CS224 - Winter 2023 - PROGRAMMING ASSIGNMENT 3 - DNN

Due: March 22, 2023 @ 11:59pm PDT

Maximum points: 15

Enter your information below:

(full) Name: Shahriar M Sakib
Student ID Number: 862393922

By submitting this notebook, I assert that the work below is my own work, completed for this course. Except where explicitly cited, none of the portions of this notebook are duplicated from anyone else's work or my own previous work.

Academic Integrity

Each assignment should be done individually. You may discuss general approaches with other students in the class, and ask questions to the TA, but you must only submit work that is yours . If you receive help by any external sources (other than the TA and the instructor), you must properly credit those sources. The UCR Academic Integrity policies are available at http://conduct.ucr.edu/policies/academicintegrity.html.

Overview

In this assignment you will extract the Deep Convolutional Neural Network features of a dataset(Question 1), implement multinomial logistic regression(Question 2) and ROC curve(Question 3).

For this assignment we will use the functionality of PyTorch, Numpy, and Matplotlib.

If you are asked to **implement** a particular functionality, you should **not** use an existing implementation from the libraries above (or some other library that you may find). When in doubt, **please ask**.

Before you start, make sure you have installed all those packages in your local Jupyter instance.

Read **all** cells carefully and answer **all** parts (both text and missing code). You will complete all the code marked `TODO` and answer descriptive/derivation questions.

In [1]:
```
pip install torch torchvision torchaudio
```

```
Requirement already satisfied: torch in c:\users\shahr\anaconda3\lib\site-packages (2.0.0)
Requirement already satisfied: torchvision in c:\users\shahr\anaconda3\lib\site-packages (0.15.0)
Requirement already satisfied: torchaudio in c:\users\shahr\anaconda3\lib\site-packages (2.0.1)
Requirement already satisfied: filelock in c:\users\shahr\anaconda3\lib\site-packages (from torch) (3.6.0)
Requirement already satisfied: typing-extensions in c:\users\shahr\anaconda3\lib\site-packages (from torch) (4.3.0)
Requirement already satisfied: sympy in c:\users\shahr\anaconda3\lib\site-packages (from torch) (1.10.1)
Requirement already satisfied: networkx in c:\users\shahr\anaconda3\lib\site-packages (from torch) (2.8.4)
Requirement already satisfied: jinja2 in c:\users\shahr\anaconda3\lib\site-packages (from torch) (2.11.3)
Requirement already satisfied: numpy in c:\users\shahr\anaconda3\lib\site-packages (from torchvision) (1.21.5)
Requirement already satisfied: requests in c:\users\shahr\anaconda3\lib\site-packages (from torchvision) (2.28.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\shahr\anaconda3\lib\site-packages (from torchvision)
(9.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\shahr\anaconda3\lib\site-packages (from jinja2->torch) (2.
0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shahr\anaconda3\lib\site-packages (from requests->torchvision)
(3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\shahr\anaconda3\lib\site-packages (from requests->torc
hvision) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shahr\anaconda3\lib\site-packages (from requests->torchvi
sion) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\shahr\anaconda3\lib\site-packages (from requests->t
orchvision) (2.0.4)
Requirement already satisfied: mpmath>=0.19 in c:\users\shahr\anaconda3\lib\site-packages (from sympy->torch) (1.2.1)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
%matplotlib inline
import numpy as np
import torch.nn as nn
import torchvision.models as models
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torch.autograd import Variable
import scipy.io as sio
```

```
import torch
import matplotlib.pyplot as plt
```

DO **NOT** MODIFY ANYTHING IF NOT MENTIONED.

Question 1: DNN [6 points]

In this problem, you are required to extract the Deep Convolutional Neural Network (CNN) features for a dataset.

The dataset provided is the **MNIST** dataset.

In [2]:
```
mnist_trainset = datasets.MNIST(root='./data', train=True, download=True, transform=None)
mnist_testset = datasets.MNIST(root='./data', train=False, download=True, transform=None)
```

You need to extract features from these images using the **ResNet-50** architecture available in PyTorch.

You need to fill in the function named `extract`, which loads the images, extracts the features and appends them to the feature list along with the corresponding labels. The output of this code is the file `'mnist_train.mat` and `mnist_test.mat`, which are to be used in the next problem. This file should have

1. `features` of dimension $m \times n$, where $m = 60000$ is the number of images and $n = 2048$ is the feature dimension obtained using ResNet-50.
2. `labels` is a vector of length m containing labels from 0 to 9 for the 10 categories.

