Lab_2

March 5, 2023

[ ]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

# Device configuration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Load the dataset
data = np.load('lab2_dataset.npz')
train_feats = torch.tensor(data['train_feats']) # 44730 examples, 11 frames/
 ↪example, 40 mfcc bins
test_feats = torch.tensor(data['test_feats']) # 4773 examples, 11 frames/
 ↪example, 40 mfcc bins
train_labels = torch.tensor(data['train_labels']) # 44730 training data labels
test_labels = torch.tensor(data['test_labels']) # 4773 testing data labels
phone_labels = data['phone_labels'] # 48 labels for the 48 classifications

# Set up the dataloaders
train_dataset = torch.utils.data.TensorDataset(train_feats, train_labels)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=64,␣
 ↪shuffle=True)

test_dataset = torch.utils.data.TensorDataset(test_feats, test_labels)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=64,␣
 ↪shuffle=False)

# Define the model architecture
class MyModel(nn.Module):
    def __init__(self):
        super(MyModel, self).__init__()
        # TODO: Fill in the model's layers here
        vals = [440, 300, 200, 150, 100, 48]
        self.lin1 = nn.Linear(vals[0], vals[1])
        self.lin2 = nn.Linear(vals[1], vals[2])
        self.lin3 = nn.Linear(vals[2], vals[3])
```

```
        self.lin4 = nn.Linear(vals[3], vals[4])
        self.lin5 = nn.Linear(vals[4], vals[5])
        self.relu = nn.ReLU()


    def forward(self, x):
        # TODO: Fill in the forward pass here
        x = torch.reshape(x, (-1, 11*40))
        x = self.relu(self.lin1(x))
        x = self.relu(self.lin2(x))
        x = self.relu(self.lin3(x))
        x = self.relu(self.lin4(x))
        x = self.lin5(x)
        return x

# Instantiate the model, loss function, and optimizer
model = MyModel()
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

def train_network(model, train_loader, criterion, optimizer):
    # TODO: fill in
    numEpochs = 12
    for epoch in range(numEpochs):
        print('Epoch', epoch, 'running...')
        for i, (inputs, labels) in enumerate(train_loader, 0):
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

def test_network(model, test_loader):
    correct = 0
    total = 0
    correctFreqs = {i:0  for i in range(48)}
    totalFreqs = {i:0 for i in range(48)}
    shMistakes = {i:0 for i in range(48)}
    pMistakes = {i:0 for i in range(48)}
    mMistakes = {i:0 for i in range(48)}
    rMistakes = {i:0 for i in range(48)}
    aeMistakes = {i:0 for i in range(48)}
    with torch.no_grad():
        for data in test_loader:
            inputs, labels = data
            outputs = model(inputs)
```

```
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            match = (predicted==labels)
            for i, label in enumerate(labels):
                if match[i]:
                    correctFreqs[label.item()]+=1
                else:
                    if label.item() == 14:
                        shMistakes[predicted[i].item()]+=1
                    if label.item() == 10:
                        pMistakes[predicted[i].item()]+=1
                    if label.item() == 39:
                        mMistakes[predicted[i].item()]+=1
                    if label.item() == 4:
                        rMistakes[predicted[i].item()]+=1
                    if label.item() == 35:
                        aeMistakes[predicted[i].item()]+=1
                totalFreqs[label.item()]+=1
            correct += match.sum().item()
    print('Test accuracy: %d %%' % (100 * correct / total))
    for i, phone in enumerate(phone_labels):
        print(i, 'Accuracy of', phone, 'is', str(int(100 * correctFreqs[i]/
 ↪totalFreqs[i]))+'%')
    print()
    print('most common mis-classification of \'sh\' is',␣
 ↪phone_labels[max(shMistakes, key=shMistakes.get)])
    print('most common mis-classification of \'p\' is',␣
 ↪phone_labels[max(pMistakes, key=pMistakes.get)])
    print('most common mis-classification of \'m\' is',␣
 ↪phone_labels[max(mMistakes, key=mMistakes.get)])
    print('most common mis-classification of \'r\' is',␣
 ↪phone_labels[max(rMistakes, key=rMistakes.get)])
    print('most common mis-classification of \'ae\' is',␣
 ↪phone_labels[max(aeMistakes, key=aeMistakes.get)])

train_network(model, train_loader, criterion, optimizer)
print()
test_network(model, test_loader)
```

```
Epoch 0 running…
Epoch 1 running…
Epoch 2 running…
Epoch 3 running…
Epoch 4 running…
Epoch 5 running…
Epoch 6 running…
Epoch 7 running…
```

```
Epoch 8 running…
Epoch 9 running…
Epoch 10 running…
Epoch 11 running…

Test accuracy: 59 %
0 Accuracy of sil is 85%
1 Accuracy of s is 53%
2 Accuracy of ao is 59%
3 Accuracy of l is 53%
4 Accuracy of r is 57%
5 Accuracy of iy is 63%
6 Accuracy of vcl is 64%
7 Accuracy of d is 59%
8 Accuracy of eh is 46%
9 Accuracy of cl is 77%
10 Accuracy of p is 62%
11 Accuracy of ix is 40%
12 Accuracy of z is 86%
13 Accuracy of ih is 26%
14 Accuracy of sh is 83%
15 Accuracy of n is 14%
16 Accuracy of v is 77%
17 Accuracy of aa is 63%
18 Accuracy of y is 78%
19 Accuracy of uw is 64%
20 Accuracy of w is 75%
21 Accuracy of ey is 69%
22 Accuracy of dx is 62%
23 Accuracy of b is 68%
24 Accuracy of ay is 72%
25 Accuracy of ng is 70%
26 Accuracy of k is 80%
27 Accuracy of epi is 79%
28 Accuracy of ch is 73%
29 Accuracy of dh is 56%
30 Accuracy of er is 55%
31 Accuracy of en is 47%
32 Accuracy of g is 68%
33 Accuracy of aw is 62%
34 Accuracy of hh is 64%
35 Accuracy of ae is 54%
36 Accuracy of ow is 48%
37 Accuracy of t is 53%
38 Accuracy of ax is 51%
39 Accuracy of m is 58%
40 Accuracy of zh is 39%
41 Accuracy of ah is 45%
```

```
42 Accuracy of el is 60%
43 Accuracy of f is 81%
44 Accuracy of jh is 58%
45 Accuracy of uh is 10%
46 Accuracy of oy is 48%
47 Accuracy of th is 49%

most common mis-classification of 'sh' is s
most common mis-classification of 'p' is k
most common mis-classification of 'm' is v
most common mis-classification of 'r' is er
most common mis-classification of 'ae' is eh
```