Additonal Frameworks

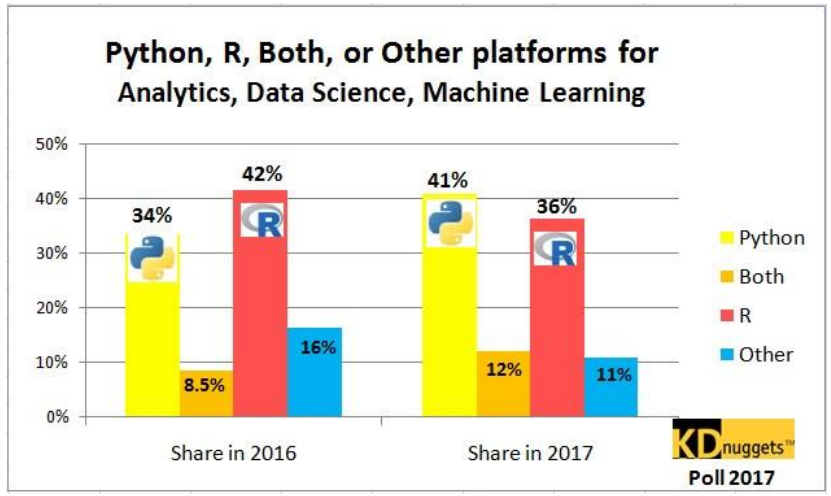
Contents

- Google Colab
- Numpy
- Pytorch basics

1-1

Google Colab 사용법

Programming language for Machine Learning



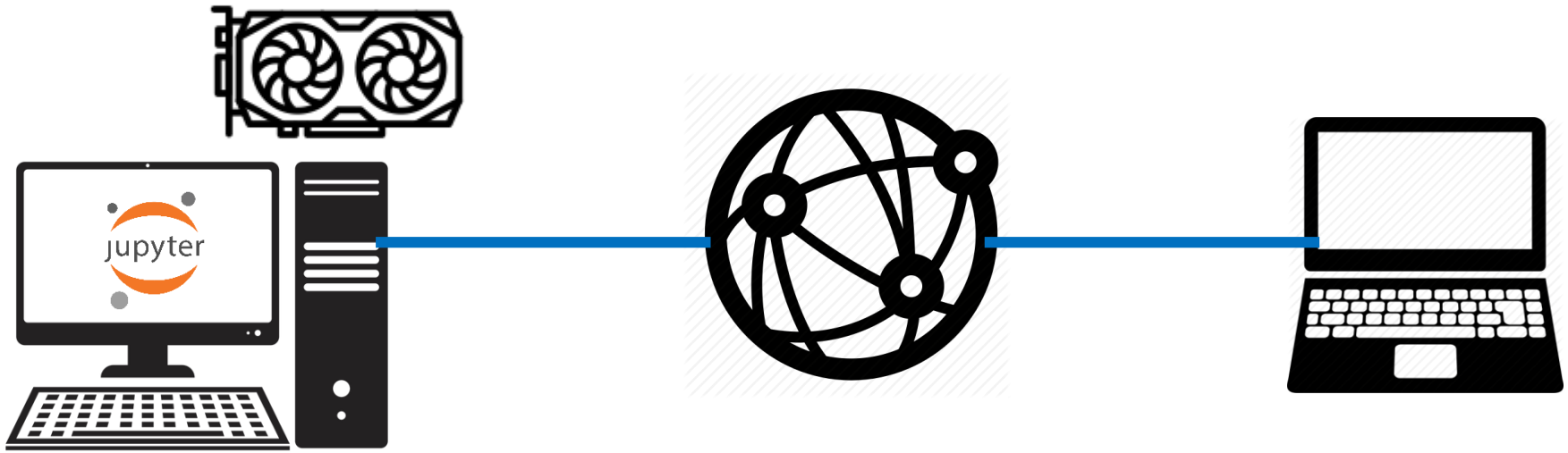
출처 - http://artificialintelligencemania.com/2018/07/02/the-best-programming-language-for-machine-learning/

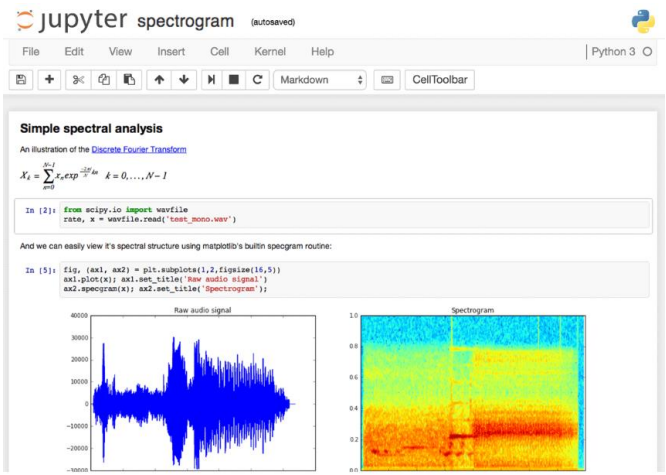
- Python
  - ML 연구 분야에 있어서 대체하기 어려운 프로그래밍 언어
  - Tensorflow, Pytorch 등의 딥러닝 프레임워크
  - Numpy, Jupyter Notebook, Matplotlib, Pandas, …

Jupyter Notebook?

- 웹 브라우저에서 파이썬 코드를 작성하고 실행해 볼 수 있는 개발도구
  - 원격 코딩 가능
  - 코드 블록 단위로 실행 / 디버깅
  - Text block을 이용한 문서화
  - Figure plotting 등 GUI

147.46.123.123:8888

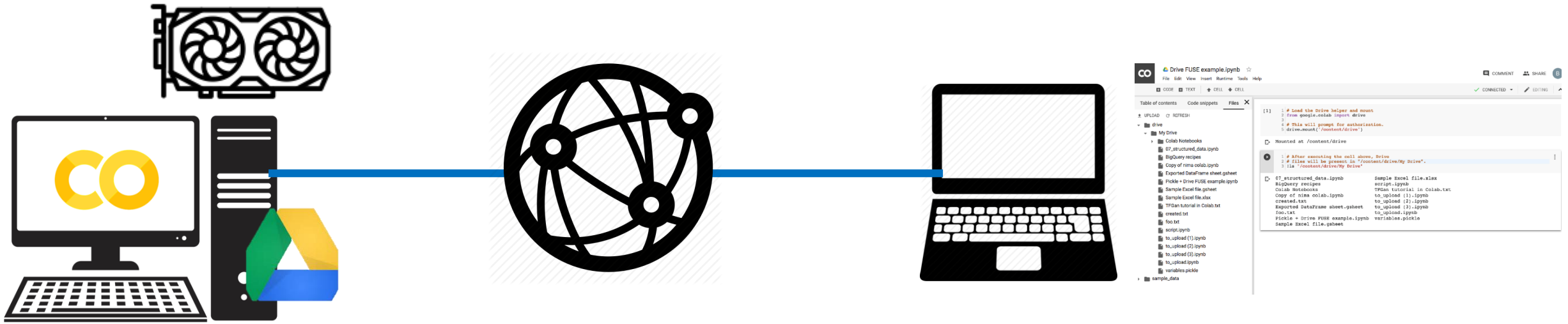




147.46.123.123

Google Colab?

