Conference Call

I팀 4조 Super SAI

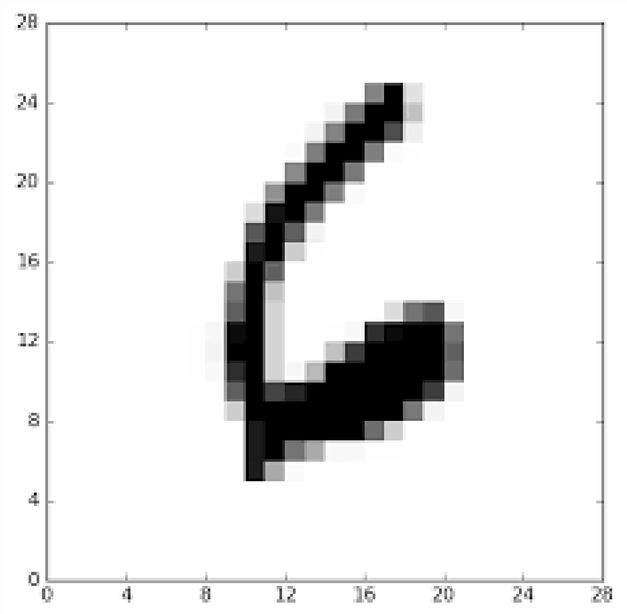
TABLE OF CONTENTS

What is MNIST?

우편번호를 인식하기 위한 숫자 0-9 범위의 손글씨 data set.

train_set 60000, test_set 10000개

총 7만개의 흑백 이미지 데이터로 구성되어 있다.



Detail

- 컴퓨터 비전 데이터 셋
- 28 x 28 size image
- 1 channel gray image
- 0 - 9 digits

Torchvision Package

Torchvision 패키지는 여러가지 유용한 데이터 셋들과 모델 등 컴퓨터 비전을 학습하기 위한 다양한 도구들이 포함되어 있습니다.

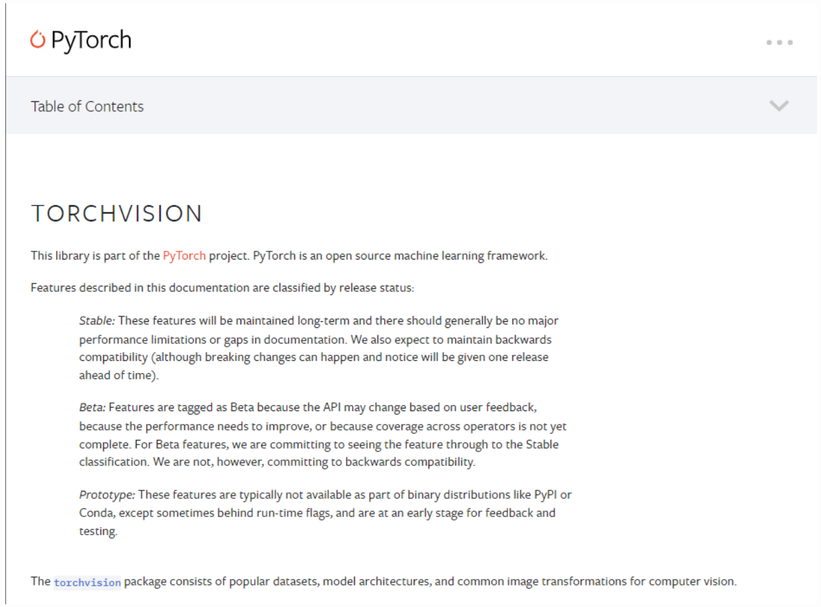

PyTorch

Table of Contents

TORCHVISION

This library is part of the PyTorch project. PyTorch is an open source machine learning framework.

Features described in this documentation are classified by release status:

Stable: These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).

Beta: Features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification. We are not, however, committing to backwards compatibility.

Prototype: These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.

