인공지능 개론

L10.1 MNIST with CNN

국민대학교

소프트웨어융합대학원

박하명

GPU 사용하기 (복습)

- pytorch에서 CUDA 사용 설정

```
import torch
USE_CUDA = torch.cuda.is_available()
DEVICE = torch.device("cuda" if USE_CUDA else "cpu")
```

MNIST 데이터 불러오기 (복습)

```
import torchvision.datasets as dsets
import torchvision.transforms as transforms

mnist_train = dsets.MNIST(root='MNIST_data/', # 데이터 위치
                          train=True, # True: 훈련 데이터, False: 테스트 데이터
                          transform=transforms.ToTensor(),  # pytorch Tensor로 변환
                          download=True # 데이터가 없을 경우 다운로드
                         )

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

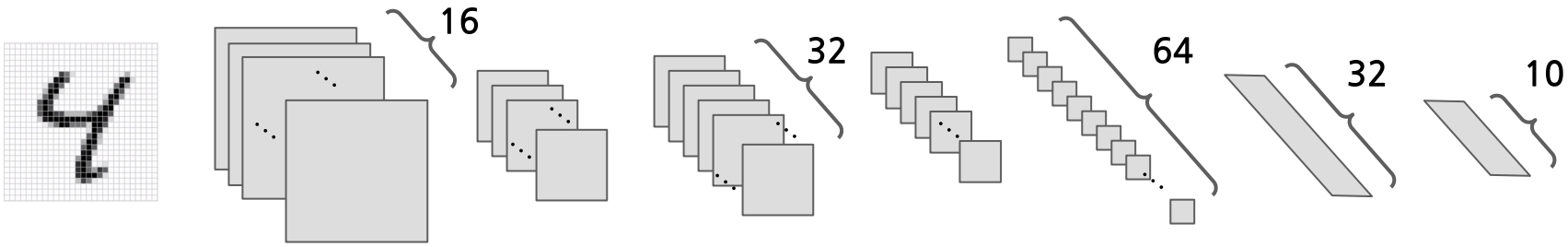
MNIST 데이터 불러오기 (복습)

- 불러온 MNIST 학습 및 테스트 데이터를 학습이 가능한 형태로 변환
- Covolutional Layer가 1+3차원 Tensor를 입력으로 받기 때문에, 아래와 같이 view를 설정 (1 x 28 x 28)

```
x_train = mnist_train.data.view(-1,1,28,28).float().to(DEVICE)
y_train = mnist_train.targets.to(DEVICE)
x_test = mnist_test.data.view(-1,1,28,28).float().to(DEVICE)
y_test = mnist_test.targets.to(DEVICE)
```

Our CNN Model

conv → pool → conv → pool → conv → fc → fc



Import modules

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

Model 생성

- Layer 생성

```
conv1 = nn.Conv2d(1, 16, 5, padding=2)
conv2 = nn.Conv2d(16, 32, 5, padding=1)
conv3 = nn.Conv2d(32, 64, 3)
maxpool=nn.MaxPool2d(2)
drop2d = nn.Dropout2d(0.2)

fc1 = nn.Linear(64*2*2, 32)
fc2 = nn.Linear(32, 10)
relu=nn.ReLU()
drop = nn.Dropout(0.2)
```

Model 생성

- Layer 합치기

```
conv_layer1 = nn.Sequential(conv1, maxpool, relu, drop2d)
conv_layer2 = nn.Sequential(conv2, maxpool, relu, drop2d)
conv_layer3 = nn.Sequential(conv3, maxpool, relu, drop2d)

conv_layer = nn.Sequential(conv_layer1, conv_layer2, conv_layer3).to(DEVICE)
fc_layer = nn.Sequential(fc1, relu, drop, fc2).to(DEVICE)
```

가중치 초기화

- Xavier 초기화 사용

```
nn.init.xavier_normal_(fc1.weight, 0.2)
nn.init.xavier_normal_(fc2.weight, 0.2)
nn.init.xavier_normal_(conv1.weight, 0.5)
nn.init.xavier_normal_(conv2.weight, 0.5)
nn.init.xavier_normal_(conv3.weight, 0.5)
```

Optimizer 설정

- conv_layer와 fc_layer의 parameter들을 학습해야한다고 optimizer에게 알려줌

```
params = list(conv_layer.parameters()) + list(fc_layer.parameters())

optimizer = optim.Adam(params, lr=0.004)
```

학습하기

```
for epoch in range(301):
 conv_layer.train() # 학습 모드로 전환 (dropout 켜기)
 fc_layer.train() # 학습 모드로 전환 (dropout 켜기)

 # FC 레이어의 입력으로 변경하기 위해 conv_layer의 결과를 view를 이용하여 쭉 핀다
 z = fc_layer(conv_layer(x_train).view(-1,64*2*2))
 cost = F.cross_entropy(z, y_train)

 optimizer.zero_grad()
 cost.backward()
 optimizer.step()

 if epoch % 10 == 0:
   conv_layer.eval() # 평가 모드로 전환 (dropout 끄기 = dropout 비율 0으로 설정)
   fc_layer.eval() # 평가 모드로 전환 (dropout 끄기 = dropout 비율 0으로 설정)
   with torch.no_grad():
     z_test = z = fc_layer(conv_layer(x_test).view(-1,64*2*2))
     cost_test = F.cross_entropy(z_test, y_test)
     accuracy = 100*(y_test == z_test.argmax(1)).sum().item()/len(z_test)
     print("epoch: {}, cost: {:.6f}, test: {:.6f}, acc: {:.3f}".format(epoch, cost, cost_test, accuracy))
```

epoch: 200, cost: 0.071278, test: 0.025799, **acc: 99.240**

Question?