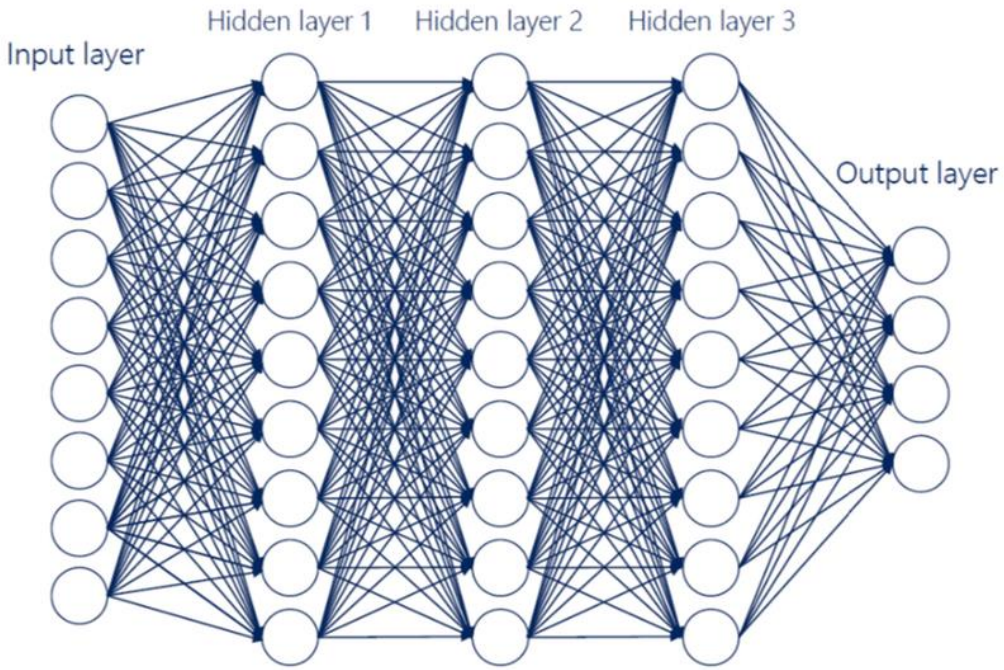
인공지능의 기초

5주차

Today

- Mnist classification using DNN
  - Optimizer
  - Training



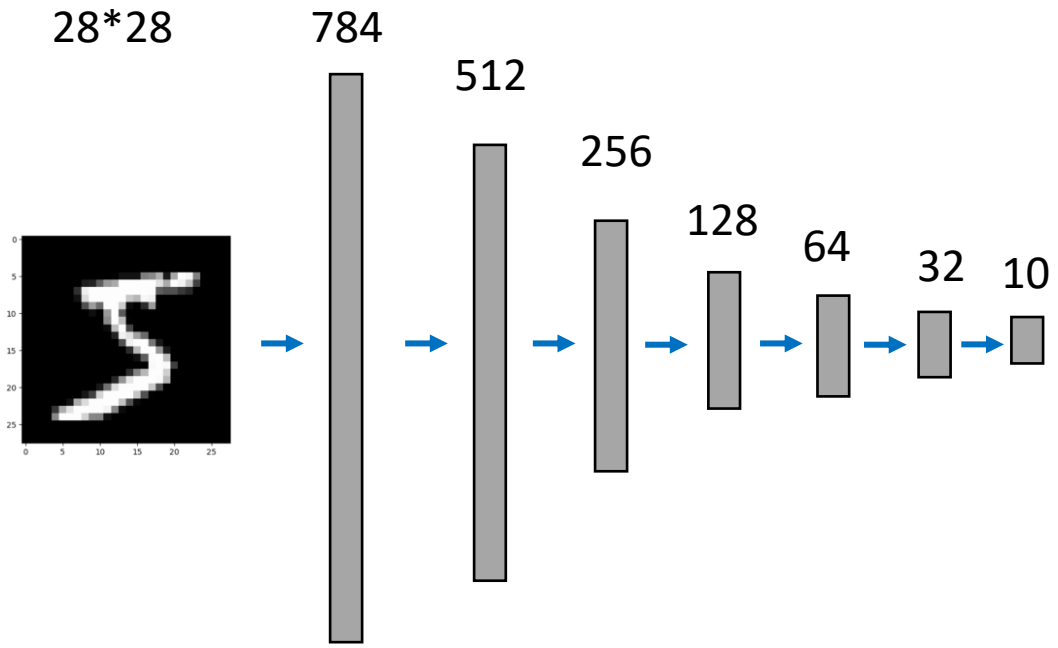
MNIST

- 0~255 흑백 이미지
- Size: 28*28
- 6만 train image
- 1만 test image

실습

✓ 5초

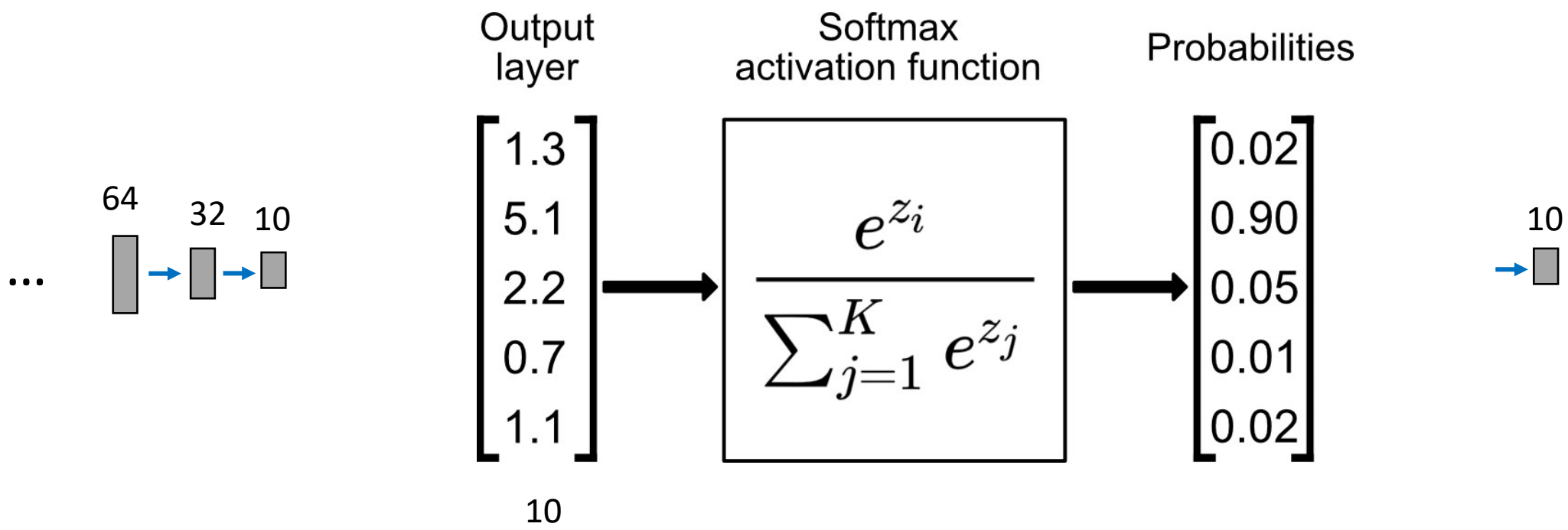
```
[1]  #Importing Library

     import torch
     import torch.nn as nn
     import torch.nn.functional as F
     import torch.optim as optim
     from torchvision import datasets, transforms
```




```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(784, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 128)
        self.fc4 = nn.Linear(128, 64)
        self.fc5 = nn.Linear(64, 32)
        self.fc6 = nn.Linear(32, 10)

    def forward(self, x):
        x = x.float()
        h1 = F.relu(self.fc1(x.view(-1, 784)))
        h2 = F.relu(self.fc2(h1))
        h3 = F.relu(self.fc3(h2))
        h4 = F.relu(self.fc4(h3))
        h5 = F.relu(self.fc5(h4))
        h6 = self.fc6(h5)
        return F.log_softmax(h6, dim=1)
```

Softmax

- Classification task 마지막 layer에 자주 쓰인다.
- 각 숫자에 대한 확률



실습

✓ 5초

```
[4]  device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

     model = Net().to(device)
     print(model)
```