The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.

torchvision.datasets

- MNIST, Fashion-MNIST, EMNIST, ImageFolder, DatasetFolder

torchvision.models

- Alexnet, VGG, ResNet, SqueezeNet

torchvision.transforms

- Transforms on PIL Image, Transforms on torch.*Tensor, Conversion Transforms, Generic Transforms, Functional Transforms

torchvision.utils

DataLoader

root

root는 train data와 test data가 저장되는 경로를 의미합니다.

train

train은 데이터셋이 training 용도인지 test 용도인지 지정하는 역할을 합니다.

Download

download = True라면, 인터넷에서 데이터를 다운로드 합니다.

Transform

transform과 target_transform은 feature와 label의 변형 방식을 지정합니다.

```
import torch
from torch.utils.data import Dataset
from torchvision import datasets
from torchvision.transforms import ToTensor
import matplotlib.pyplot as plt


training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor()
)

test_data = datasets.FashionMNIST(
    root="data",
    train=False,
    download=True,
    transform=ToTensor()
)
```

FashionMNIST 데이터를 통한 예시 코드

Torch.utils.DataLoader

DataLoader

로드할 데이터를 지정합니다.

batch_size

데이터를 몇 개씩 잘라서 학습할지 결정합니다.

shuffle

batch_size의 이미지를 불러올 때, 무작위로 불러올 것 인지 결정합니다.

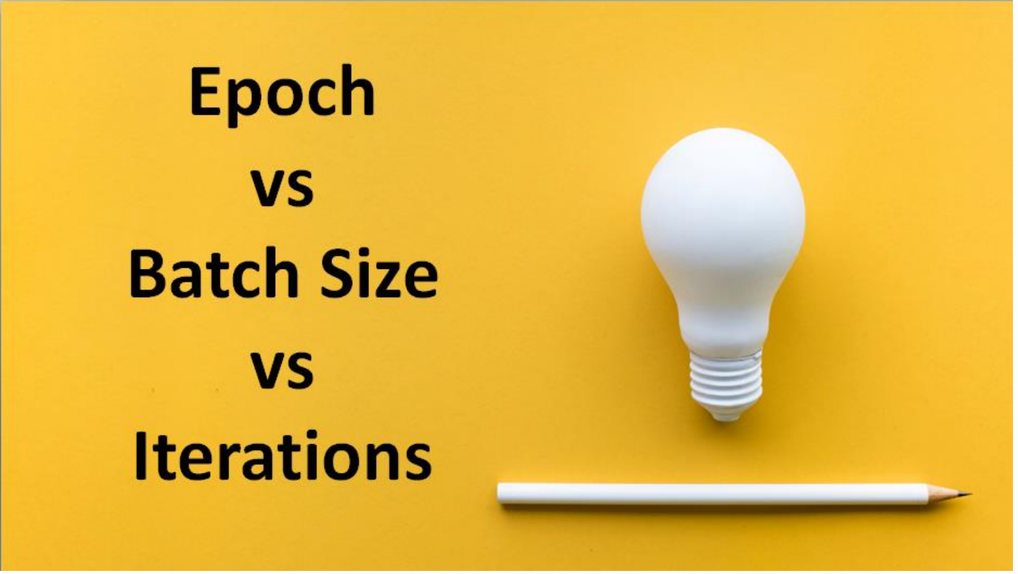
drop_last

batch_size만큼 잘라서 가져올 때, 남는 데이터를 버릴 것인지 결정합니다.

```
# dataset loader
data_loader = torch.utils.data.DataLoader(dataset=mnist_train,
                                          batch_size=batch_size,
                                          shuffle=True,
                                          drop_last=True)
```

```
# MNIST dataset
mnist_train = dsets.MNIST(root='MNIST_data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='MNIST_data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)
```



MNIST의 경우 train_data가 60000이므로, batch_size = 100이라면
batch는 600개가 됩니다.

Epoch

train_set 전체의 학습을 완료한 횟수입니다.

