MSAI-349, Fall 2021
Homework #3: Neural Networks
Due Date: Friday, November 12th @ 11:59PM
Total Points: 10.0 (plus optional 2.0 bonus points)

In this assignment, you will work with your project group to implement the softmax function and two multi-class classification neural networks. You may discuss the homework with other groups, but do not take any written record from the discussions. Also, do not copy any source code from the Web or other groups.

You will be working with a fictitious dataset to estimate the insurability of individuals based upon age and lifestyle choices. There are three independent variables (`age, exercise and cigarettes`) and three classes (`good, neutral and bad`) in this dataset. From this data we constructed `three_train.csv`, `three_valid.csv` and `three_test.csv` splits with 2000, 200 and 200 observations, respectively.

In addition, you will be working with the MNIST handwritten digits dataset from Homework #2. In this dataset the digits zero through nine are represented as a series of integers on a 28x28 grid, where the integers provide the grayscale intensity. From this data we constructed `mnist_train.csv`, `mnist_valid.csv` and `mnist_test.csv` splits with 2000, 200 and 200 observations, respectively. Each of the files contain one observation per row with a 'label' followed by 784 'grayscale' intensity values.

A short program `starter3.py` contains function signatures for each of the four networks that you will be asked to implement (three required and one optional) as well as examples of how to read each dataset.

Steps to complete the homework

1. (4.0 points) Implement a simple feed-forward neural network in PyTorch to make classifications on the "insurability" dataset. The architecture should be the same as the one covered in class (see Slide 11-8). You can implement the neural network as a class or by using in-line code. You should use the SGD `optimizer()`, and you must implement the `softmax()` function yourself. You may apply any transformations to the data that you deem important. Train your network one observation at a time. Experiment with three different hyper-parameter settings of your choice (*e.g.* bias/no bias terms, learning rate, learning rate decay schedule, stopping criteria, temperature, etc.). In addition to your code, please provide (i) learning curves for the training and validation datasets, (ii) final test results as a confusion matrix and F1 score, (iii) a description of why using a neural network on this dataset is a bad idea, and (iv) short discussion of the hyper-parameters that you selected and their impact.
2. (4.0 points) Implement neural network in PyTorch with an architecture of your choosing (a deep feed-forward neural network is fine) to perform 10-class classification on the MNIST dataset. Apply the same data transformations that you used for Homework #2. You are encouraged to use a different optimizer and the cross-entropy loss function that comes with PyTorch. Describe your design choices, provide a short rationale for each and explain how this network differs from the one you implemented for Part 1. Compare your results to

Homework #2 and analyze why your results may be different. In addition, please provide the learning curves and confusion matrix as described in Part 1.

3. (2.0 points) Add a regularizer of you choice to the 10-class classifier that you implemented for Part 2. Describe your regularization, the motivation for your choice and analyze the impact of the performance of the classifier. This analysis should include a comparison of learning curves and performance.
4. (2.0 bonus points) Re-implement the 3-class classification feed-forward neural network from Part 1 by calculating and applying the gradients without a PyTorch optimizer. Do not use a bias term for this part of the assignment. Also, it may be easier to perform Part 4 with in-line code (*e.g.,* weight vectors, matrix algebra operations and your own `sigmoid()` function). Note: You can check your gradient calculations by comparing before/after update weight matrices against a parallel implementation using a PyTorch optimizer.

Please note there several aspects of this assignment are deliberately vague. This is to give you some experience with the issues you will likely encounter when working on your Final Projects. You may need to make simplifying assumptions about your data, devised workarounds for computation bottlenecks or balance time spent on hyper-parameter tuning versus results.

Submission Instructions

Turn in your homework as a single zip file, in Canvas. Specifically:

1. Create a single pdf file hw3.pdf with the answers to the questions above and summaries of your results.
2. Create a single ZIP file containing:
   - `hw3.pdf`
   - All of your `.py` code files
3. Turn the zip file in under Homework #3 in Canvas.

Good luck, and have fun!