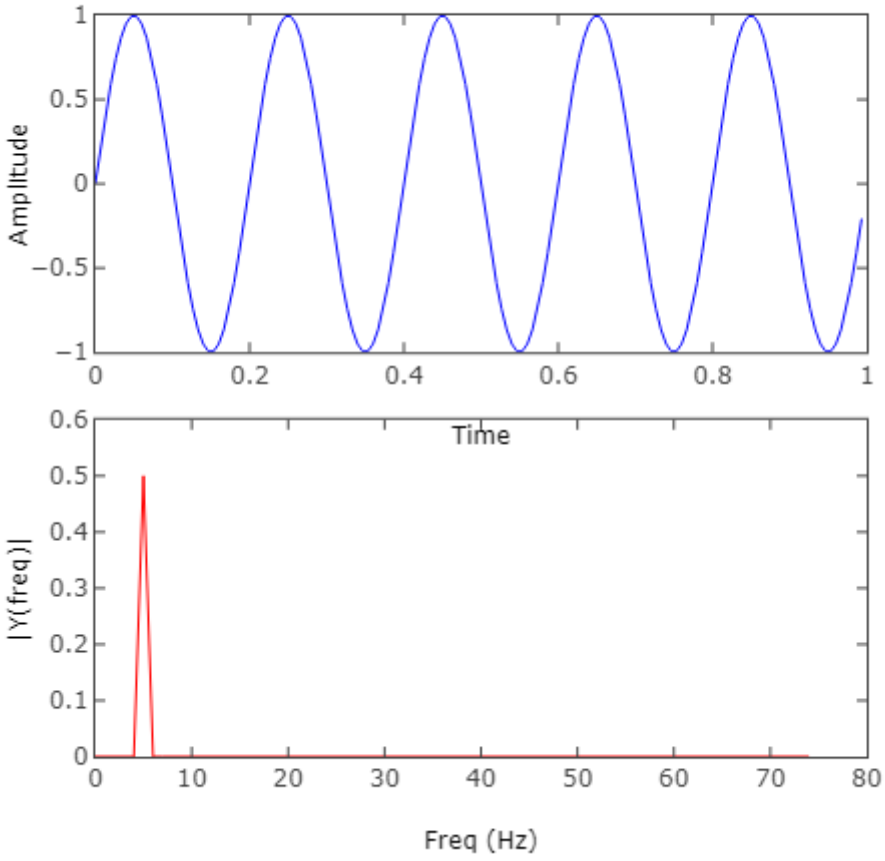
9. 영상과 음성처리를 위한 기초: FFT

FFT는 Fast Fourier Transform의 약자로 다양한 분야에서 많이 사용되고 있으나 엔지니어들은 한번씩 들어보았지만 비전공자들은 한번도 들어보지 못한 경우가 많습니다. 수식은 빼고, 프로그램을 통해서 시간 영역에서의 데이터가 주파수 영역의 신호로 변경되는 것을 확인할 수 있습니다.

1) FFT 예제

```
import matplotlib.pyplot as plt
import numpy as np

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # period
t = np.arange(0,1,Ts) # time vector

ff = 5;   # frequency of the signal
y = np.sin(2*np.pi*ff*t)

n = len(y) # length of the signal
k = np.arange(n)
T = n/Fs
frq = k/T # two sides frequency range
frq = frq[range(int(n/2))] # one side frequency range

Y = np.fft.fft(y)/n # fft computing and normalization
Y = Y[range(int(n/2))]

fig, ax = plt.subplots(2, 1)
ax[0].plot(t,y)
```

```
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].plot(frq,abs(Y),'r') # plotting the spectrum
ax[1].set_xlabel('Freq (Hz)')
ax[1].set_ylabel('|Y(freq)|')

plt.show()
```



2) 음식 데이터 캡쳐하기

음성 파일은 다음과 같이 녹음할 수 있습니다. 이 때, 라즈베리파이에 마이크와 스피커가 연결되어 있어야 하며, 그렇지 않을 경우에는 Respeaker라는 모듈을 다음과 같이 사용해볼 수 있습니다.

