Exploring Classification Using Feed Forward Neural Network

1 Introduction

In HW-2, you will be implementing classification on toy datasets with Feed Forward Neural Network using PyTorch Lightning. There are two datasets, one is a 4-class classification dataset and the other is digits dataset (10 classes). Your key goal in this assignment is to correctly implement the FFNN model and analyze the results you obtain.

Starter code and dataset are available on the same github repository under the directory "hw2": https://github.com/ashutoshbsathe/cs725-hw/tree/main/hw2

Before starting, install all the modules required for this homework which are mentioned in requirements.txt.

2 Data

The data/ directory from our repository contains the toy datasets. The data/simple directory contains dataset for 4-class classification. The data/digits directory contains dataset for 10-class classification. Each of these directories contains the training and validation data in the form of NumPy arrays.

The train-set source (input) is named as `train_x.npy` and the train-set target is named as `train_y.npy`. Similarly the validation-set source (input) is named as `valid_x.npy` and the validation-set target is named as `valid_y.npy`.

3 Starter Code

The starter code contains the following files:

1. *args.py*: This file specifies the Python classes for training arguments and visualization arguments. You can check all the arguments and their default values in this file. You should not modify this file.
2. *model.py*: Contains boilerplate implementation of FFNN. You must go through the comments of each function to understand their expected behavior. There are three class definitions in this file. `LitGenericClassifier` for general functions which are to be used for both the datasets, `LitSimpleClassifier` for functions specific to the "simple" dataset, and `LitDigitsClassifier` for functions specific to the "digits" dataset. You need to implement the functions which are there in these three class definitions.
3. *train.py*: A minimal training loop that makes use of above functions. Also neatly saves training history and the weights that lead to best validation accuracy to a folder. You should not modify this file.
4. *train_with_visualization.py*: Similar to `train.py` above but also visualizes decision boundary of logistic regression after every gradient update.
5. *utils.py*: Various utility functions used by other scripts. Do not use this file for defining your own utilities, make sure your implementation is contained entirely in `model.py` itself.
6. *evaluate_submission.py*: Use this after completing the assignment to get an idea of what accuracy and loss values the autograder will see with your model. Do note that the actual autograder will use the test split of the model which is not available to you during the assignment.

Specifically, your task is to complete the FFNN model implementation in *model.py*.

4 Report

After the implementation is complete, study the effects of various design choices of hyper-parameters like learning rate, number of training epochs etc. The values of these hyper-parameters can be set accordingly through argument passing while executing *train.py* on CLI. The exact command can be found on the github repository. In your report, you must provide analysis for the following:

Analyze the effect of *learning_rate* and *num_epochs* on the loss and accuracy. Run the experiment with different values of the hyper-parameters *learning_rate* and *num_epochs* and analyze how these affect the training and validation performance as measured by accuracy and loss (4 plots total for each dataset). You can try any values of *learning_rate* and *num_epochs* you want to experiment with. You need to perform this analysis for both of these classification tasks (i.e. for both datasets). You also need to report a) The best epoch, b) Data-preprocessing performed (if any, write "not done" if you did not require any preprocessing) and c) What did you do in order to prevent overfitting.

1. Classification on **simple**(4-class) dataset: **[7 Marks]**
2. Classification on **digits** dataset: **[7 Marks]**

The main focus of this analysis is to analyze the effect of hyper-parameter values on loss and accuracy.

Apart from the above analysis, there are also some marks for the following.

1. Performance of your model on the blind **test-sets** (not available to you). **[5+5 Marks]**
2. Your position on the **Kaggle Leaderboard**. **[4 Marks]**
   There is a Kaggle competition for the **digits** part of this homework. You can join the competition at https://www.kaggle.com/t/7beff2ca03e443269091305d96e4efa2. All you need to do is to submit the predictions you get for your validation-set there on Kaggle. you can generate the file to be submitted using make_kaggle_submission.py. The leaderboard will rank teams according to higher accuracy. More details about this can be found on the competition page.
   NOTE-1: One person from a group should join the competition. After making the first submission you can add your team details there under the "team" tab. You need to strictly follow the team naming convention which is mentioned there on the competition page.
   NOTE-2: Do not worry if the accuracy shown on Kaggle is different from the one you get during your experiment. This may happen because only half of the validation-set is being used to judge your predictions on the public leaderboard of Kaggle. The other half will be used to generate the ranking on the private leaderboard, which will only be visible when the competition ends.
   NOTE-3: Kaggle is for learning with healthy competition. You should only submit the predictions you get using your trained model. Your performance on Kaggle will be cross-checked by evaluating the model submitted by you on the validation set. If any discrepancy is found the group will be penalized for that.

5 Submission Structure

Once you are done with the assignment, submit (1) your best models for both the datasets (2) your code for reproducing the best results i.e. your *model.py* (3) the report with the results you obtain. Since the assignment will be autograded, it is important to maintain the submission structure as mentioned below:

```
submission/
    model.py
    best_simple.ckpt
    best_digits.ckpt
    report.pdf
```

Pack everything under `submission/` directory under a `.tar.gz` archive named as `rollno1_rollno2.tar.gz` and upload that to Moodle. Refer to the *README.md* on the repo for more concrete instructions.