Some portions of the code is already filled in for convenience. Please do **not** modify anything if not mentioned.

In [3]:
```
def extract(dataset, filename):
    features = []
    labels = []

    transform_test = transforms.Compose([
        transforms.Grayscale(num_output_channels=3),
        transforms.ToTensor(),
        transforms.Normalize((0.5), (0.5))
    ])

    extractor = models.resnet50(weights='IMAGENET1K_V1')
```

```
    extractor.eval()

    for (_img, label) in dataset:
        # TODO: fill in to load image, preprocess, and extract features
        # the output variable F expected to be the feature of the image of dimension (2048,)

        img = transform_test(_img)
        img = img.unsqueeze(0)
        F = extractor(img).squeeze().detach().numpy()
        features.append(F)
        labels.append(label)

    sio.savemat(filename, mdict={'features': features, 'labels': labels})
```

Run the code below to get extracted features and labels of MNIST dataset, and then save it to `.mat` file. (This might take a while.)

You do **not** need to submit the `.mat` file along with the PDF file.

In [12]:
```
extract(mnist_trainset, 'mnist_train.mat')
extract(mnist_testset, 'mnist_test.mat')
```

Question 2: Multinomial Logistic Regression [6 points]

In this problem, you will implement the multinomial logistic regression using the extracted features and labels in Question 1.

You should use variables `trfeature` and `trlabel` for training and `tefeature` and `telabel` for testing.

Please remember to map the labels properly for testing. You need to fill in the function named `apply_gradient`, which returns the updated parameter θ after a single pass of gradient descent using the given data points and labels. You also need to fill up certain the portions as mentioned in function `mlr`.

- Using built-in functions like `sklearn.linear_model.LogisticRegression()` will **not** give you any points.
- Please do **not** modify anything if not mentioned.