- Google Colaboratory = Google Drive + Jupyter Notebook
  - 구글 계정 전용의 가상 머신 지원 – **GPU 포함**
  - Google drive 문서와 같이 링크만으로 접근 / 협업 가능
  - 코드 실행 시 딜레이 존재



Google server

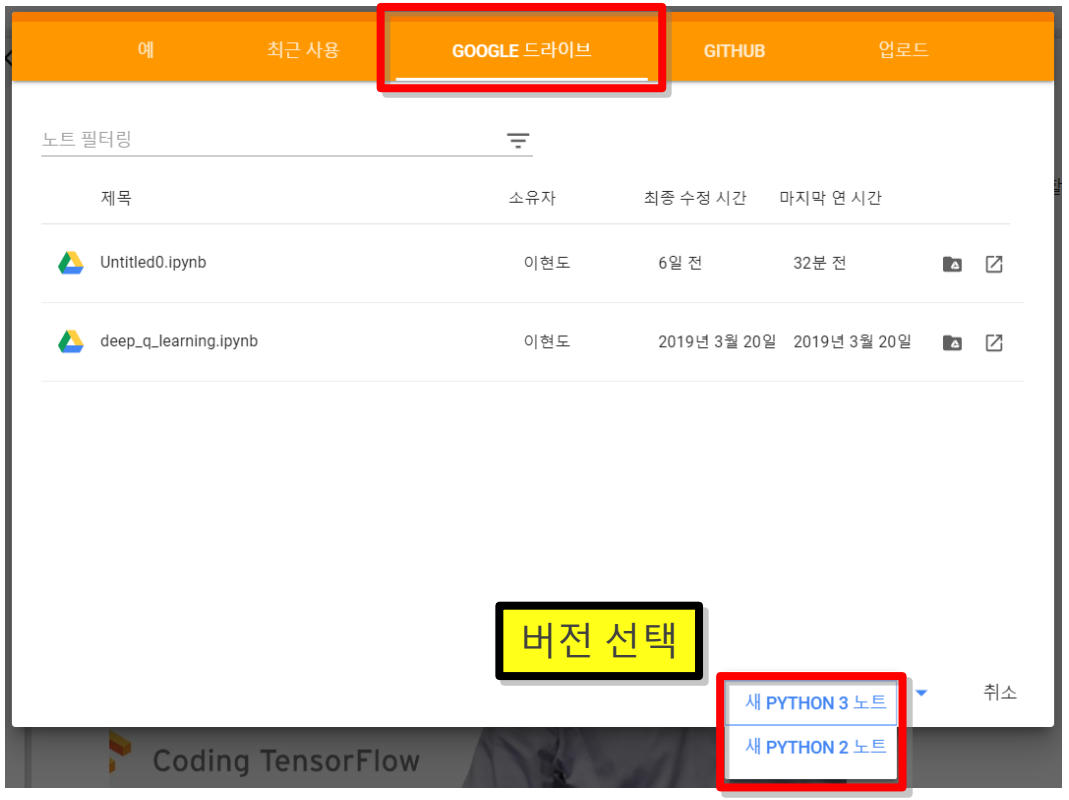
Google Colab - 사용법

- 개인 구글 계정 필요

- Colab과 Jupyter Notebook 사용 방법은 유사한 부분이 많음
  - 실습 수업에서는 Colab 위주로 설명
  - GPU가 내장된 서버를 사용할 수 있을 시 로컬에서 작업을 권장

Google Colab - 사용법

- 파일 생성/접근 방법
  - 개인 구글 계정으로 접속
  - https://colab.research.google.com 접속
  - GOOGLE 드라이브 탭 이동
  - 새 PYTHON 노트 선택



Google Colab - 사용법

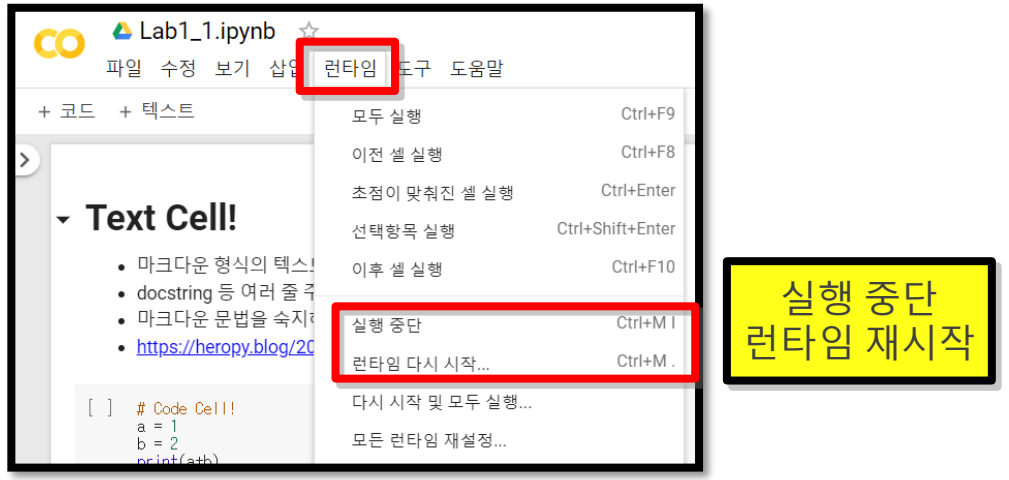
- 파일 이름 변경

- Code cell, Text cell
  - .ipynb 파일은 code cell과 text cell로 구성
  - 각 셀 하단에 마우스를 대거나, 화면 좌상단 버튼으로 셀 추가 가능
  - 셀 선택(마우스) 후 셀 우상단 삭제버튼으로 셀 삭제 가능

Google Colab - 사용법

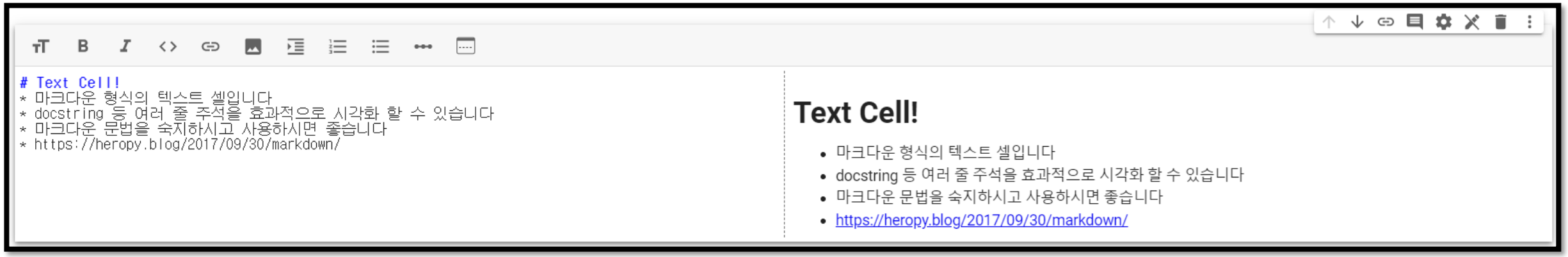
- Code cell
  - 일반적인 파이썬 코딩 방식과 동일
  - 각 셀은 한번에 실행할 단위를 뜻함
  - 실행 이후에도 메모리는 유지되어 다른 셀 실행 시 영향을 줌
    - 런타임 다시 시작 시 초기화

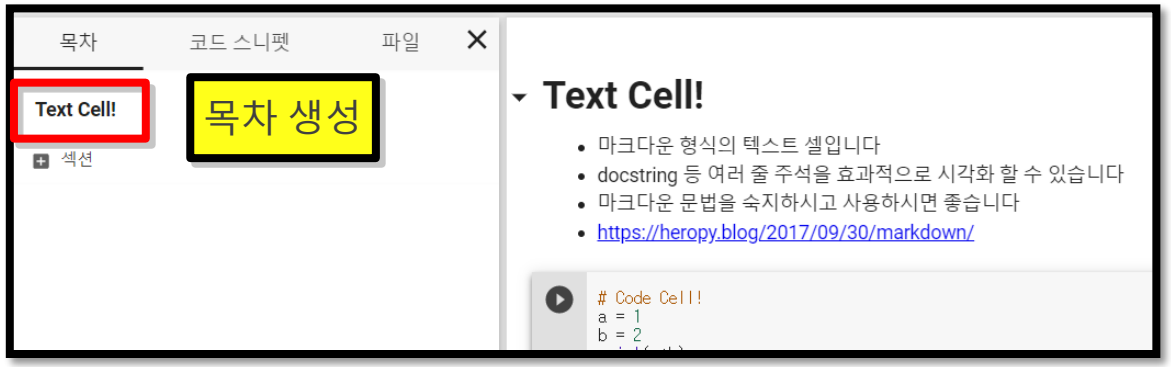
  - 상단 메뉴의 런타임
    - 실행 중인 셀 중단
    - 런타임 다시 시작



