ECSE 4965/6965

Introduction to Deep Learning

**Program Assignment 1**

Due date: 11:59 pm, Feb. 26th

In this programming assignment, you will implement techniques to learn a multi-class logistic regressor for image digit classification on the MNIST dataset (https://en.wikipedia.org/wiki/MNIST\_database). The classifier will take an image of a hand-written numerical digit between 1 and 5 as input and classify it into one of 5 classes corresponding to digit 1 to 5 respectively. Given the training data, you will follow the equations in the lecture notes to train a multi-class logistic regressor using the gradient descent method and then evaluate its performance on the given testing data. For training, you will learn the discriminant function regression parameters ******Wk**,Wk,0)t, where k=1,2,3, 4, 5, **Wk** is a vector for kth discriminant function, and Wk,0 is its bias.

Specifically, given the training data **D**={**x**[m], ***y***[m]}, m=1,2, …, M, where **x**[m] is a grayscale image of 28x28 (a 784x1 vector) and ***y***[m] is output vector that follows the 1 of K encoding, with kth element of being 1 and the rest being 0. You will implement the following methods to learn the parameters ****

1. For students taking this class at 4000 level, implement in Tensorflow the gradient descent method to solve for **** iteratively using all data in **D**. Initialize ****to small values and literately update ****with appropriate learning rate until convergence. Save ****and plot the (**W**k) for each class using matplotlib.pyplot functions (see below for details).
2. For students taking this class at 6000 level, implement in Tensorflow the Stochastic Gradient Descent method with L2 regularization. Initialize ****to small values and literately update ****with appropriate learning rate and regularization parameter  until convergence. Save ****and plot the (**W**k) for each class using matplotlib.pyplot functions (see below for details).
3. Using the given testing dataset **T**, evaluate the performance of your trained classifier by computing the classification error for each digit as well as the average classification errors

for all five digits. The classification error for each digit is computed as the ratio of incorrect classification to the total number images for that digit

Do not use Tensorflow’s existing gradient descent and stochastic gradient descent functions. Submit the following

1. a report that summarizes the theories for multi-class logistic regression, discuss your model architecture (number of hidden layers and nodes for each layer), experimental settings, hyper-parameters, and plot the training and testing errors over epochs or iterations and the learnt weights.
2. Show the classification performance in terms of classification error for each digit.
3. Submit your Tensorflow code and the saved weights **W** in the required format (see below).

**Data format**

The training data **D** contains 25112 training images (00001.jpg – 25112.jpg) in train\_data folder, and their labels are stored in train\_label.txt with each raw being the label of the corresponding image.

The testing data **T** contains 4982 testing images (00000.jpg – 04982.jpg) in test\_data folder, and their labels are stored in test\_label.txt.

**Load images and normalize**

Load the images in the order

Convert it to vector and normalize it to [0,1] by dividing the intensity of each pixel by 255

The following link teaches how to load images:

<http://learningtensorflow.com/lesson3/>

**Output**

Use the code below to save the learned parameters matrix **W** in the following format



**W**k is the learnt weights for the kth digit and Wk,0 is the corresponding bias.

--------------------

import pickle

filehandler = open("multiclass\_parameters.txt","wb")

pickle.dump(**W**, filehandler)

filehandler.close()

-----------------

**Plot**

Use the following function to plot **W**k as an image for each digit:

-----------------

import matplotlib.pyplot as plt

Img = W\_k.reshape(28,28)

plt.imshow(img)

plt.colorbar()

plt.show()

-----------------