CS 725 (Autumn 2022): Programming Assignment

This assignment is due by **11.59 PM on October 30, 2022**.

General Instructions

Please read the following important instructions before getting started on the assignment. 1. This assignment can be completed in groups. (Groups of size 2-5, as recommended for the final projects, will be allowed. Individual submissions will also be allowed.) 2. This assignment is entirely programming-based. A corresponding Kaggle task is hosted here. Please signup on Kaggle using your IITB LDAP email accounts, with Kaggle `Display Name = <your_roll_number>`. Reach out to the TAs for more instructions on joining the Kaggle competition. 3. Your final submission should be a .tar.gz bundle of a directory organized exactly as described in the Submission Instructions. Submissions that do not strictly adhere to this structure will be penalized. 4. Successful completion of the assignment would include: (A) Submitting .tar.gz on Moodle and (B) Having your roll number appear on the Kaggle leaderboard for the regression task. It is enough for one of your team members to make the submission and represent your team on the leaderboard.

Implement a Feedforward Neural Network using Python

This assignment will familiarize you with training and evaluating feedforward neural networks. You will work on a regression task where you will have to predict the release year of a song from a set of timbre-based audio features extracted from the song. This consists of a year range between 1922 to 2011. (More details about this corpus are available here.) Click here to download the training, development and test sets from Kaggle.

Dataset Information

- The dataset contains three files `train.csv`, `dev.csv`, and `test.csv`
- Each row in the `__.csv` file contains timbre-based audio features extracted from a song.
- The dataset has 90 features: 12 timbre average values and 78 timbre covariance values. Each column denotes a feature.
- `train.csv` and `dev.csv` contains following columns:

```
1. label - Year of release of the song in the range [1922, 2011]
2. TimbreAvg1
3. TimbreAvg2
.
.
13. TimbreAvg12
```

```
14. TimbreCovariance1
15. TimbreCovariance2
.
.
91. TimbreCovariance78
```

- `test.csv` contains same features except `label`

Part 1

In Part 1, you will implement the neural network, train it using the data in `train.csv` and report its performance on dev data in `dev.csv`.

Part 1.A (25 Points)

Implement the functions definitions given in nn.py to create and train a neural network. Run mini-batch gradient descent on Mean Squared Error (MSE) loss function. We have provided a helper file helper.md to give step-by-step introduction to the nn.py file

For both Part 1.A and Part 1.B, use fixed settings:

```
- Seed for numpy: 42
- Use ReLU activation function for ALL HIDDEN layers.
```

Initialization of weights and biases (for both Part 1 and Part 2) Initialize weights and biases using the uniform distribution in the range [-1, 1].

What to submit in Part 1.A? For Part 1.A, only code needs to be submitted in the file `nn_1.py`.

Part 1.B (10 Points)

Plot a graph showing the train and dev set loss after each epoch on each of the batch sizes: 32 and 64. Do it for the first 100 epochs. You can use existing libraries (for example, `matplotlib`) to create the plots. No restrictions are imposed on the remaining hyper-parameters.

What to submit in Part 1.B? Image files: - `train_32.png`: Plot for train set loss for batch size 32 - `dev_32.png`: Plot for dev set loss for batch size 32 - `train_64.png`: Plot for train set loss for batch size 64 - `dev_64.png`: Plot for dev set loss for batch size 64

Part 2 (10 points)

In Part 2, you will evaluate your network's performance on test data given in `test.csv`.

In this part, there is no restriction on any hyper-parameter values. You are also allowed to explore various hyper-parameter tuning and cross-validation techniques.

You are also free to create any wrapper functions over given functions in nn.py

Submit your predictions on test data on Kaggle competition in a `<roll_number>.csv` file in the following format:

```
Id,Predictions
1,2000
2,1976
3,2002
.
.
5100,1943
```

In a CSV file (`part_2.csv`), write the name of the hyper-parameter/any new variable you've introduced and the value you set it to.

What to submit in Part 2? Create a two-column csv file `part_2.csv` and write the name of hyper-parameter in first column and value in the second column. Also submit your code for part 2 in `nn_2.py`. This code may contain all the enhancements you did for part 2.