Google Colab - 사용법

- Text cell
  - 여러 줄 주석의 효과적인 시각화
  - 마크다운(Markdown) 문법
  - 자동 목차 생성



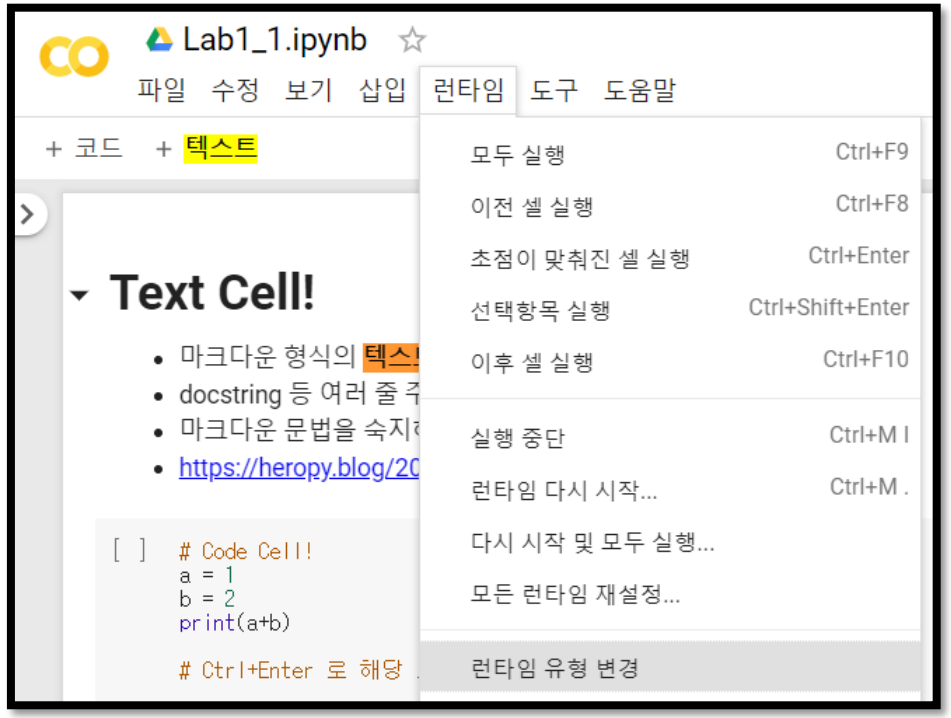


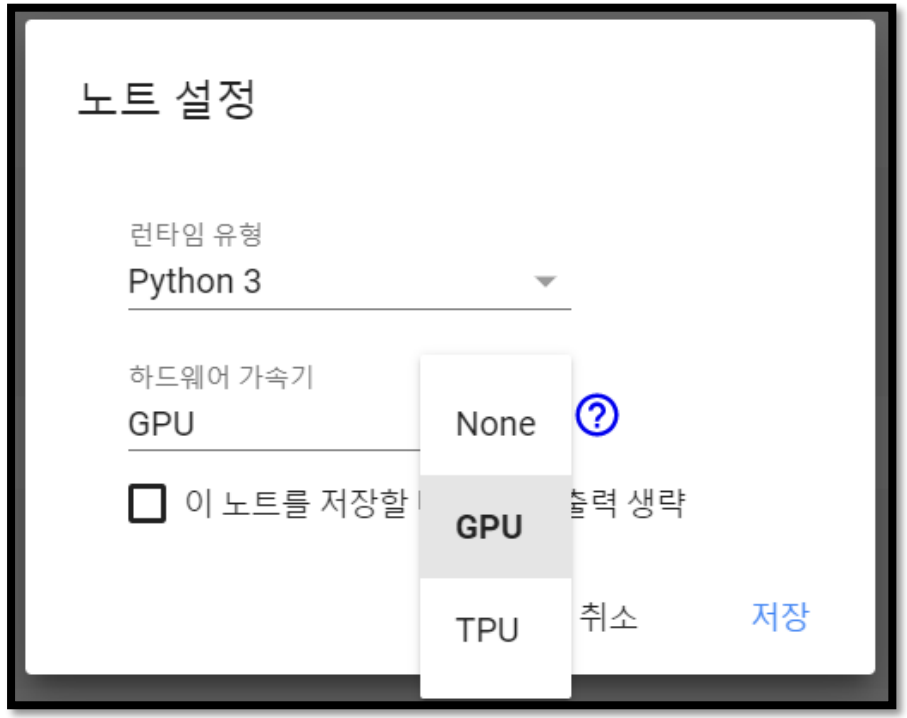
Google Colab - 사용법

- 단축키
  - 대부분의 작업은 단축키로 실행 가능
  - 단축키 설정 가능
  - 단축키 설정화면 – Ctrl+M H
  - 유용한 단축키
    - 코드 셀 생성 – Ctrl+M A(B)
    - 코드 셀 실행 – Ctrl+Enter
    - 셀 삭제 – Ctrl+M D
    - 실행중인 셀 중단 – Ctrl+M I
    - 런타임 다시 시작 – Ctrl+M .
    - 코드(텍스트) 셀로 변환 – Ctrl+M Y(M)
    - 마지막 셀 작업 실행취소 – Ctrl+Shift+Z

Google Colab - 사용법

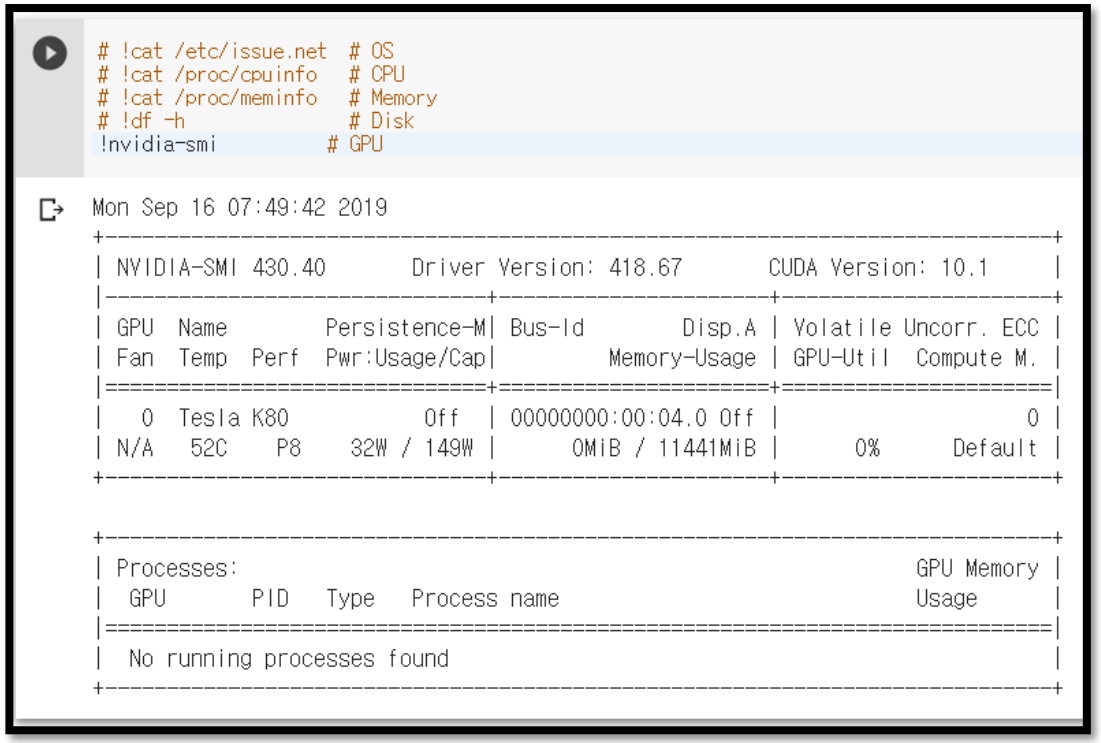
- GPU 설정
  - 런타임 -> 런타임 유형 변경 -> 하드웨어 가속기를 GPU로 변경
  - 유의사항 – GPU는 최대 12시간 실행을 지원
    - 12시간 실행 이후에는 런타임 재시작으로 VM을 교체해야 함