제품 링크: http://mechasolution.com/shop/goods/goods_view.php?goodsno=576738&category=

Respeak 설치 방법:

```
(cv) $ git clone https://github.com/respeaker/seeed-voicecard
(cv) $ cd seeed-voicecard
(cv) $ sudo ./install.sh
```

out.wav라는 파일 녹음하고 플레이해보기

```
(cv) $ arecord --format=S16_LE --duration=5 --rate=16000 --file-type=wav out.wav

(cv) $ aplay --format=S16_LE --rate=16000 out.wav
```

소리를 재생하고 그래프로 출력하는 코드

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
import pygame
import time

pygame.init()
```

```
pygame.mixer.init()
sounda = pygame.mixer.Sound('/home/pi/out.wav')
sounda.play()
time.sleep(1)

fs, data = wavfile.read('/home/pi/out.wav')
time = np.linspace(0, 100, num=(len(data)))
plt.plot(time, data)
plt.show()
```

스터디 주제:

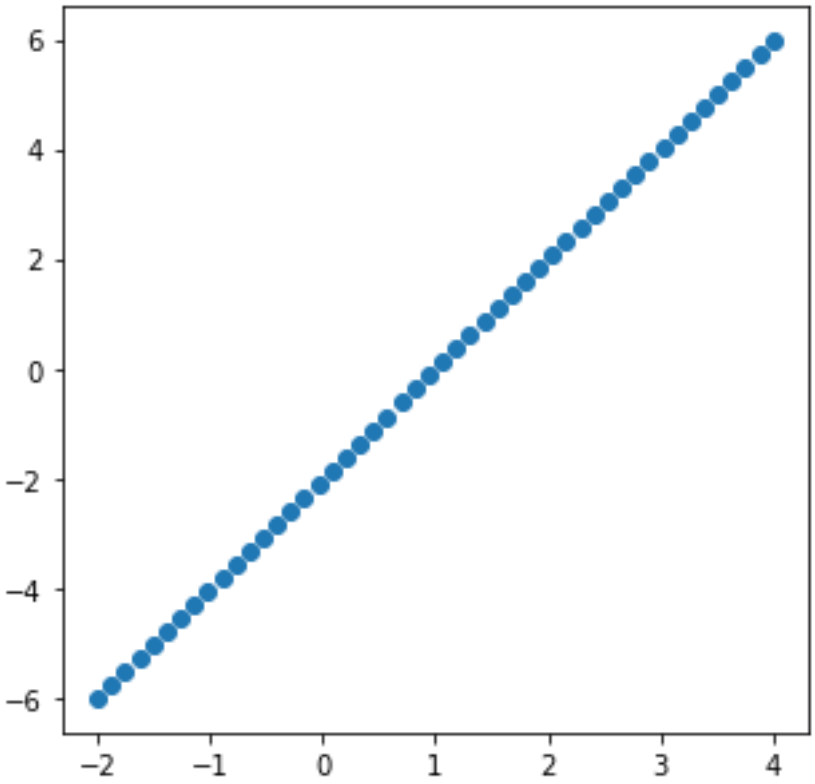
음성을 녹음하여 FFT로 변환하고 노이즈에 해당하는 부분을 제거하기

많은 수의 음성을 샘플로 저장하고, 러닝 데이터로 활용하기

10. 머신러닝과 딥러닝 실습

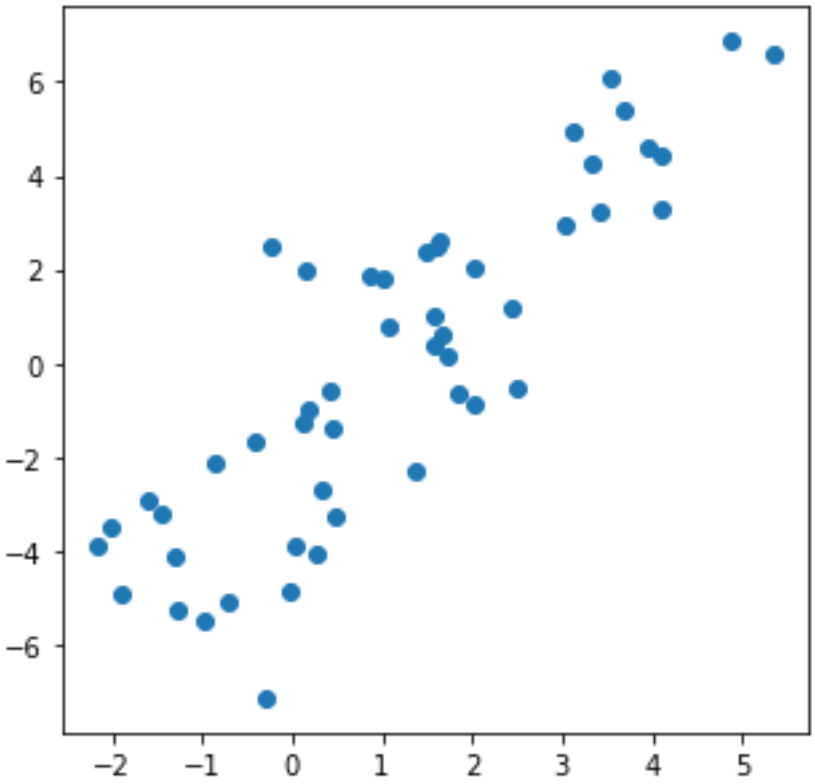
1) 그래프 출력해보기 (graph.py)

