THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Machine Learning II
Miniproject #1
A. Jafari, M. Hagan
Due: *Nov-06-2017*

Q1:Training MLP networks with Torch on MNIST dataset

The objective of this miniproject is to become familiar with Torch for training deep networks on large datasets. You will be provided with a sample program that loads in the MNIST data set and sets up a simple multilayer network to be trained on it. (See http://yann.lecun.com/exdb/mnist/ for a description of the data set and a list of various efforts to train on it.) The instructions below provide a rough framework of what you should do for the miniproject. You should experiment with using Torch on this large dataset. The project is open-ended. Learn as much as you can about using deep multilayer networks, and relate what you have learned in your project report. For all parts below, and any other experiments you run, include the results into one **PDF file**, and upload it to the BlackBoard. Include all program listings, plots, command line printouts, discussion, etc.

1. We are using the dp package for Torch. If you are using GCP or HPC, dp should already be installed.

2. Download the `train_mnist.lua` file from GitHub (See https://github.com/amir-jafari/Deep-Learning/tree/master/Torch_). It is based on the following website https://github.com/nicholas-leonard/slides/blob/master/torch7.md, which you may want to review before proceeding.

3. Run the program in ZerobraneStudio and investigate and verify its performance.

4. Modify the program to run on the GPU.

5. The program is set up to perform stochastic gradient descent. Modify the program to use **minibatches**. Experiment with different minibatch sizes.

6. Experiment with different numbers of layers and different numbers of neurons. Maintain the total number of weights and biases in the network, while increasing the number of layers in the network. Describe how the performance changes as the number of layers increases – both in terms of training time and performance.

7. Try one other training function from the list on this page: https://github.com/torch/optim/blob/master/doc/algos.md. Compare the performance with gradient descent.

8. Try different transfer functions and do a comparison study between those. Explain your results.

9. Make a chart and or table to see the advantage between mini bach , full batch and stochastic gradient. Calculate the timing by using the system clock command.

10. **Bonus**: Plot confusion matrix (as table or a figure) and explain each element of the confusion matrix. Calculate accuracy and misclassification rate.

11. **Bonus**: Plot ROC curve and explain the results. Calculate AUC and explain the results. Note: You need to explain ROC and AUC and the use of these techniques in pattern recognition.