CMPUT 328 Fall 2023

Assignment 1

Worth 8% of the total weight

Part 1 (In-lab): Logistic Regression [worth 1% of the total weight]:

Implement Logistic Regression in PyTorch (You can make use of the multiple linear regression notebook from lecture):

a) Train and test on MNIST by defining your own data pipeline for training, validation and testing using PyTorch dataloader.
   - Use the last 12,000 samples of the train set as validation set.
   - Test the trained model on the validation set every few epochs to prevent overfitting
   - **Do not use the test set for training**.

b) Add a **regularization term** to improve your model (L1 or L2 regularization, whichever gives better accuracy)

Expected Performance: A correctly implemented, and somewhat well-tuned version of this algorithm will have an accuracy of **92-94%** on both test and validation sets of MNIST

Quiz [worth 2% of the total weight]

You are expected to have completed this part by the end of the first lab, though the submission deadline for this part is the same as the other parts. There will be an in-lab quiz at the end of that lab with questions related to this part.

We will also collect attendance in the lab at some point during the lab hours using your ONEcard for identity verification.

If you are not present in the lab (or fail to produce your ONEcard) when the attendance is taken, you will lose the marks for the quiz (2% weight) entirely.

Part 2 (Out-of-lab): Repeat part 1 for CIFAR10 [Worth 1% of the total weight]

Take note of the differences between MNIST and CIFAR10 datasets to make your code work for both. For example:

- MNIST images are grayscale, while CIFAR10 images are RGB.
- MNIST and CIFAR10 images have different spatial dimensions (height and width).

Expected accuracy is **38-40%** for CIFAR10 dataset.

You need to complete the function ***logistic_regression*** in *A1_submission.py* for parts 1 and 2.

Part 3 (Out-of-lab): Hyperparameter Search [Worth 4% of the total weight]

Find optimal hyperparameters for Adaptive Moment Estimation (Adam) and Stochastic Gradient Descent (SGD) on CIFAR10 dataset including the regularization method you used in part 1.

- You should perform **grid search** or **random search** for finding the optimal hyper-parameters using accuracy on the validation set and select the best configuration.
- You can also use more advanced search strategies like evolutionary search, but you are **not** allowed to use any automatic parameter search methods like *scorch.*
- You **cannot** use the test set during this process.

You need to complete the function ***tune_hyper_parameter*** in *A1_submission.py* for this part.

Template Code

You are provided with template code in the form of two files: *A1_main.py* and *A1_submission.py*.

You need to complete the two functions in *A1_submission.py* (i.e. *logistic_regression* for part 1 and 2, *tune_hyper_parameter* for part 3).

You can add any other functions or classes you want to *A1_submission.py* but do not make any changes to *A1_main.py.*

Running the code

Your own machine

Install python (version >= 3.6) if needed and install the required packages by running:

```
python3 -m pip install numpy torch torchvision tqdm paramparse
```

Run the code using:

```
python3 A1_main.py
```

It is recommended to use an IDE like pycharm or vscode to make debugging easier.

Colab

Run this from a code cell in a notebook:

```
!python3 "<full path to A1_main.py >"
```

You can optionally install the *paramparse* package to enable command line arguments :

```
!python3 -m pip install paramparse
```

You can then use command line arguments as:

part 1:

```
!python3 "<full path to A1_main.py >" mode=logistic dataset=MNIST
```

part 2:

```
!python3 "<full path to A1_main.py >" mode=logistic  dataset=CIFAR10
```

part 3:

```
!python3 "<full path to A1_main.py >" mode=tune  dataset=MNIST
!python3 "<full path to A1_main.py >" mode=tune  dataset=CIFAR10
```

Alternatively, you can directly copy code from the template scripts into notebook cells and run those. However, make sure that *A1_main.py* runs on Colab without errors using the above method before submitting *A1_submission.py* since that is how it will be evaluated.

Submission

You need to submit only the completed *A1_submission.py*. Make sure to import any additional libraries you need so it can be used as a standalone Python module from *A1_main.py*.

To reiterate, please **do not** submit *A1_main.py* or any other files generated by running the code.

Marking

Parts 1 and 2: Marks will depend on correctness of the implementation along with the following metrics:

- **Runtime**: The total runtime of your submission (including training and testing) **should not exceed 300 seconds** for either dataset on Colab GPU.
  - One trick to improve your run time is to grid search the hyperparameters first but only put in the best hyperparameters you found in your submission.
- **Accuracy**: Each dataset will give you 50% of the total score.
  - **MNIST**: score scales linearly from **84 - 94%** accuracy on the test set
  - **CIFAR10**: score scales linearly from **30 - 40%** accuracy on the test set

Part 3: Marks will depend only on the correctness of your search implementation.

- **Runtime**: The runtime of your submission **should not exceed 1500 seconds** on Colab GPU.
- **Accuracy**: There are no specific accuracy requirements except there should be improvement in loss / accuracy compared to part 1.

Collaboration Policy

This must be your own work.

- Do not share or look at the code of other people (whether they are inside or outside the class).
- Do not search for or copy code from the Internet.
- You can talk to others that are in the class about solution ideas but not in so much detail that you are verbally sharing or hearing about or seeing the code.
- You must cite whom you talked to and what web resources you used in the comments of your programs.

All submissions involved in detected cases of plagiarism will be penalized with no marks for this assignment irrespective of who copied from whom.