✓ 2초

```
[5]  transform = transforms.Compose([
                    transforms.ToTensor(),
                    transforms.Normalize((0.1307,), (0.3081,))])

     train_loader = torch.utils.data.DataLoader(
       datasets.MNIST('../data', train=True, download=True,
                    transform=transform),
         batch_size = 64, shuffle=True, )

     test_loader = torch.utils.data.DataLoader(
            datasets.MNIST('../data', train=False, download=True,
                    transform=transform),
         batch_size=64, shuffle=True, )
```

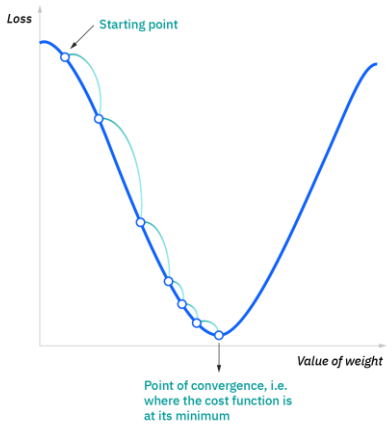
실습

✓ 0초

```
[6]  epochs = 10
     lr = 0.01
     momentum = 0.5
     no_cuda = True
     seed = 1
     log_interval = 200

     torch.manual_seed(seed)
     optimizer = optim.SGD(model.parameters(), lr=lr, momentum=momentum)
```

실습

- 1 epoch: 모든 데이터셋을 한 번 학습
- Optimizer(SGD, Momentum)



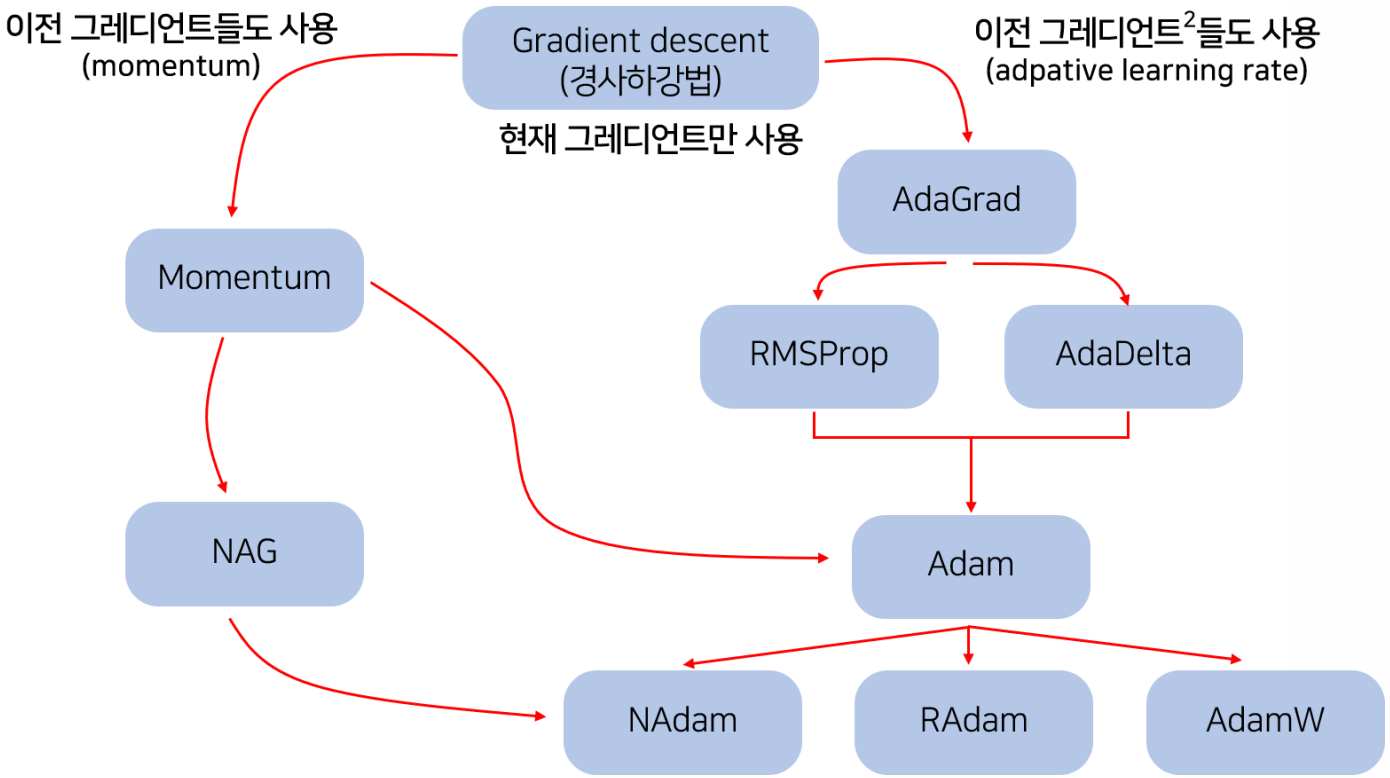
$$w = w - \alpha \frac{\partial loss}{\partial w}$$