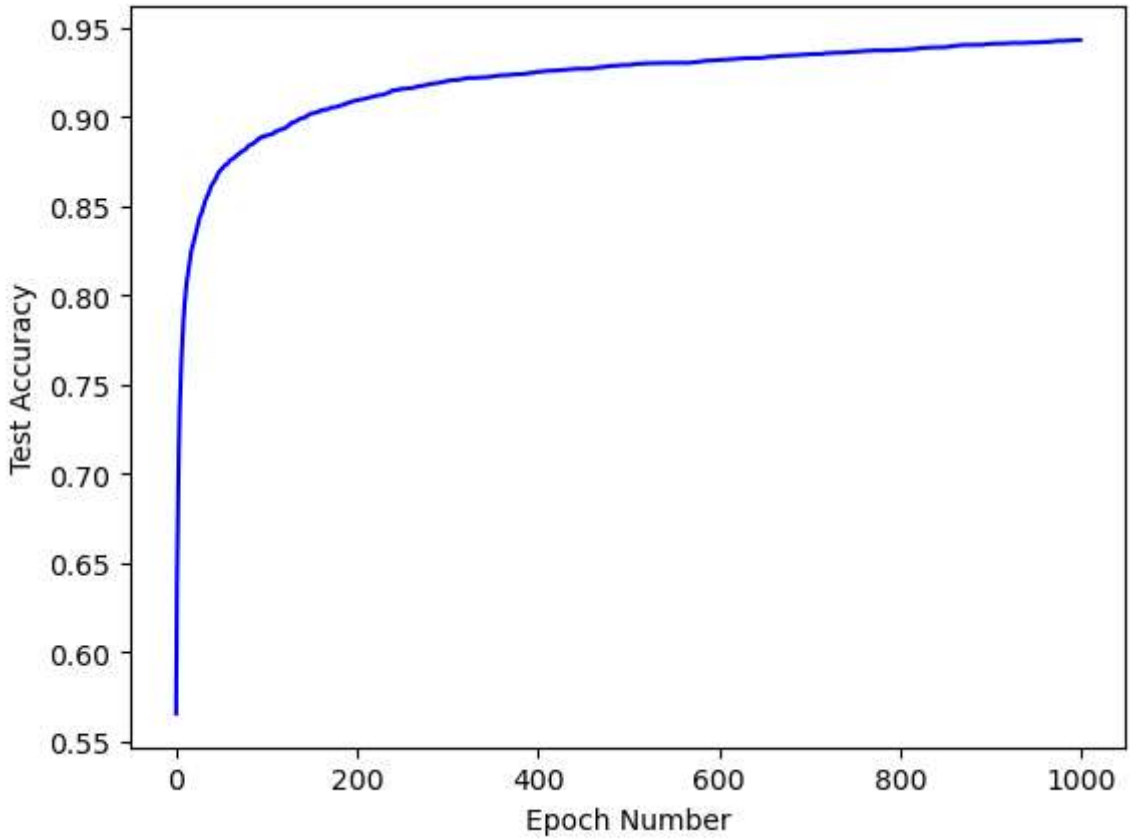
In [15]:
```
BATCH_SIZE = 512        # TODO: fill in and modify to see change in performance
LR = 0.001              # TODO: learning rate; fill in and modify to see change in performance

def get_one_hot(labels):
    cats = np.unique(labels)
```

```
    onehot = np.zeros((labels.size, cats.size))
    onehot[np.arange(labels.size), labels] = 1.
    return onehot

def plot(acc):
    plt.plot(np.arange(len(acc)),  acc, 'b-')
    plt.xlabel('Epoch Number')
    plt.ylabel('Test Accuracy')
    plt.show()

# X is a matrix of size n_samples x n_feature
# L is a vector of size n_samples x n_category
# theta is a matrix of size n_feature x n_category
def apply_gradients(X, L, theta):
    logits = np.matmul(X, theta)
    probs = np.exp(logits) / np.sum(np.exp(logits), axis=1, keepdims=True)
    grad = np.matmul(X.T, probs-L) / X.shape[0]
    new_theta = theta - LR*grad
    return new_theta

def mlr(trfeature, tr_onehot, tefeature, te_onehot):
    #tr_onehot = get_one_hot(trlabel)
    #te_onehot = get_one_hot(telabel)
    m_tr = tr_onehot.shape[0]
    m_te = te_onehot.shape[0]
    n_feature = trfeature.shape[1]
    n_category = tr_onehot.shape[1]

    theta = np.zeros((n_feature, n_category))
    diff = 1
    epoch = 0

    predonehot = []
    test_accuracy_list = []
    while diff > 1e-10 and epoch < 1000:
        theta_old = theta
        # Train
        for i in range(0, m_tr, BATCH_SIZE):
            endpos = min(m_tr, i+BATCH_SIZE-1)
            theta = apply_gradients(trfeature[i:endpos,:], tr_onehot[i:endpos,:], theta)

        diff = np.linalg.norm(theta_old-theta)

        # Predict on the test dataset
        logits = torch.mm(torch.tensor(tefeature, dtype=torch.float32), torch.tensor(theta, dtype=torch.float32))
```

```
        probs = torch.nn.functional.softmax(logits, dim=1)
        predonehot = probs.detach().numpy()

        pred = np.argmax(predonehot, axis=1)
        test_accuracy = np.mean(pred == telabel)
        test_accuracy_list.append(test_accuracy)

        epoch += 1
        # Update learning rate if you want

    plot(test_accuracy_list)

    print('Test Accuracy: %.5f'%(test_accuracy_list[-1]))

    return predonehot
```

Run the code below, check the accuracy plot, and report the test accuracy you obtain.

In [16]:
```
# get the extracted features from Question 1
trmat = sio.loadmat('mnist_train.mat')
temat = sio.loadmat('mnist_test.mat')

trfeature, trlabel = trmat['features'], trmat['labels']
tefeature, telabel = temat['features'], temat['labels']

teonehot = get_one_hot(telabel)
# fit multinomial logistic regression
# we will need the variable predonehot for next question
predonehot = mlr(trfeature, get_one_hot(trlabel), tefeature, teonehot)
```




```
Test Accuracy: 0.94300
```

Question 3: ROC [3 points]

In this problem, you need to implement the Receiver Operating Characteristics (ROC) curve. The output of the function `getROC` should be `TPR`, `FPR` representing True Positive Rate and False Positive Rate respectively.