Batch_size

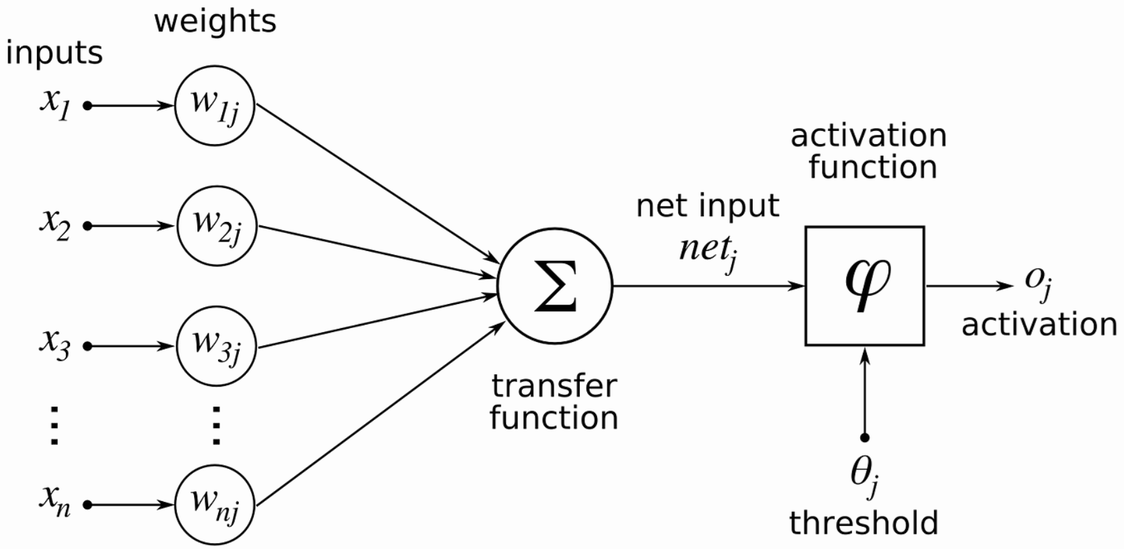
가능하다면 train_set을 한 번에 사용하는게 좋지만, 메모리에 한계가 있으므로 나눠서 학습합니다.

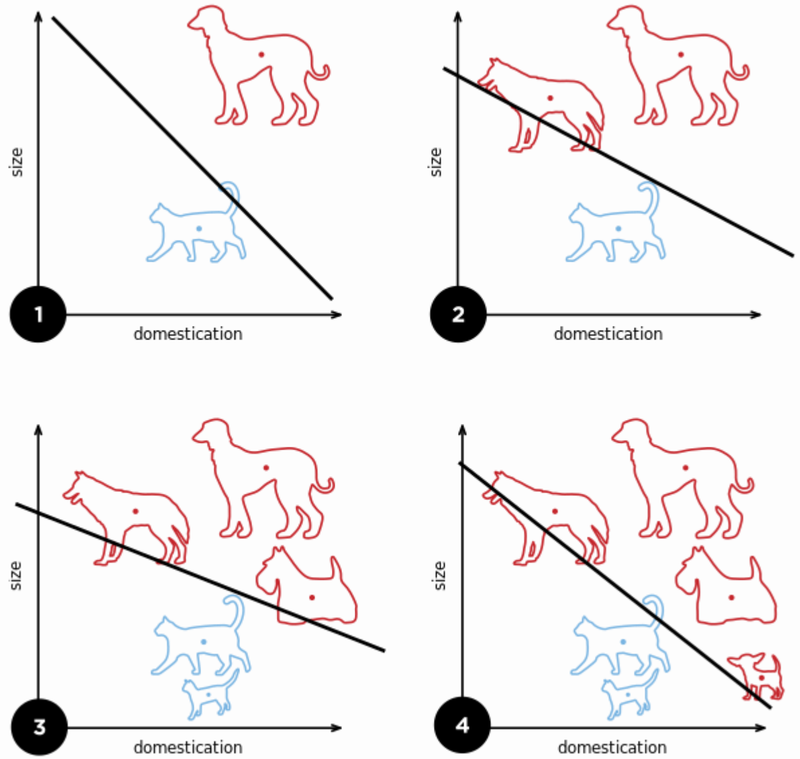
Iteration

batch를 학습에 사용한 횟수를 의미합니다.

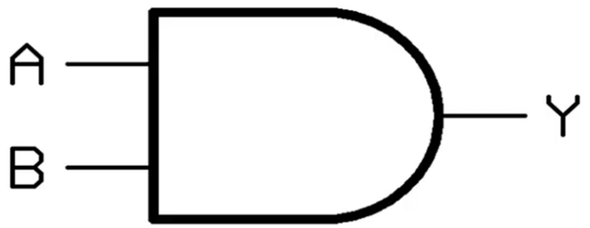
Perceptron

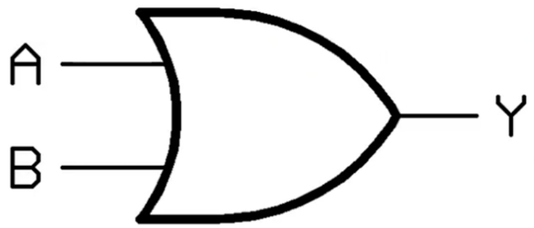
Perceptron이란 1개의 linear layer를 사용한 신경망으로,
각각의 입력값에 특정 가중치를 부여해 데이터를 학습시킵니다.





AND, OR

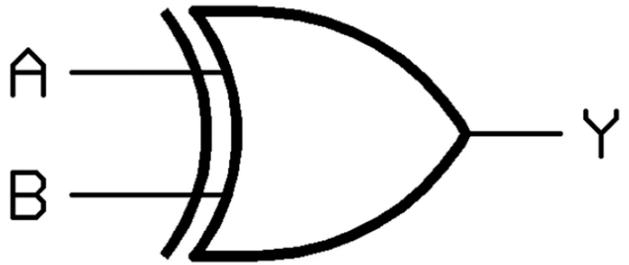




A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

XOR

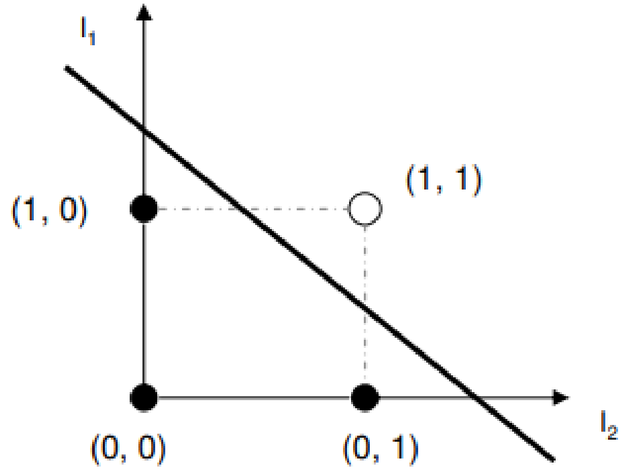


A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

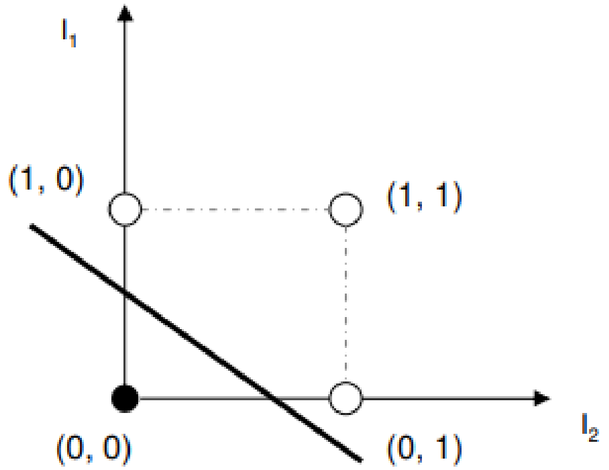
SLP(Single Layer Perceptron)의 한계:
Linear Binary Classifier는 AND, OR 연산과 같은 데이터의 경우 분류가 가능하나 XOR 연산과 같은 데이터는 분류하지 못합니다.

