DSCI 6671 ST: Deep Learning
Project 1: Fully Connected Neural Network
Due by September 29 11:59 PM

In this project, you will explore a feedforward fully connected neural network. Note that implementation should be done in python using low level functions like dot, exp, max, min, sum, logarithm. Only exception is pseudo-random generator. Furthermore, employ efficient computation with minimal use of 'for' loop.

1. In the first part, build and train a neural network of three layers (including the output layer) to perform classification of Iris dataset in order to validate that your program works. Details of the network and training procedure are as follows:
    a. Each hidden layer includes 100 neurons. Use ReLU as the activation function for hidden units and softmax for the output neurons.
    b. Partition the dataset into 120 for training and 30 for test.
    c. Initialize the weights sampled from a Gaussian distribution with mean of zero and variance of $\frac{2}{number\ of\ inputs}$, which is known as Kaiming initialization. Initialize bias to zeros.
    d. Train the network by minimizing cross-entropy loss function using gradient descent method for 1000 epochs.
    e. Test the performance of the learned model on the test data.
    f. Plot train and test accuracies vs. epochs. Which point should you choose for your model and why?

Since step b requires some high-level function, I am providing you with its code here.

```
import numpy as np
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data
y = iris.target

train_idx =
np.concatenate((np.arange(0,40),np.arange(50,90),np.arange(100,140)))
test_idx = np.setdiff1d(np.arange(0,150), train_idx)
X_train = X[train_idx]

#normalize the training data in the range of [0,1]
max_X = np.max(X_train, axis=0)
min_X = np.min(X_train, axis=0)
X_train = (X_train - min_X)/(max_X-min_X)

X_test = X[test_idx]
X_test = (X_test - min_X)/(max_X-min_X)

y_train = y[train_idx]
y_test = y[test_idx]
```

2. The goal of this part to investigate the behavior of neural network components (e.g., activation function), architecture, and the selection of hyperparameters on the learning performance through experiments on MNIST dataset for classification task. While this dataset consists of 2D images of 10 handwritten digits (0 to 9), you can only use two classes corresponding to digits 1 and 3 for this project. Flatten each image to have 784-length one-dimensional vector to be used as inputs to fully connected network.

   At least, you should study the followings.
   a. Plot training and validation loss against epochs.
   b. Impact of the size of mini-batch.
   c. Compare loss functions - Mean squared error vs. Cross-entropy.
   d. Impact of initialization. How does it affect the gradients?
   e. Learning rate.
   f. Impact of width vs. depth on the model performance.
   g. The role of various activations functions such as ReLU, sigmoid, hyperbolic tangent.
   h. Visualized some misclassified objects from the test data and explain why the model failed for those cases.

Instructions for dataset downloading and processing:
Download the MNIST dataset from https://s3.amazonaws.com/img-datasets/mnist.npz and save it to your local storage.

Load MNIST dataset:
```
import numpy as np
f = np.load('dir_to_mnist.npz')
X_train = f['x_train'].astype(np.float32)
X_train = X_train.reshape(X_train.shape[0],-1)
X_test = f['x_test'].astype(np.float32)
X_test = X_test.reshape(X_test.shape[0],-1)


X_train /= 255. # normalize
X_test /= 255.

y_train = f['y_train'].astype(np.float32)
y_test = f['y_test'].astype(np.float32)
```

You should submit a complete final report including an introduction, method detailing steps and equations, and an in-depth analytical study of your findings. Also, submit your working code so I can check if I intend to do so.