Google Colab - 사용법

- 명령어 실행하기
  - !코드 셀에 를 붙이고 터미널 명령어를 입력하여 실행하면 터미널에서 실행하는 것과 같은 결과가 출력됨
  - 예외로 cd 명령어는 %cd /your/desired/path

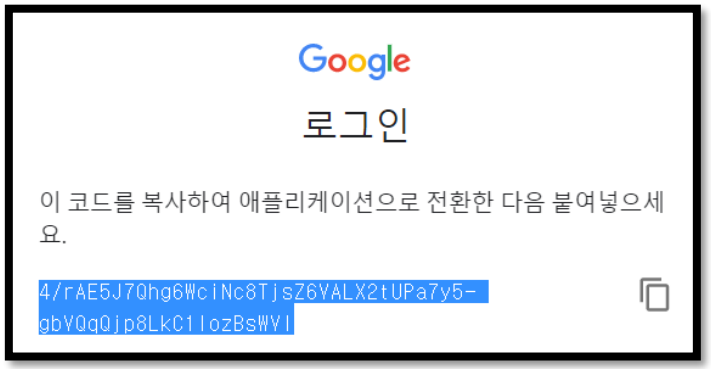


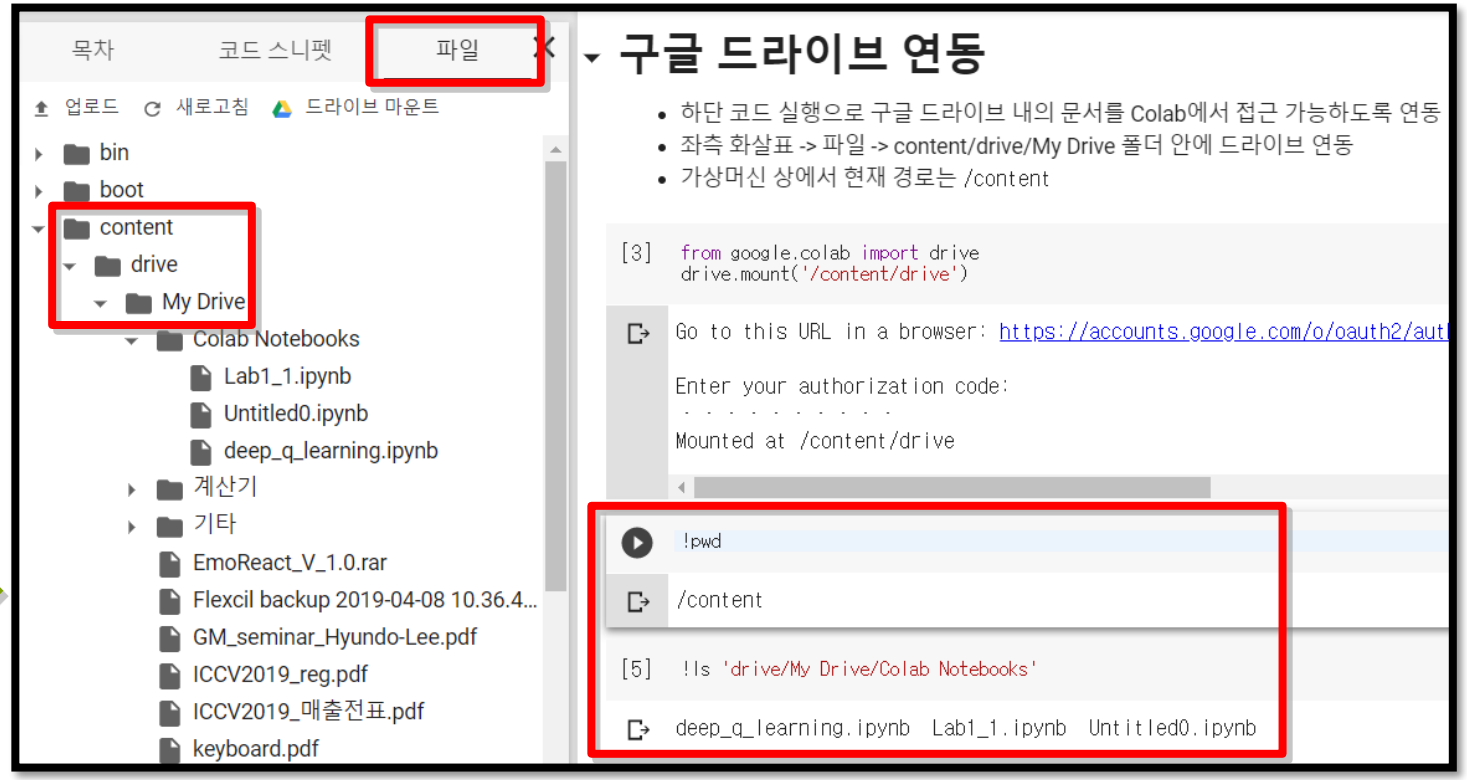
```
# !cat /etc/issue.net  # OS
# !cat /proc/cpuinfo   # CPU
# !cat /proc/meminfo   # Memory
# !df -h               # Disk
!nvidia-smi          # GPU
```

```
Mon Sep 16 07:49:42 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 430.40       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   52C    P8    32W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Google Colab - 사용법

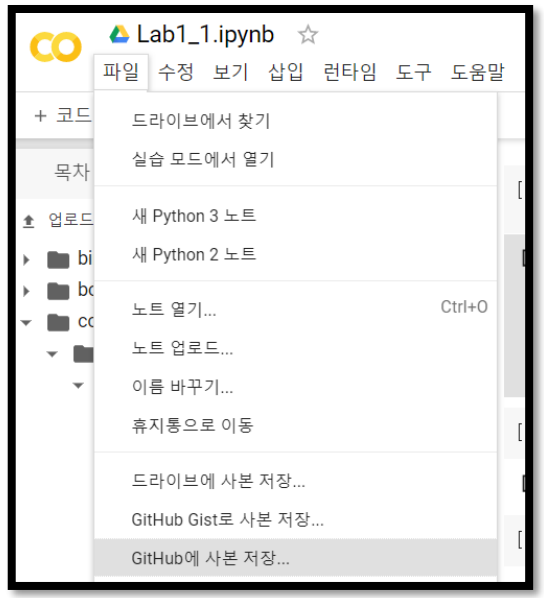
- 구글 드라이브 연동
  - 간단한 인증 절차 이후 구글 드라이브의 파일을 Colab에서 접근 가능





Google Colab - 사용법

- Github 연동
  - 단일 .ipynb 파일을 clone 하는 방법
    - https://github.com/~~~ 부분을 https://colab.research.google.com/github/~~~ 로 교체
    - 파일 -> 드라이브에 사본 저장
  - 전체 repository cloning
    - !git clone project.git
  - github repository에 파일을 올리는 방법
    - 파일 - Github에 사본 저장 선택
    - 저장소, 브랜치, 경로 지정



Google Colab - tips

- 장시간 사용할 때의 issue들

> - The **'maximum lifetime'** of a running notebook is **12 hours** (browser open)
> - An **'Idle'** notebook instance cuts-off after **90 minutes**
> - You can have a maximum of **2 notebooks** running concurrently
> - If you close the notebook window and open it while the instance is still running, the cell outputs and variables will still persist. However if the notebook instance has been recycled, your cell outputs and variables will no longer be available.

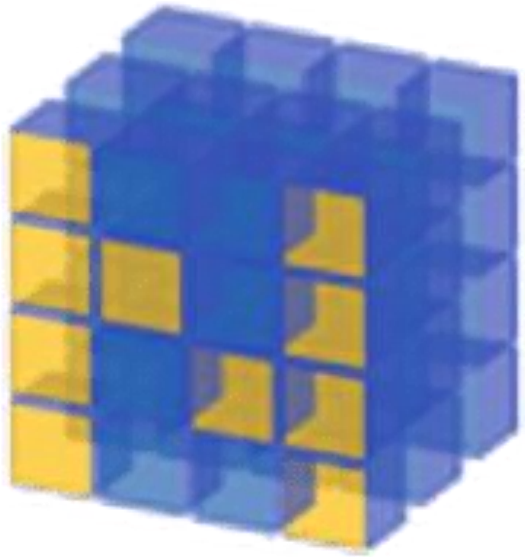
  - GPU 사용/미사용 관계없이 최대 12시간
  - 아무것도 안 하는 idle 상태 돌입 이후 90분에 런타임 자동으로 shutdown