```
import numpy as np
import matplotlib.pyplot as plt

num_examples = 50
X = np.array([np.linspace(-2, 4, num_examples), np.linspace(-6, 6, num_examples)])
plt.figure(figsize=(5,5))
plt.scatter(X[0], X[1])
plt.show()
```



2) 그래프에 노이즈 (perturbation) 추가하기 (graphperturb.py)

```
import numpy as np
import matplotlib.pyplot as plt

num_examples = 50
X = np.array([np.linspace(-2, 4, num_examples), np.linspace(-6, 6, num_examples)])
X += np.random.randn(2, num_examples)

plt.figure(figsize=(5,5))
plt.scatter(X[0], X[1])
plt.show()
```

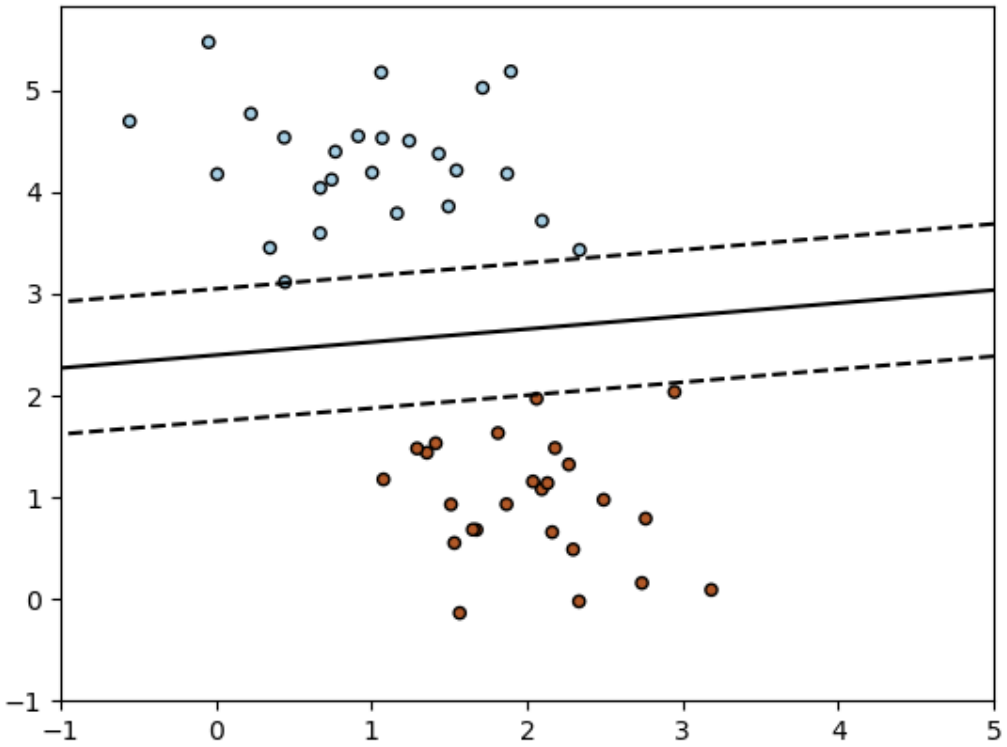


3) 경사하강법 (gradient.py)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.datasets.samples_generator import make_blobs

# we create 50 separable points
X, Y = make_blobs(n_samples=50, centers=2, random_state=0, cluster_std=0.60)

# fit the model
clf = SGDClassifier(loss="hinge", alpha=0.01, max_iter=200, fit_intercept=True)
clf.fit(X, Y)

# plot the line, the points, and the nearest vectors to the plane
xx = np.linspace(-1, 5, 10)
yy = np.linspace(-1, 5, 10)

X1, X2 = np.meshgrid(xx, yy)
Z = np.empty(X1.shape)
```

```
for (i, j), val in np.ndenumerate(X1):
    x1 = val
    x2 = X2[i, j]
    p = clf.decision_function([[x1, x2]])
    Z[i, j] = p[0]
levels = [-1.0, 0.0, 1.0]
linestyles = ['dashed', 'solid', 'dashed']
colors = 'k'
plt.contour(X1, X2, Z, levels, colors=colors, linestyles=linestyles)
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired,
            edgecolor='black', s=20)

plt.axis('tight')
plt.show()
```

