Homework Assignment 2

Chaoji Zuo chaoji.zuo@rutgers.edu
CZ296 / 190003416

Problem (a): Use only Numpy to train and test this network. You are NOT allowed to use deep learning framework (e.g. Pytorch, Tensorflow etc.) and the corresponding autograd or nn module to train the network.
I tried to solve this problem but failed. My training always end up with all the classes having the same probability.

Problem (b): Use deep learning framework to train and test this network. You are allowed to use the corresponding autograd or nn module to train the network.
For this problem, I tried two ways to solve, the first is not use the 'nn.module' and pytorch optimizer, just use the autograd tensor to train, which have a slow training process and not very great performance.
The second method is using the nn.module, which get a pretty great result. I set the learning rate as 0.3 and use 128 as the batch size. Here are the screen snapshot of my program.

```
training loss of iteration 1:0.502437869392669
training time of iteration 1:0.5606019496917725

training loss of iteration 2:0.18621173971243252
training time of iteration 2:0.6114969253540039

training loss of iteration 3:0.13117332770816806
training time of iteration 3:0.5872447490692139

training loss of iteration 4:0.10245601998677831
training time of iteration 4:0.5552151203155518

training loss of iteration 5:0.08396222856122135
training time of iteration 5:0.5444231033325195

training loss of iteration 6:0.0707190956246219
training time of iteration 6:0.5477256774902344

training loss of iteration 7:0.06057672608476966
training time of iteration 7:0.5398659706115723

training loss of iteration 8:0.052427819350670805
training time of iteration 8:0.540463924407959

training loss of iteration 9:0.04575944484284978
training time of iteration 9:0.5716700553894043

training loss of iteration 10:0.04016362618072145
training time of iteration 10:0.5675041675567627

Finished Training
```

Figure 1: Training loss and time

```
test_XX=torch.tensor(test_X,dtype=torch.float32)
```

```
with torch.no_grad():
    res = net(test_XX)
```

```
print("precision on test:",np.count_nonzero(~(np.array(res.argmax(axis=1)) - test_Y))/float(len(test_Y)))
```

```
precision on test: 0.9987
```

Figure 2: Accuracy on test set

Here are the whole code of my program:

```
import torch.nn as nn
import torch.nn.functional as F
import time
```

```
4
5  class Net(nn.Module):
6      def __init__(self):
7          super(Net,self).__init__()
8          self.fc1 = nn.Linear(784,200)
9          self.fc2 = nn.Linear(200,50)
10         self.fc3 = nn.Linear(50,10)
11         #self.softmax = nn.Softmax(dim=1)
12
13     def forward(self,x):
14         x = F.relu(self.fc1(x))
15         x = F.relu(self.fc2(x))
16         #x = self.softmax(self.fc3(x))
17         return x
18
19 net=Net()
20 a=list(zip(train_XX,train_YY))
21 import torch.optim as optim
22 criterion = nn.CrossEntropyLoss()
23 optimizer = optim.SGD(net.parameters(),lr=0.3)
24 trainloader=torch.utils.data.DataLoader(a,batch_size=128)
25
26 for epoch in range(10):
27     running_loss = 0.0
28     batch_size=128
29     index_list=[x for x in range(60000) if x%batch_size==0] # mini-batch SGD
30     loss_list=[]
31     start_time=time.time()
32     #initialize weights
33     for start_index in index_list[:]:
34         batch_X=train_XX[start_index:start_index+batch_size]
35         batch_y=train_YY[start_index:start_index+batch_size]
36         inputs,labels=batch_X,batch_y
37         #print(inputs.dtype)
38
39         optimizer.zero_grad()
40
41         outputs = net(inputs)
42         loss = criterion(outputs,labels)
43         loss.backward()
44         optimizer.step()
45         running_loss = loss.item()
46         loss_list.append(running_loss)
47     end_time=time.time()
48     print("training loss of iteration {}:{}".format(epoch+1,np.mean(loss_list)
       ))
49     print("training time of iteration {}:{}".format(epoch+1,end_time-
       start_time))
50     print()
51 print('Finished Training')
```