For example:

```
Name,Value
learning_rate,0.001
batch_size,30
dropout,0.10
```

All teams who outperform or perform comparably to the TA's baseline/untuned network's performance on `test.csv` will get 5/10 points straightaway. The remaining 5 points will depend on your team's standing on the leaderboard.

Part 3 (5 points)

Say you are asked to do some form of feature selection and create a new feature set based on the original feature set consisting of 90 features. The size of the new feature set should be strictly smaller than 90, and performance using this subset of features should ideally be no worse than what you get with using the complete feature set. Note that each feature in the new feature set could be a combination of features in the original feature set. You are free to use any technique to identify potentially useful features and create any wrapper functions over given functions in nn.py.

What to submit in Part 3?

- Submit your code for this part in `nn_3.py` that contains the implementation for feature selection. (The starting point for this part can be the best settings you have identified for part 2.)
- In a CSV file (`part_3.csv`), write the name of the hyper-parameter/any new variable you've introduced and the value you set it to.
- In another CSV file (`features.csv`) with a single column, write the features (from the original features) that are present in your new feature set. Remember to take care of transformation in the wrapper function itself (and total number of features < 90 altogether). For example:

```
TimbreAvg2
TimbreAvg5
TimbreCovariance5
TimbreCovariance10
```

Extra credit assignment (10 points)

You will have to work on a classification task where you will predict a label among ("Very Old", "Old", "New" and "Recent") based on when the song was released. Click here to download the training, development and test sets from Kaggle. The corresponding kaggle task is hosted here.

Dataset Information

- The dataset contains three files `train.csv`, `dev.csv`, and `test.csv`
- Each row in the `__.csv` file contains timbre-based audio features extracted from a song.
- The dataset has 90 features: 12 timbre average values and 78 timbre covariance values. Each column denotes a feature.
- `train.csv` and `dev.csv` contains following columns:

```
1. label - "Very Old", "Old", "New" and "Recent", based on when it was released
2. TimbreAvg1
3. TimbreAvg2
.
.
13. TimbreAvg12
14. TimbreCovariance1
15. TimbreCovariance2
.
.
91. TimbreCovariance78
```

- `test.csv` contains same features except `label`

Task

You need to modify the nn.py used in the regression task to support the classification task and evaluate your network's performance on test data given in `test.csv`.

In this task, there is no restriction on any hyper-parameter values. You are also allowed to explore various hyper-parameter tuning and cross-validation techniques.

You are also free to create any wrapper functions over given functions in nn.py

Submit your predictions on test data on Kaggle competition in a `<roll_number>.csv` file in the following format:

```
Id,Predictions
1,Very Old
2,Old
3,New
.
.
5100,Recent
```

In a CSV file (`params.csv`), write the name of the hyper-parameter and the value you used.

What to submit? Create a two-column csv file `params.csv` and write the name of hyper-parameter in first column and value in the second column. Also submit your code for part 2 in `nn_classification.py`. This code may contain all the enhancements you did for part 2.

For example:

```
Name,Value
learning_rate,0.001
batch_size,30
dropout,0.10
```

Tips to improve your rank on leaderboard You can explore following techniques to get better generalization performance - Feature Scaling - Feature Selection - Dropout - Batch Normalization - Early Stopping

Submission Instructions

- Your submission directory should be:

```
<your_roll_number>
  nn_1.py
  nn_2.py
  nn_3.py
```

```
part_2.csv
part_3.csv
features.csv
plots_1b
    dev_32.png
    dev_64.png
    train_32.png
    train_64.png
classification `This submission directory is optional`
    params.csv
    nn_classification.py
readme.txt
```

- Use `Readme.txt` to describe any other information needed to run your code successfully.
- Add these files to directory `<your_roll_number>`.
- Compress the directory `<your_roll_number>` in .tgz format using following command:

  ```
  tar -czf <your_roll_number>.tar.gz <your_roll_number>
  ```

- Submit the `<your_roll_number>.tar.gz` file.

Resources for Python This could be useful for both beginners and those with some Python background. Please reach out to the TAs if you need more resources/pointers. - Intro to Python by Brandon Rohrer - Learn Python in Y Minutes - Automate the Boring Stuff with Python by Al Sweigart `For those with more time on their hands`