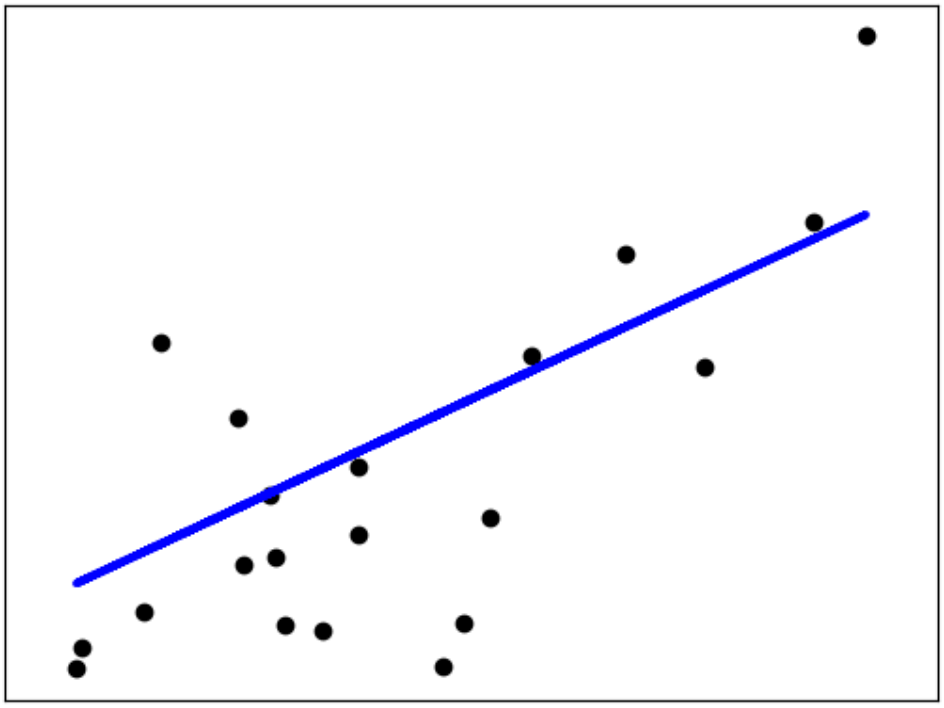
출처: http://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_separating_hyperplane.html#sphx-glr-auto-examples-linear-model-plot-sgd-separating-hyperplane-py



4) 선형 회귀 (regression.py)

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes = datasets.load_diabetes()



# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: ₩n', regr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
```

```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,    color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



5) 텐서플로를 이용한 선형 회귀 (regression_tf.py)

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

data = 100
```

```
dataset =[]

for i in range(data):
    x = np.random.normal(0.0, 0.5)
    y = x *0.1 + 0.3 + np.random.normal(0.0, 0.03)
    dataset.append([x, y])

x_raw = [v[0] for v in dataset]
y_raw = [v[1] for v in dataset]

plt.plot(x_raw, y_raw, 'ro')
plt.show()