    - Shutdown 전에 창을 끄고 다시 켜도 런타임은 유지됨
  - 동시에 최대 2개의 notebook 가동 가능

Google Colab - tips

- Jupyter로 코딩하고 싶은데 Colab GPU만 쓰고 싶은 경우
  - 로컬에서 작업 후 drive에 올리고 Colab으로 실행
  - 학습된 모델 weight만 다운로드 후 로컬에서 분석

- Jupyter도 쓰기 싫은 경우
  - Drive에 올리고 실행해야 할 파일(ex: train.py)를 Colab에서 커맨드로 실행 (!train.py)

- 그냥 Colab 쓰는게 싫은 경우
  - 학기마다 GPU 서버 사용 신청을 할 수 있음(서울대학교 구성원)



1-2

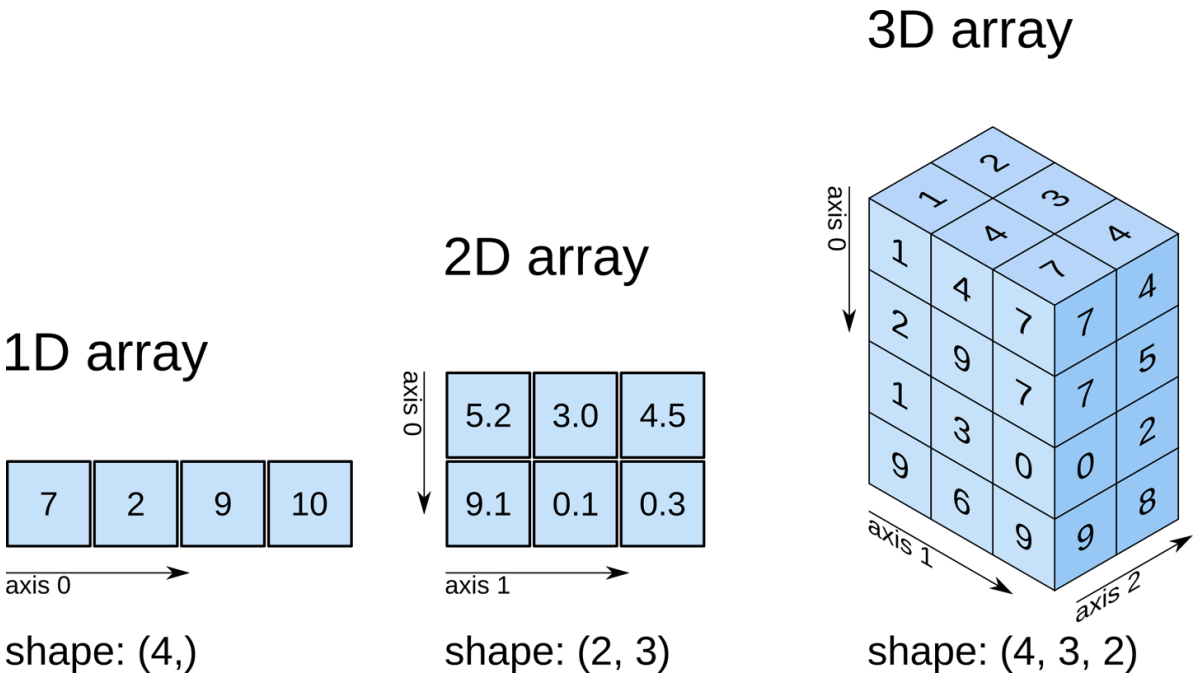
Numpy, Pytorch tensor

Numpy

- ML 프로그래밍의 데이터 자료형은 기본적으로 multi-dimensional array (혹은 tensor)

- Numpy는 multi-dimensional array 연산을 편리하게 하기 위한 라이브러리
  - Numpy 모듈 설치 - *pip3 install numpy*
    - Colab에는 기본적으로 설치되어 있다.
  - Import - *import numpy (as np)*

Numpy array



출처:
https://www.safaribooksonline.com/library/view/elegant-scipy/9781491922927/ch01.html

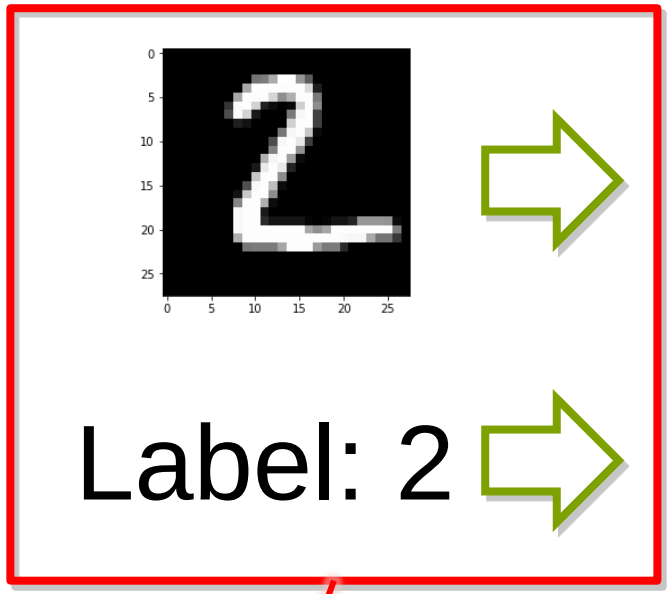
- numpy array는 모양(shape)과 데이터 타입(dtype)을 가진다.
  - 길이가 n인 정수 벡터 – shape: n, dtype: *np.int32*
  - $n \times m$ 실수 행렬 – shape: (n, m), dtype: *np.float32(64)*
  - $n \times m \times k$ 복소수 텐서 – shape: (n, m, k), dtype: *np.complex64*

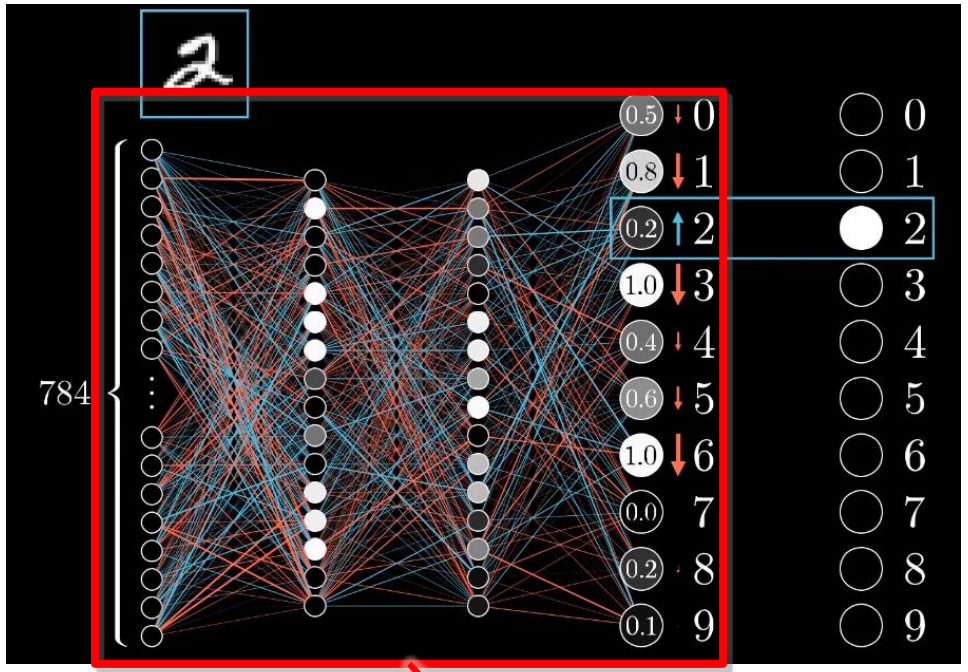
1-3

Pytorch basics

Pytorch basics

- Pytorch를 활용한 딥러닝 프로그램의 기본 구조





Dataset
Input, Labels

Model
Structure, Weights

Output -> Label
Optimizing model weight

Training Phase

Test Phase
Evaluating output

Pytorch basics

- Pytorch를 활용한 딥러닝 프로그램의 기본 구조
  - Model(nn.Module) : Class
    - 전체적인 인공신경망의 구조를 선언하는 부분
    - __init__, forward 등의 method는 꼭 필요함
  - Dataset : Class
    - 학습에 필요한 데이터셋을 선언하는 부분
    - __init__, __len__, __getitem__의 method는 꼭 필요함
  - Training
    - 학습을 진행하는 부분
  - {Validation & Test}
    - 학습된 결과를 확인하는 부분

Pytorch basics

- Model