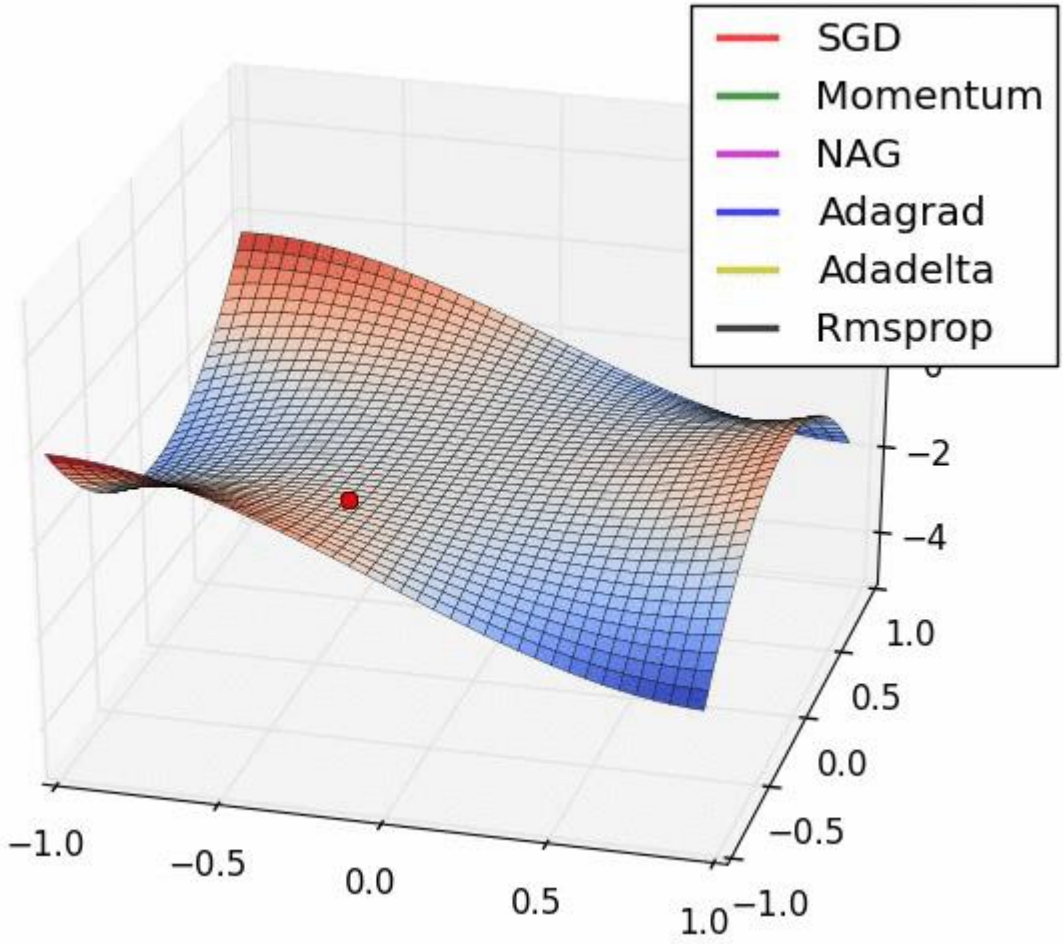
$$\mathrm{W(t+1)} = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

$$\mathrm{V(t)} = m * V(t-1) - \alpha \frac{\partial}{\partial w} Cost(w)$$

$$\mathrm{W(t+1)} = W(t) + V(t)$$

Optimizer


이전 그레디언트들도 사용
(momentum)
Gradient descent
(경사하강법)
현재 그레디언트만 사용
이전 그레디언트^{2}들도 사용
(adpative learning rate)
AdaGrad
Momentum
RMSProp
AdaDelta
NAG
Adam
NAdam
RAdam
AdamW