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W*x_raw + b

loss = tf.reduce_mean(tf.square(y - y_raw))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

for step in range(4):
    sess.run(train)
    print (step, sess.run(W), sess.run(b))
    print (step, sess.run(loss))

    plt.plot(x_raw, y_raw, 'ro')
    plt.plot(x_raw, sess.run(W)*x_raw+sess.run(b))
    plt.xlabel('x')
    plt.xlim(-2, 2)
    plt.ylim(0.1, 0.6)
    plt.ylabel('y')
    plt.show()
```

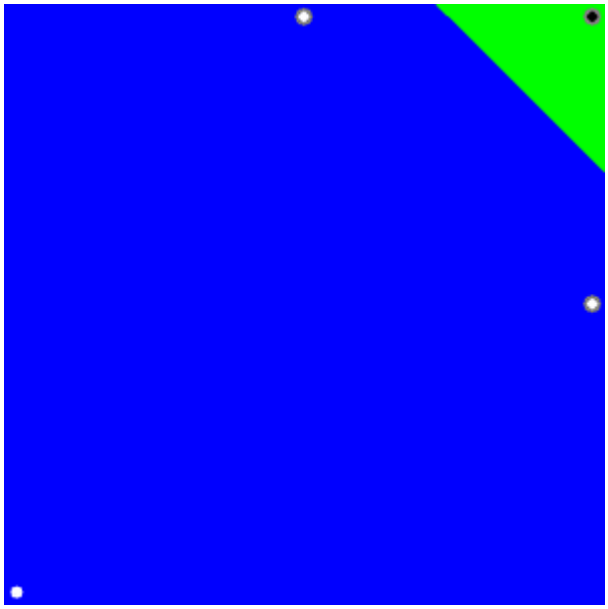
6) Support Vector Machine (OpenCV 사용) (svm_cv.py)

```
import cv2 as cv
import numpy as np
# Set up training data
labels = np.array([1, -1, -1, -1])
trainingData = np.matrix([[501, 10], [255, 10], [501, 255], [10, 501]], dtype=np.float32)
# Train the SVM
svm = cv.ml.SVM_create()
svm.setType(cv.ml.SVM_C_SVC)
svm.setKernel(cv.ml.SVM_LINEAR)
svm.setTermCriteria((cv.TERM_CRITERIA_MAX_ITER, 100, 1e-6))
svm.train(trainingData, cv.ml.ROW_SAMPLE, labels)
# Data for visual representation
width = 512
height = 512
image = np.zeros((height, width, 3), dtype=np.uint8)
# Show the decision regions given by the SVM
green = (0,255,0)
blue = (255,0,0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        sampleMat = np.matrix([[j,i]], dtype=np.float32)
        response = svm.predict(sampleMat)[1]
        if response == 1:
            image[i,j] = green
        elif response == -1:
            image[i,j] = blue
# Show the training data
thickness = -1
cv.circle(image, (501, 10), 5, ( 0, 0, 0), thickness)
cv.circle(image, (255, 10), 5, (255, 255, 255), thickness)
cv.circle(image, (501, 255), 5, (255, 255, 255), thickness)
cv.circle(image, ( 10, 501), 5, (255, 255, 255), thickness)
# Show support vectors
thickness = 2
sv = svm.getUncompressedSupportVectors()
for i in range(sv.shape[0]):
```