MLP(Multi Layer Perceptron)

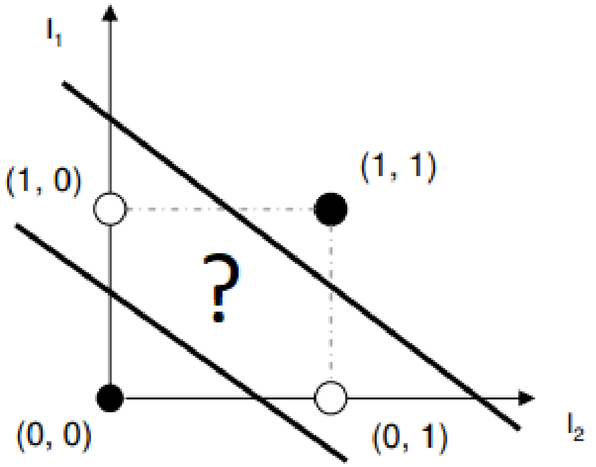
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

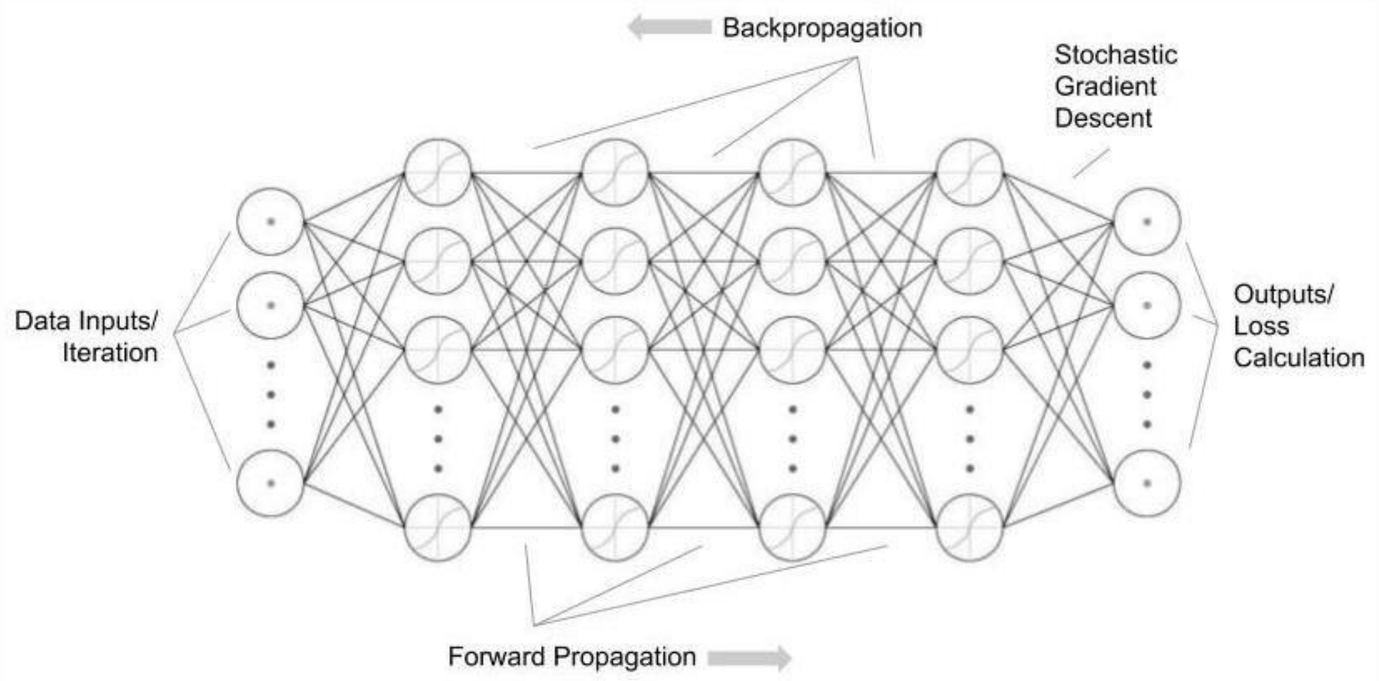


SLP(Single Layer Perceptron)의 한계:

Linear Binary Classifier는 AND, OR 연산과 같은 데이터의 경우 분류가 가능하나 XOR 연산과 같은 데이터는 분류하지 못합니다.