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
−1.0
−0.5
0.0
0.5
1.0
−2
−4
1.0
0.5
0.0
−0.5
−1.0

실습

```
def train(log_interval, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]₩tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
```

실습

0초

```
def test(log_interval, model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)

    print('₩nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)₩n'.format
          (test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
```

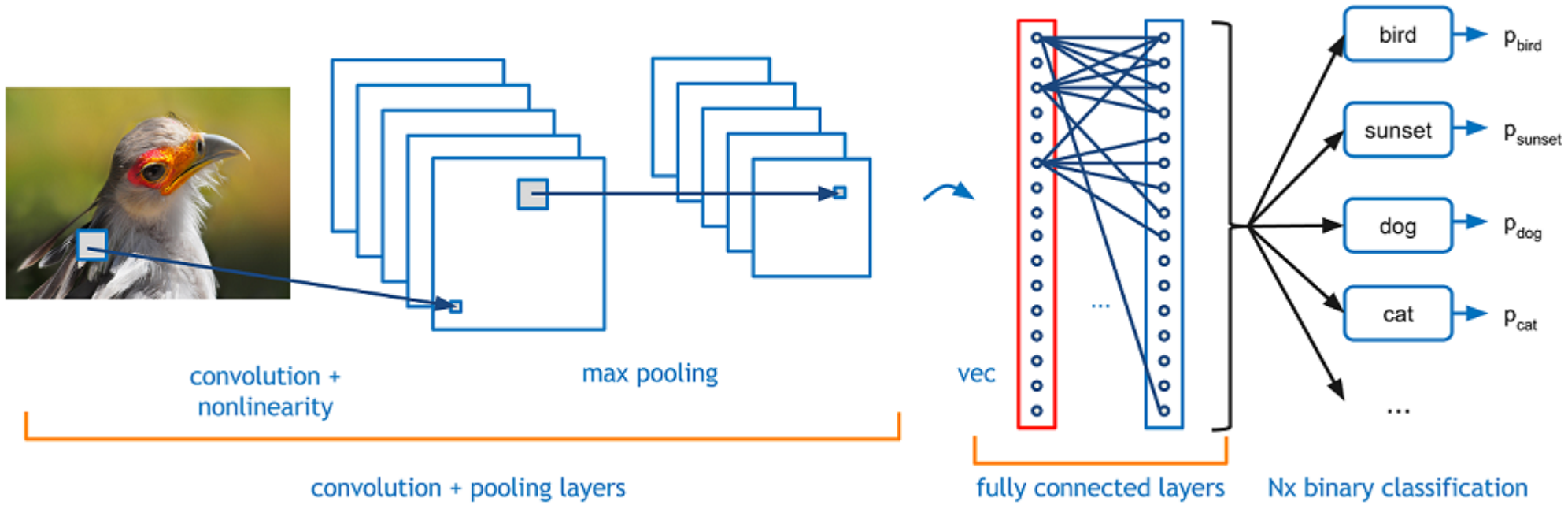
✓ 2분

```
# Train and Test the model and save it.
for epoch in range(1, 11):
    train(log_interval, model, device, train_loader, optimizer, epoch)
    test(log_interval, model, device, test_loader)
torch.save(model, './model.pt')
```

```
import matplotlib.pyplot as plt
import random

with torch.no_grad(): # torch.no_grad()를 하면 gradient 계산을 수행하지 않는다.
    X_test = mnist_test.test_data.view(-1, 28 * 28).float().to(device)
    Y_test = mnist_test.test_labels.to(device)
    for _ in range(5):
      r = random.randint(0, len(mnist_test) - 1)
      X_single_data = mnist_test.test_data[r:r + 1].view(-1, 28 * 28).float().to(device)
      Y_single_data = mnist_test.test_labels[r:r + 1].to(device)

      print('Label: ', Y_single_data.item())
      single_prediction = model(X_single_data)
      print('Prediction: ', torch.argmax(single_prediction, 1).item())

      plt.imshow(mnist_test.test_data[r:r + 1].view(28, 28), cmap='Greys', interpolation='nearest')
      plt.show()
```

예고

- CNN


convolution +
nonlinearity
max pooling
vec
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

감사합니다

출처

- https://xangmin.tistory.com/129
- https://twinw.tistory.com/247