```
    cv.circle(image, (sv[i,0], sv[i,1]), 6, (128, 128, 128), thickness)
cv.imwrite('result.png', image) # save the image
cv.imshow('SVM Simple Example', image) # show it to the user
cv.waitKey()
```

출처: https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html

결과



7) Support Vector Machine (SKLEARN 사용) (svm_sklearn.py)

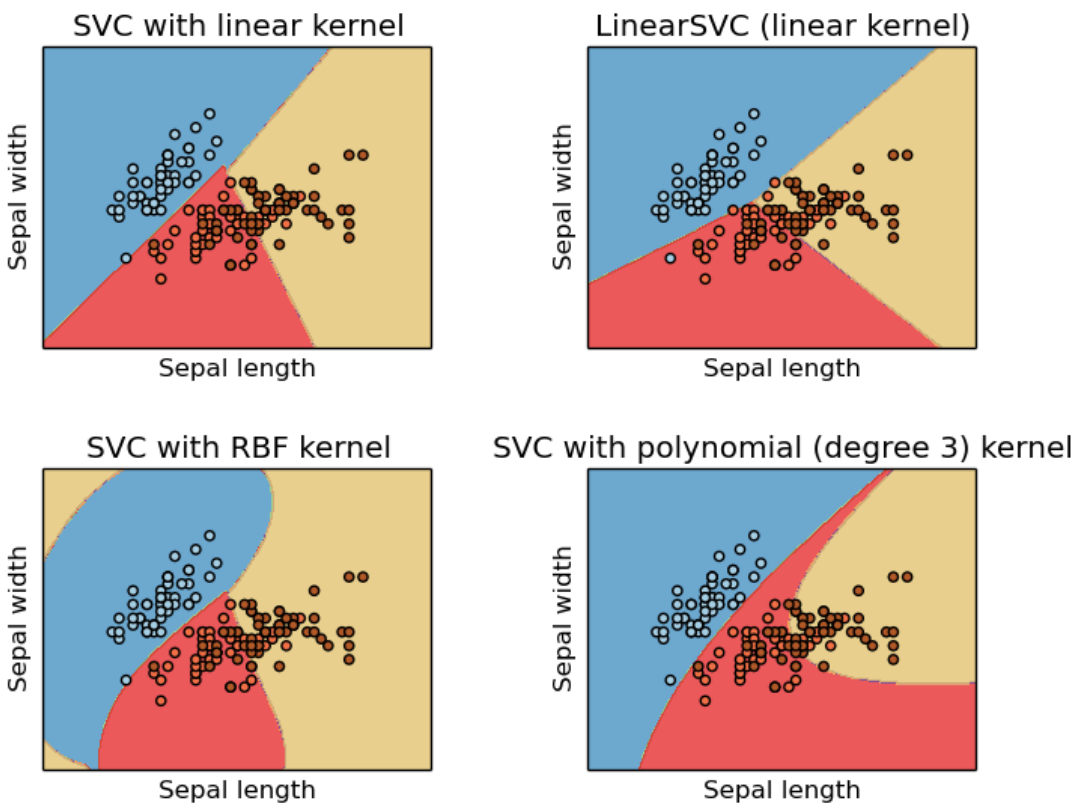
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

iris = datasets.load_iris()
X = iris.data[:,:2]
y = iris.target
svc = svm.SVC(kernel='linear', C=1,gamma=100).fit(X, y)
x_min, x_max = X[:,0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:,1].min() - 1, X[:,1].max() + 1
h = (x_max / x_min) / 100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
plt.subplot(1,1,1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
```

```
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```

출처: http://scikit-learn.org/0.16/auto_examples/svm/plot_iris.html

svc = svm.SVC(kernel='linear', C=1,gamma=100).fit(X, y)
위의 커널을 linear로 해보았다면, kernel='rbf'로 바꾸어보고, C=1, 100, 10000일 경우와 gamma = 1, 100, 10000을 비교해봅시다.

결과



8) K-means 클러스터링 (kmean.py)

K-means 알고리즘은 클러스터 (군집/모임)로 그룹화하기 위한 알고리즘으로, 각 클러스터에 있는

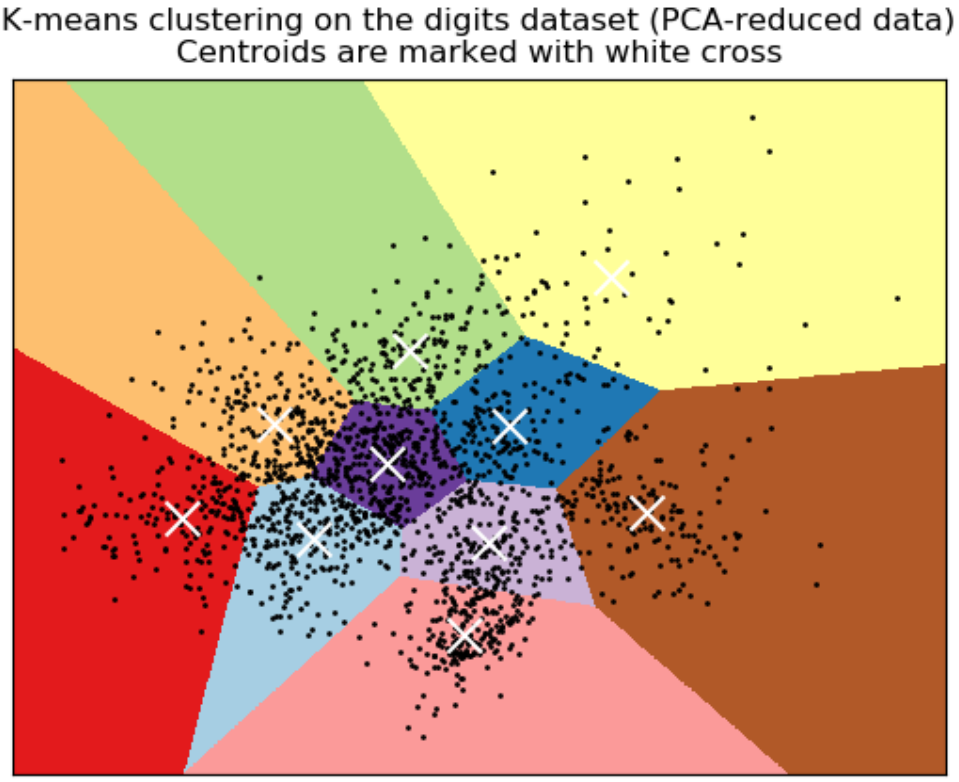
데이터들은 동일하거나 비슷한 특성으로 다른 그룹과 구별됩니다. K-means 알고리즘은 각 클러스터의 중심인 centroid를 결과로 내고, 각각의 데이터들은 이 centroid와의 거리를 통해 클러스터에 포함이 됩니다.