```
Class Model(nn.Module):
    def __init__(self, PARAMETERS):
        super(Model, self).__init__()
        …SOME DECLAIRATION…


    def forward(self, INPUTS):
        …SOME COMPUTATIONS…
        return OUTPUT
```

Pytorch basics

- Dataset

```
Class Dataset(Dataset):
    def __init__(self, PARAMETERS):
        …SOME DECLAIRATION…


    def __len__(self):
        return length_of_dataset


    def __getitem__(self, index):
        return Dataset[index]
```

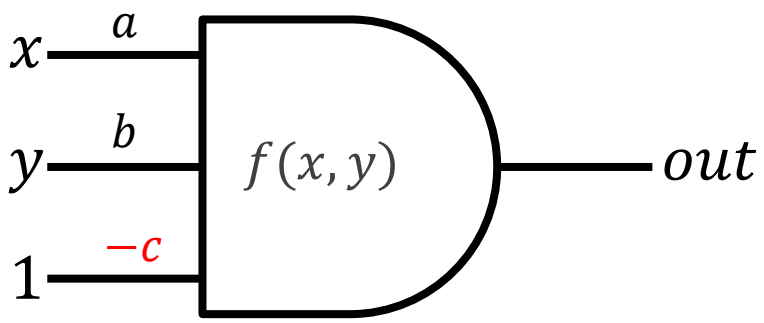
Pytorch basics

- Training part

```
DECLAIR DATASET, DATALOADER
DECLAIR HYPERPARAMETERS

model = Model(PARAMETERS)
optimizer = OPTIMIZER(model.parameters(), lr=LEARNING_RATE)

for epoch in range(NUM_EPOCHES):
    for (INPUTS, REAL_VALUES) in DATALOADER:
        optimizer.zero_grad()
        OUTPUTS = model(INPUTS)
        loss = LOSS_FUNCTION(OUTPUT, REAL_VALUES)
        loss.backward()
        optimizer.step()
```

Logic Gate Example

- Single Perceptron Separator



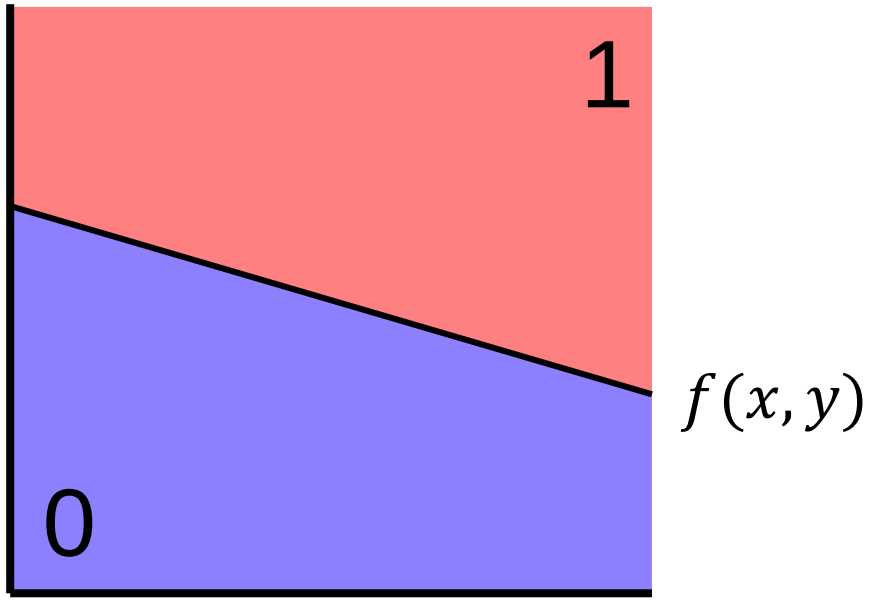
  - 단일 퍼셉트론 : $\text{out} = f(x,y) = g(ax + by - c \times 1) = g(ax + by - c)$

Bias == 역치

  - Activation function $g(x)$: sigmoid, relu, etc. (Softmax는 포함 안됨)

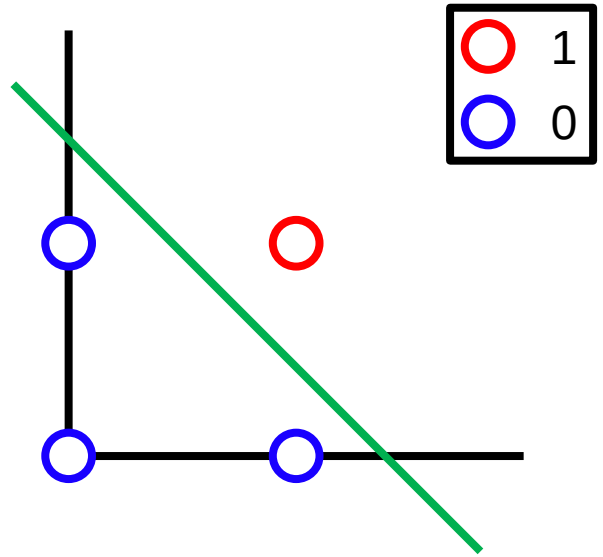
Logic Gate Example

- Single Perceptron Separator
  - 단일 퍼셉트론 : $\text{out} = f(x, y) = g(ax + by - c \times 1) = g(ax + by - c)$
  - 공간 분할 측면에서의 퍼셉트론 (Sigmoid)



Logic Gate Example

- Single Perceptron Separator
  - 문제 : 단순한 Boolean 함수 근사(AND, OR, XOR)
  - 입력
    - 노이즈를 포함하는 0 혹은 1의 값을 가지는 x, y
  - 출력
    - 예측된 함수의 결과 : val_(val^*)
  - Loss function
    - 실제 함수의 결과와 예측된 결과의 차이를 최소화
    - $\mathcal{L} = \sum_{n=1}^{N}(val - val^*)^2$ (L2 loss, reduce sum)



Logic Gate Example

- Imports
  - torch : 가장 기본적인 pytorch 모듈
  - torch.nn : 인공신경망 관련 함수/클래스 등이 선언된 모듈
  - torch.optim : optimizer가 정의된 모듈
    - Adam : Adam Optimizer
  - torch.utils.data.* : 학습 세팅을 편하게 해주기 위한 dataset의 class가 선언된 모듈들
  - matplotlib : 결과를 시각화 하는 모듈

