

Artificial Intelligence Lab Work (3)
レポート解答用紙 (Report Answer Sheet)

学生証番号 (Student ID): 18521071

名前(Name): ズオン・ミン・ルオン (Duong Minh Luong)

問題 1.

(プログラム)

##ライブラリの読み込み

```
import torch
import matplotlib.pyplot as plt
import numpy as np
import torch.nn.functional as F
import torchvision as tv
```

#訓練データとテストデータの読み込み (初めて実行するときはデータをネットからダウンロードする)

```
train_dataset=tv.datasets.MNIST(root=".",train=True,transform=tv.transforms.ToTensor(),download=True)
test_dataset=tv.datasets.MNIST(root=".",train=False,transform=tv.transforms.ToTensor(),download=True)
```

#訓練データとテストデータのミニバッチ処理・ミニバッチサイズ=100・データの順番をシャッフル

```
train_loader=torch.utils.data.DataLoader(dataset=train_dataset,batch_size=100,shuffle=True)
test_loader=torch.utils.data.DataLoader(dataset=test_dataset,batch_size=100,shuffle=False)
```

```
for i in range(10):
    print(train_dataset[i])
    plt.imshow(train_dataset[i][0][0],cmap='gray')
    txt="label: "+str(train_dataset[i][1])
    plt.text(2,2,txt,color="white")
    plt.show()
```

#l1 と l2 は線形変換の関数

```
l1=torch.nn.Linear(784,300)
l2=torch.nn.Linear(300,10)
params=list(l1.parameters())+list(l2.parameters())
```

```

optimizer=torch.optim.Adam(params)
#Adam というパラメータ最適化手法を使う。パラメータ更新に関わるパラメータ群 params を渡す
def mynet(x):
    h=F.relu(l1(x))
    y=l2(h)
    return y


def train():
    #データ全体を 回学習する
    for e in range(10):
        loss=0
        for images,labels in train_loader:
            images=images.view(-1,28*28) # (100×1×28×28) から (100×784) に変形
            optimizer.zero_grad() #勾配を初期化
            y=mynet(images) #ネットワークの計算 (ラベルの予測)
            batchloss=F.cross_entropy(y,labels) #正解ラベルに対する出力の損失
            batchloss.backward() #誤差に対する勾配を計算
            optimizer.step() #パラメータ更新
            loss=loss+batchloss.item() #ミニバッチでの損失を loss に足す
        print("epoch: ",e,"loss: ",loss)


def test():
    correct=0
    total=len(test_loader.dataset)
    for images,labels in test_loader:
        images=images.view(-1,28*28)
        y=mynet(images)
        pred_labels=y.max(dim=1)[1] #ラベルの予測 (最大値となるラベル)
        correct=correct+(pred_labels==labels).sum()
    #100 個のうち正解数がいくつか数える
    print("correct: ",correct.item())
    print("total: ",total)
    print("accuracy: ",correct.item()/total)

train()
test()

```

(実行結果)

 `train()`

 epoch: 0 loss: 194.9045080728829
epoch: 1 loss: 81.96370410174131
epoch: 2 loss: 55.60747605469078
epoch: 3 loss: 41.164965299889445
epoch: 4 loss: 31.122589415870607
epoch: 5 loss: 24.04020838229917
epoch: 6 loss: 19.233246938325465
epoch: 7 loss: 14.811302168061957
epoch: 8 loss: 11.480905142612755
epoch: 9 loss: 9.595466721686535

`[36] test()`

correct: 9807
total: 10000
accuracy: 0.9807

問題 2

```
(プログラム)
##ライブラリの読み込み
import torch
import matplotlib.pyplot as plt
import numpy as np
import torch.nn.functional as F
import torchvision as tv

#訓練データとテストデータの読み込み(初めて実行するときはデータをネットからダウンロードする)
train_dataset=tv.datasets.MNIST(root="./",train=True,transform=tv.transforms.ToTensor(),download=True)
test_dataset=tv.datasets.MNIST(root="./",train=False,transform=tv.transforms.ToTensor(),download=True)

#訓練データとテストデータのミニバッチ処理・ミニバッチサイズ=100・データの順番をシャッフル
train_loader=torch.utils.data.DataLoader(dataset=train_dataset,batch_size=100,shuffle=True)
test_loader=torch.utils.data.DataLoader(dataset=test_dataset,batch_size=100,shuffle=False)

for i in range(10):
    print(train_dataset[i])
    plt.imshow(train_dataset[i][0][0],cmap='gray')
    txt="label: "+str(train_dataset[i][1])
    plt.text(2,2,txt,color="white")
    plt.show()

#l1 と l2 は線形変換の関数
l1=torch.nn.Linear(784,800)# 300 to 800
l2=torch.nn.Linear(800,10) # 300 to 800
params=list(l1.parameters())+list(l2.parameters())
optimizer=torch.optim.Adam(params)
#Adam というパラメータ最適化手法を使う。パラメータ更新に関わるパラメータ群 params を渡す
def mynet(x):
    h=F.relu(l1(x))
    y=l2(h)
    return y
```

```


def train():
    #データ全体を 回学習する
    for e in range(10):
        loss=0
        for images,labels in train_loader:
            images=images.view(-1,28*28) #(100×1x28×28)から(100×784)に変形
            optimizer.zero_grad() #勾配を初期化
            y=mynet(images) #ネットワークの計算(ラベルの予測)
            batchloss=F.cross_entropy(y,labels) #正解ラベルに対する出力の損失
            batchloss.backward() #誤差に対する勾配を計算
            optimizer.step() #パラメータ更新
            loss=loss+batchloss.item() #ミニバッチでの損失を loss に足す
        print("epoch: ",e,"loss: ",loss)

def test():
    correct=0
    total=len(test_loader.dataset)
    for images,labels in test_loader:
        images=images.view(-1,28*28)
        y=mynet(images)
        pred_labels=y.max(dim=1)[1] #ラベルの予測(最大値となるラベル)
        correct+= (pred_labels==labels).sum()
    #100 個のうち正解数がいくつか数える
    print("correct: ",correct.item())
    print("total: ",total)
    print("accuracy: ",correct.item()/total)

train()
test()

```

(実行結果)

 train()

```
epoch: 0 loss: 161.23681027814746
epoch: 1 loss: 61.19480243511498
epoch: 2 loss: 39.819305231329054
epoch: 3 loss: 27.3653686568141
epoch: 4 loss: 19.504109292756766
epoch: 5 loss: 14.60631992132403
epoch: 6 loss: 10.914668398618232
epoch: 7 loss: 8.732307245314587
epoch: 8 loss: 7.560406287724618
epoch: 9 loss: 5.102648169005988
```

[16] test()

```
correct: 9817
total: 10000
accuracy: 0.9817
```