```
from time import time
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale

np.random.seed(42)

digits = load_digits()
data = scale(digits.data)

n_samples, n_features = data.shape
n_digits = len(np.unique(digits.target))
labels = digits.target

sample_size = 300

print("n_digits: %d, ₩t n_samples %d, ₩t n_features %d"
      % (n_digits, n_samples, n_features))



print(82 * '_')
print('init₩t₩ttime₩tinertia₩thomo₩tcompl₩tv-meas₩tARI₩tAMI₩tsilhouette')



def bench_k_means(estimator, name, data):
    t0 = time()
    estimator.fit(data)
    print('%-9s₩t%.2fs₩t%i₩t%.3f₩t%.3f₩t%.3f₩t%.3f₩t%.3f₩t%.3f'
          % (name, (time() - t0), estimator.inertia_,
```

```
                  metrics.homogeneity_score(labels, estimator.labels_),
                  metrics.completeness_score(labels, estimator.labels_),
                  metrics.v_measure_score(labels, estimator.labels_),
                  metrics.adjusted_rand_score(labels, estimator.labels_),
                  metrics.adjusted_mutual_info_score(labels,  estimator.labels_),
                  metrics.silhouette_score(data, estimator.labels_,
                                           metric='euclidean',
                                           sample_size=sample_size)))

bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
                  name="k-means++", data=data)

bench_k_means(KMeans(init='random', n_clusters=n_digits, n_init=10),
                  name="random", data=data)

# in this case the seeding of the centers is deterministic, hence we run the
# kmeans algorithm only once with n_init=1
pca = PCA(n_components=n_digits).fit(data)
bench_k_means(KMeans(init=pca.components_, n_clusters=n_digits, n_init=1),
                  name="PCA-based",
                  data=data)
print(82 * '_')

#
##############################################################################
#####
# Visualize the results on PCA-reduced data

reduced_data = PCA(n_components=2).fit_transform(data)
kmeans = KMeans(init='k-means++', n_clusters=n_digits, n_init=10)
kmeans.fit(reduced_data)

# Step size of the mesh. Decrease to increase the quality of the VQ.
h = .02        # point in the mesh [x_min, x_max]x[y_min, y_max].

# Plot the decision boundary. For that, we will assign a color to each
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

```
# Obtain labels for each point in mesh. Use last trained model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
                extent=(xx.min(), xx.max(), yy.min(), yy.max()),
                cmap=plt.cm.Paired,
                aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
                 marker='x', s=169, linewidths=3,
                 color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA-reduced data)₩n'
              'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

출처: http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html



9) 텐서플로를 이용한 손글씨 인식 (Training)

(cv) $ cd model_mnist

```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.optimizers import Adam
from keras.utils import np_utils

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(
```

```
    X_train.shape[0], X_train.shape[1], X_train.shape[2], 1).astype('float32')
X_test = X_test.reshape(
    X_test.shape[0], X_test.shape[1], X_test.shape[2], 1).astype('float32')