```
import torch
import torch.nn as nn
from torch.optim import Adam
from torch.utils.data.dataset import Dataset
from torch.utils.data import DataLoader
import numpy as np
import matplotlib.pyplot as plt
```

Logic Gate Example

- Class Separator
  - torch.nn.Parameter(*TENSOR*)
    - 학습의 대상이 되는 변수(w)를 설정
  - torch.Tensor(*INITIAL_VALUE*)
    - Tensor의 형태로 initial value를 변환
    - Pytorch에서는 Tensor로 연산
    - Tensor.item() : 값을 가져옴
  - torch.sigmoid
    - 모든 원소에 sigmoid 함수 연산 수행
  - forward(*INPUT*)
    - 모델이 계산할 main function

```
class Separator(nn.Module):
  def __init__(self, func = None):
    super(Separator, self).__init__()
    self.a = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    self.b = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    self.c = torch.nn.Parameter(torch.Tensor([np.random.normal()]))
    if func is not None:
      self.func = func
    else:
      self.func = torch.sigmoid

  def forward(self,x,y):
    val_ = self.func(self.a * x + self.b * y - self.c)
    return val_, (self.a.item(), self.b.item(), self.c.item())
```

Logic Gate Example

- Class DataGenerator
  - val : (x,y)값이 (0,0), (0,1), (1,0), (1,1)일 때의 값을 저장함
  - x,y : {0,1}에서 값을 가져오고 N(0,0.16)의 노이즈를 더함

```
class DataGenerator(Dataset):

  def __init__(self,type_,length,custom=None):
    self.length=length
    if type_ == 'and':
      self.val_l = [0,0,0,1]
    elif type_ == 'or':
      self.val_l = [0,1,1,1]
    elif type_ == 'xor':
      self.val_l = [0,1,1,0]
    elif type_ == 'custom' and custom is not None:
      self.val_l = custom
    else:
      self.val_l = [0,0,0,0]

    self.dataset = []
    for i in range(length):
      x = np.random.normal(i%2,0.16)
      y = np.random.normal((i//2)%2,0.16)
      val = self.val_l[i%4]
      self.dataset.append((x,y,val))

  def get_dataset(self):
    return self.dataset

  def __len__(self):
    return self.length

  def __getitem__(self,idx):
    x,y,val = self.dataset[idx]
    return (torch.Tensor([x]),torch.Tensor([y]), torch.Tensor([val]))
```

Logic Gate Example

- Training setup
  - LEARNING_RATE
    - 한번의 학습을 통해 변화시키는 정도
    - 문제마다 적절한 값 설정 필요
  - BATCH_SIZE
    - 한번에 전체 데이터를 넣는 것은 크기가 너무 클 뿐 아니라, 좋지 않음
    - Batch의 크기를 설정하여 random sampling
  - NUM_EPOCHES
    - 전체 반복할 총 epoch의 수
    - 1 epoch == 데이터셋의 한 iter

```
DATASET = DataGenerator('and',1000)
LEARNING_RATE = 0.01
BATCH_SIZE = 20
NUM_EPOCHES = 20
NUM_WORKERS = 4
GRAPH_X = np.linspace(-1.0,2,2)

params = {
    'batch_size': BATCH_SIZE,
    'shuffle': True,
    'num_workers': 4,
}

dataloader = DataLoader(DATASET,**params)
model = Separator().cuda()
optimizer = Adam(model.parameters(), lr=LEARNING_RATE)
```

Logic Gate Example

- Training part
  - tot_loss
    - 한 epoch에서의 모든 loss를 계산
    - 결과 출력을 위함
  - Tensor.cuda()
    - 기본적으로 Tensor를 선언하면 CPU에 할당
    - .cuda()를 통해 GPU로 계산하도록 함
  - optimizer.zero_grad()
    - 각 batch마다 gradient 값을 초기화

```
for epoch in range(NUM_EPOCHES):
  tot_loss = 0
  for x, y, val in dataloader:
    x = x.cuda()
    y = y.cuda()
    val = val.cuda()
    optimizer.zero_grad()
    val_, params = model(x,y)
    loss = torch.sum(torch.pow(val-val_,2))

    loss.backward()
    optimizer.step()
    tot_loss+=loss.item()
  print("Loss : {:.5f}".format(tot_loss/len(DATASET)))

  if epoch % 5 == 4:
    for item in DATASET.get_dataset():
      x,y,val = item
      if val ==1:
        plt.scatter(x,y,c='red')
      else:
        plt.scatter(x,y,c='blue')
    plt.plot(GRAPH_X,-(GRAPH_X*params[0]-params[2])/(params[1]+1e-10))
    plt.show()
```

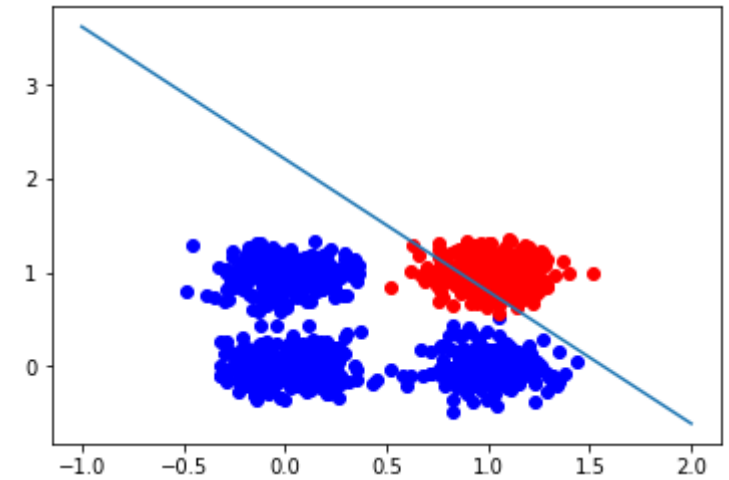
Logic Gate Example

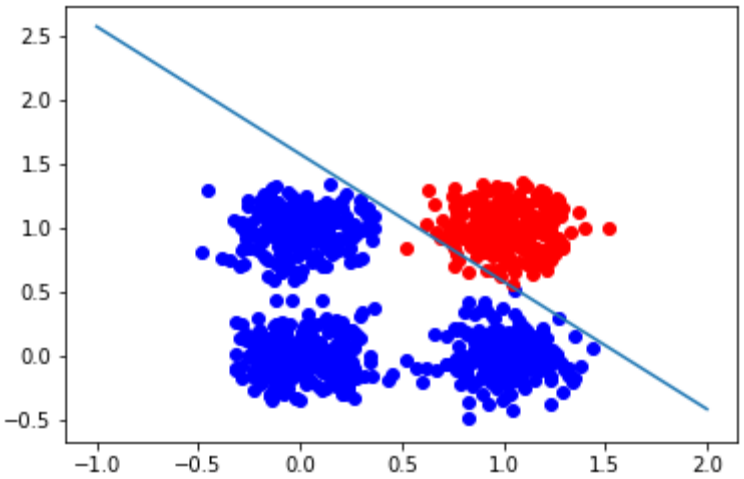
- Training part
  - loss.backward()
    - 계산된 tensor에 대해 역으로 계산하며 gradient를 구함
  - optimizer.step()
    - gradient를 활용하여 backpropagation을 수행, w를 update
- Test part
  - plt.scatter
    - 그래프에 산포도를 찍는 함수
  - plt.plot
    - 그래프에 선을 긋는 함수
  - plt.show
    - 그래프를 보여주는 함수

