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) #(100×1x28×28)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=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) |

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=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) |