# normalize inputs from 0-255 to 0-1
X_train /= 255
X_test /= 255

# one hot encode
number_of_classes = 10
y_train = np_utils.to_categorical(y_train, number_of_classes)
y_test = np_utils.to_categorical(y_test, number_of_classes)

# create model
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(
    X_train.shape[1], X_train.shape[2], 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))

# Compile model
model.compile(loss='categorical_crossentropy',
                optimizer=Adam(), metrics=['accuracy'])

# Fit the model
model.fit(X_train, y_train, validation_data=(
    X_test, y_test), epochs=5, batch_size=200)

# Save the model
model.save('models/mnistCNN.h5')

# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
```

```
print(metrics)
```

10) 텐서플로를 이용한 손글씨 인식 (Testing)

압축된 파일은 blog.naver.com/roboholic84에 업로드 했습니다.

```
from keras.models import load_model
model = load_model('models/mnistCNN.h5')

from PIL import Image
import numpy as np

# Test all 10 images
for index in range(10):
    img = Image.open('data/' + str(index) + '.png').convert("L")
    img = img.resize((28, 28))
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1, 28, 28, 1)
    # Predicting the Test set results
    y_pred = model.predict(im2arr)
    print(y_pred)
```

출처: https://medium.com/coinmonks/handwritten-digit-prediction-using-convolutional-neural-networks-in-tensorflow-with-keras-and-live-5ebddf46dc8

11) 이미지넷을 이용한 사물인식 시스템

이미지넷 설치하기

```
(cv) git clone https://github.com/tensorflow/models.git
```

다 설치가 끝난 후에는 다음의 폴더로 이동합니다.

cd models/tutorials/image/imagenet

그리고, 파이썬 코드를 실행하면 내장되어 있는 판다곰에 대한 분류 결과를 확인할 수 있습니다.

python3 classify_image.py

이 때, 테스트삼아 분류하고자하는 이미지를 다음의 폴더에 넣고 코드를 수정할 수 있습니다. 다

른 경로여도 상관없습니다.

```
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $ ls
BUILD  classify_image.py
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $
```

구글에서 "불고기" 이미지를 다운로드 받아보았습니다.

다운받은 경로는 /home/pi로 하였습니다.

Nano 에디터를 이용해서 파이썬 코드를 편집합니다. Cropped_panda.jpg를 다운받은 testimage.jpeg로 바꾸겠습니다. 이 때, 경로 설정도 유의해야 합니다.

```
(cv) $ nano classify_image.py
```

```
def main(_):
  maybe_download_and_extract()
  image = (FLAGS.image_file if FLAGS.image_file else
           os.path.join(FLAGS.model_dir, 'cropped_panda.jpg'))
  run_inference_on_image(image)
```

```
def main(_):
  maybe_download_and_extract()
  image = (FLAGS.image_file if FLAGS.image_file else
           os.path.join(FLAGS.model_dir, '/home/pi/testimage.jpeg'))
  run_inference_on_image(image)
```

CTRL+X, Y, 엔터 순으로 저장한 후, python3 classify_image.py를 실행하면 다음과 같이 inception 이라는 데이터베이스를 다운로드 받는 것을 확인할 수 있습니다.

```
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $ ls
BUILD  classify_image.py  testimage.jpeg
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $ nano classify_image.py
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $ python3 classify_image.p
y
/home/pi/.virtualenvs/cv/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarni
ng: compiletime version 3.4 of module 'tensorflow.python.framework.fast_tensor_u
til' does not match runtime version 3.5
  return f(*args, **kwds)
/home/pi/.virtualenvs/cv/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarni
ng: builtins.type size changed, may indicate binary incompatibility. Expected 43
2, got 412
  return f(*args, **kwds)
>> Downloading inception-2015-12-05.tgz 2.0%
```

다음과 같은 결과를 얻을 수 있습니다. 다만, 아쉬운 점은 plate가 45.5%로 분류를 했다는 것인데, 일반적으로 80% 이상이 나오면 우리가 생각하는 것과 비슷한 범위로 분류를 하는 것 같습니다.

```
plate (score = 0.45555)
mixing bowl (score = 0.06820)
hen-of-the-woods, hen of the woods, Polyporus frondosus, Grifola frondosa (score
 = 0.03732)
consomme (score = 0.03534)
frying pan, frypan, skillet (score = 0.03113)
(cv) pi@raspberrypi:~/models/tutorials/image/imagenet $
```