```
for epoch in range(NUM_EPOCHES):
  tot_loss = 0
  for x, y, val in dataloader:
    x = x.cuda()
    y = y.cuda()
    val = val.cuda()
    optimizer.zero_grad()
    val_, params = model(x,y)
    loss = torch.sum(torch.pow(val-val_,2))

    loss.backward()
    optimizer.step()
    tot_loss+=loss.item()
  print("Loss : {:.5f}".format(tot_loss/len(DATASET)))

  if epoch % 5 == 4:
    for item in DATASET.get_dataset():
      x,y,val = item
      if val ==1:
        plt.scatter(x,y,c='red')
      else:
        plt.scatter(x,y,c='blue')
    plt.plot(GRAPH_X,-(GRAPH_X*params[0]-params[2])/(params[1]+1e-10))
    plt.show()
```

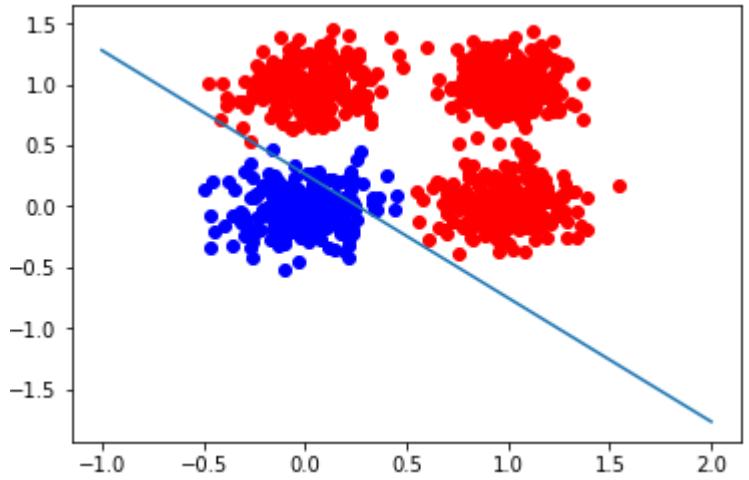
Logic Gate Example

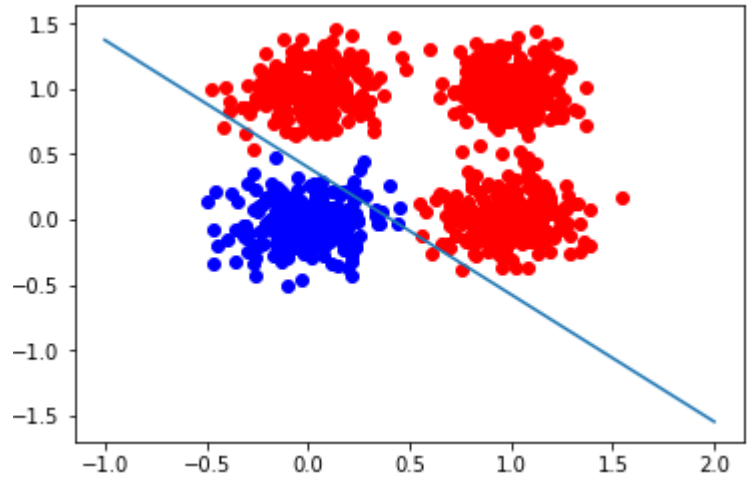
- 결과 1 (AND)





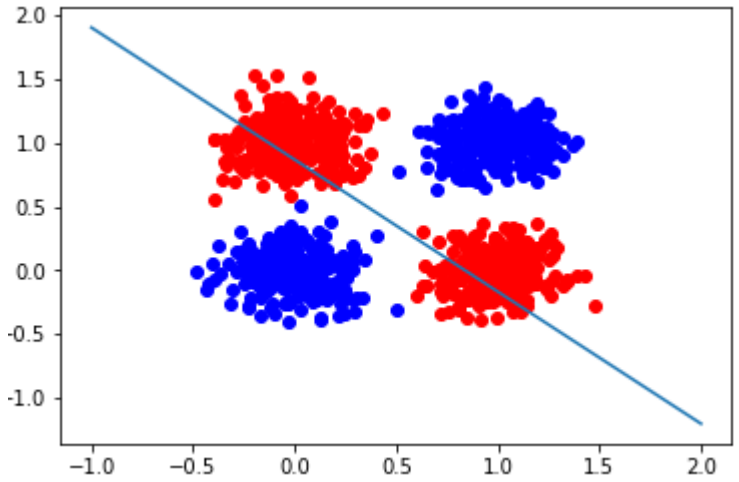
- 결과 2 (OR)

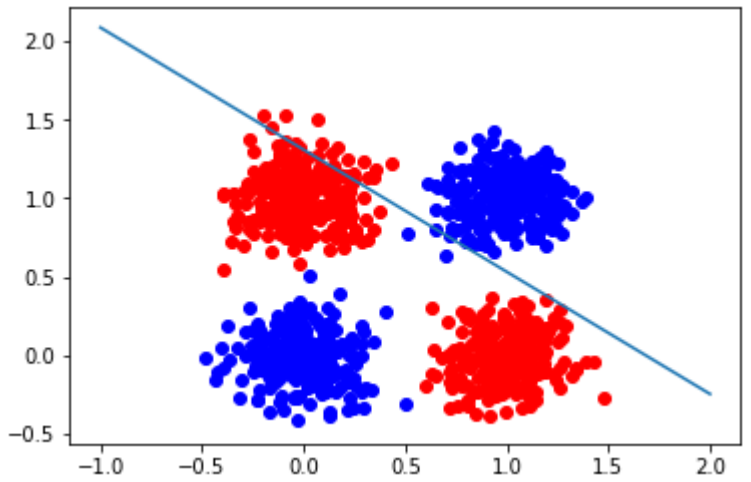




Logic Gate Example

- 결과 3 (XOR)
  - 단일 퍼셉트론만으로는 XOR에 해당하는 함수를 근사할 수 없음
  - 해결방법?





Codes

- 실습 코드 Github 주소
  - https://github.com/illhyhl1111/SNU_ML2019

- Colab 링크
  - https://colab.research.google.com/github/illhyhl1111/SNU_ML2019/blob/master/Lab1_1.ipynb
  - https://colab.research.google.com/github/illhyhl1111/SNU_ML2019/blob/master/Lab1_2.ipynb
  - https://colab.research.google.com/github/illhyhl1111/SNU_ML2019/blob/master/Lab1_3.ipynb

References

- https://zzsza.github.io/data/2018/08/30/google-colab/
- https://stackoverflow.com/questions/55050988/can-i-run-a-google-colab-free-edition-script-and-then-shutdown-my-computer