MLP(Multi Layer Perceptron)

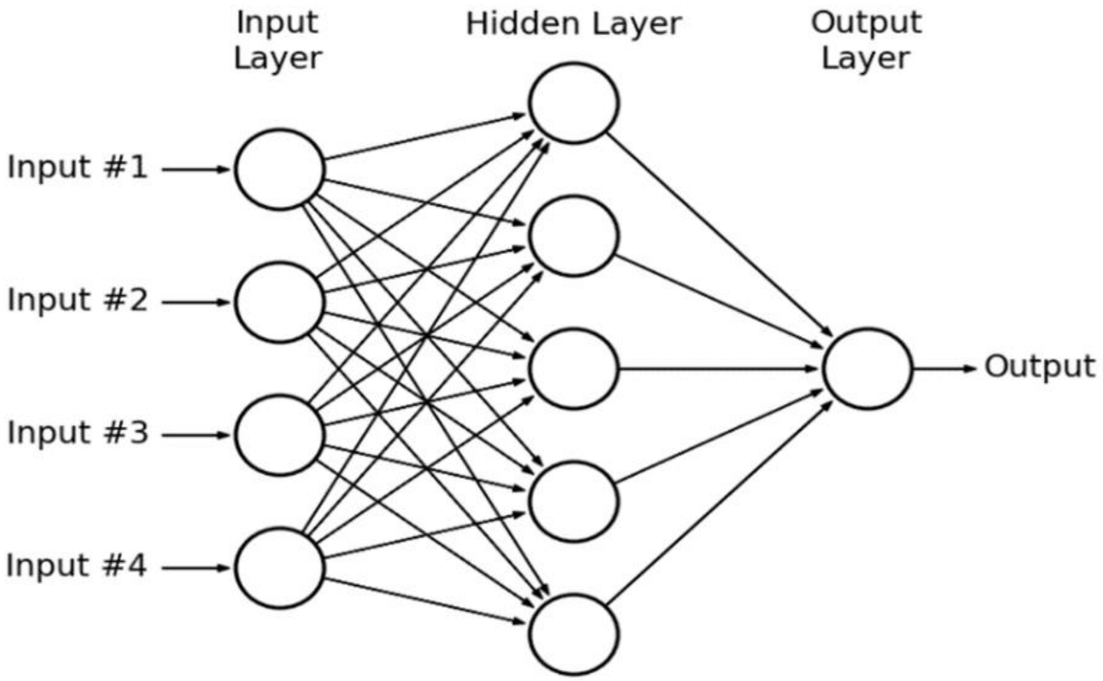
Multi-layer Perceptron



단일 퍼셉트론으로는 XOR 문제를 해결할 수 없습니다.

역전파(Backpropagation)를 이용하여 loss를 최소화 할 수 있도록 weight를 업데이트 하는 방식을 사용하면, 다층 퍼셉트론을 이용한 학습이 가능합니다.

Backpropagation



활성 함수가 내놓는 결과값이 실제값과 오류가
최소가 되도록 입력층에서 전달되는 값들에
곱해지는 가중치의 값을 결정합니다.

single layer perceptron은 활성 함수가 1개인 반면,
Multi layer Perceptron은 은닉층과 출력층에 존재하는
활성 함수와 가중치가 여러개입니다.

Backpropagation : code review

```
import torch.optim as optim

# Optimizer를 생성합니다.
optimizer = optim.SGD(net.parameters(), lr=0.01)

# 학습 과정(training loop)은 다음과 같습니다:
optimizer.zero_grad()   # 변화도 버퍼를 0으로
output = net(input)
loss = criterion(output, target)
loss.backward()
optimizer.step()    # 업데이트 진행
```

Pytorch 공식 가이드

loss.backward() 함수를 통해 역전파를
이용한 가중치 업데이트를 진행합니다.

FIN.