ECE 449/590, Fall 2022
Project 5: Training with MLP Model

Due: 12/4 (Sun.), by the end of the day (Chicago time)
No Extension

1 Summary

In this project, you are required to implement the backprop algorithm together with SGD to train the MLP model that is slightly different than what we have used in Project 4 for the MNIST data. For you convinience, we have processed the MNIST training data and provided a sample python program for you to start with. You will need to implement the training process utilizing only the NumPy library.

This project should be done individually. Discussions are encouraged. However, all the programs (except those from the lectures) and writings should be by yourself. COPY without proper CITATION will be treated as PLAGIARISM and called for DISCIPLINARY ACTION.

NEVER share your programs/writings with others.

2 Working with Your Projects

Please continue to work with your Git repository for Project 5. Here is a brief introduction of the files.

- `prj05.py`: this contains the sample python program for you to start with and you should only modify this file for this project. This program will store the trained model in the file `p5_params.npz`.
- `grade_p5.py`: this is the grading script to verify your trained model for Project 5 that is stored in `p5_params.npz`. There are 4 levels of accuracy goals organized into 4 questions. You should not modify this file.
- `mnist_train.npz`: the training examples from the MNIST dataset, stored as NumPy arrays for you to train the MLP model.

After training the model, the file `p5_params.npz` should be created or updated. Then run the grading script to see if all questions pass.

```
python3 prj05.py
ls -l p5_params.npz
python3 grade_p5.py
```

3 Deliverables and Grading

You should commit and push `prj05.py` and `p5_params.npz` frequently to avoid loss of data due to any possible failure with your system.

Project 5 will have a full grade of 100 points. Each question from `grade_p5.py`, if passed in the CI system, will give you 5 points, and a failed question will earn 0 points. If at least one of these questions is passed, you will earn 30 points for your python code in `prj05.py`; otherwise, we will give at most 25 points for your `prj05.py` depending on how close your code is from completion. Please make sure you are not using data from `mnist_test.npz` directly or indirectly in `prj05.py`; **otherwise we will deduct** 30 **points from your project grading**. This will not be a concern if you do not modify `prj05.py` to read additional files.

You are required to submit a project report of 3-6 pages for 50 points to Blackboard. The project reports should include the following items.

- (10 points) A brief description of the MNIST dataset, the MLP model, and the training process.
- (10 points) Discuss the different purposes of the training, validation, and testing data.
- (30 points) How the three hyperparameters `bound`, `epsilon`, and `batch_size` affect the accuracy through validation? You should experiment with different combinations of the three hyperparameters and use tables and figures to explain your discovery.

Here are a few hints for you to implement `prj05.py`.

- Read the code carefully to understand the general training flow and then identify the parts that you need to modify.
- Make sure you fully understand Homework 3 Question 2 before you start to modify the code.
- Implement backprop and update_theta first. DO NOT introduce new initialization algorithm or variants of SGD.
- If your implementation is correct, without tuning the hyperparameters you should be able to see better accuracy after the first epoch than that of random guess. (Accuracy for random guess is around 0.1 as there are 10 classes.)

- Try different set of hyperparameters. Instead of changing the value of a hyperparameter into another one that is close, e.g. from 1 to 1.1 or 0.9, try to change it a lot, e.g. from 1 to 10 or 0.1. In addition, if your processor is slow, you don't need to use all the 20000 training samples for this purpose.
- Training of 10 epochs should be more than enough if your choice of hyperparameters is reasonably good. If your implementation is reasonably efficient, the whole training process will take 1 to 10 minutes depending on how good your processor is.