ECE 449/590, Fall 2022
Project 6: Training with CNN Model

Due: 12/7 (Wed.), 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 CNN model that is slightly different than what we have used in Project 4 for the MNIST data. You should first complete Project 5 before working on this project. 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 6. Here is a brief introduction of the files.

- `prj06.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 `p6_params.npz`.
- `grade_p6.py`: this is the grading script to verify your trained model for Project 6 that is stored in `p6_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 CNN model.

After training the model, the file `p6_params.npz` should be created or updated. Then run the grading script to see if all questions pass.

```
python3 prj06.py
ls -l p6_params.npz
python3 grade_p6.py
```

3 Deliverables and Grading

You should commit and push `prj06.py` and `p6_params.npz` frequently to avoid loss of data due to any possible failure with your system. **We will NOT have extensions for this project. If you system fails just before the deadline and you have not pushed anything to the CI system, you will get ZERO point for your code and model.**

Project 6 will have a full grade of 100 points. Each question from `grade_p6.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 `prj06.py`; otherwise, we will give at most 25 points for your `prj06.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 `prj06.py`; **otherwise we will deduct** 30 **points from your project grading**.

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 CNN model and the training process.
- (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.
- (10 points) Discuss the difference in capacities of your MLP model in Project 5 and the CNN model for Project 6.

Here are a few hints for you to implement `prj06.py`.

- Implementations of Conv2d and MaxPool as well as their backprop algorithms are provided. You may use them to greatly simplify your code and to improve the efficiency.
- It is sufficient to train the CNN model with only 20000 or less samples and in a total of 5 epochs.

Hints below from Project 5 also apply.

- 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.