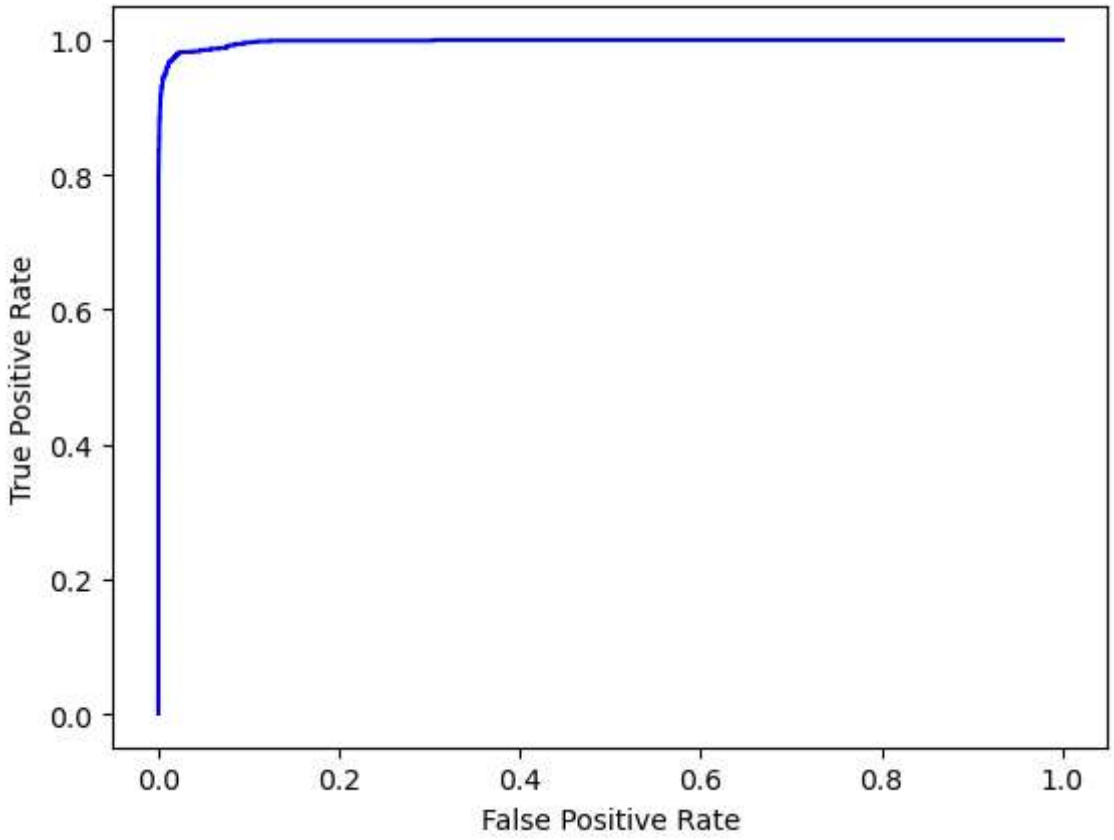
- Using built-in functions like `sklearn.metrics.roc_curve()` will **not** give you any points.

In [17]:
```
# pred is a vector of predictions of size n_samples x 1
# gt is the ground truth vector of 1 or 0 of size n_samples x 1 (1 indicates a positive and 0 negative)
# TPR is the True Positive Rate
# FPR is the False Positive Rate
def getROC(pred, gt):
    n_samples = len(gt)
```

```
    # Sort the predictions in descending order
    sorted_indices = pred.argsort()[::-1]
    sorted_pred = pred[sorted_indices]
    sorted_gt = gt[sorted_indices]

    # Initialize true positive and false positive counts
    TP_count = 0
    FP_count = 0

    # Initialize true positive rate and false positive rate lists
    TPR = []
    FPR = []

    # Iterate over all possible thresholds
    for i in range(n_samples):
        if sorted_gt[i] == 1:
            TP_count += 1
        else:
            FP_count += 1
        # Calculate TPR and FPR for current threshold
        TPR.append(TP_count / sum(sorted_gt))
        FPR.append(FP_count / (n_samples - sum(sorted_gt)))

    return TPR, FPR
```

Run the code below to plot the ROC you got for the 9-th category.

In [18]:
```
TPR, FPR = getROC(np.array(predonehot)[:, 9], teonehot[:, 9])

plt.plot(FPR, TPR, 'b-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

In [ ]: