CS6375 Assignment 1

<https://github.com/iAmBrig12/CS6375_Assignment1>

Brigham Thornock

bxt210005

## Introduction and Data

The intent of this project is to predict Yelp review scores based on the given review. This is done using a dataset that includes review text associated with star rating y ∈ Y := {1, 2, 3, 4, 5}. My role in the project was to implement the forward function of the neural network, where the model performs transformations on the data to produce a prediction. I implemented the functions for both a Feed Forward Neural Network (FFNN) and a Recurrent Neural Network (RNN). I also added the Glove word embedding to the RNN to vectorize the text data as well as adding testing functionality and CUDA compatibility for GPU acceleration.

The initial data given was already split into training, testing, and validation datasets. This data, however, was imbalanced and would therefore lead to inaccurate results. I chose instead to combine all of the data and split it on my own, 80% for training, 10% for validation, and 10% for testing. This led to a more balanced dataset that could be properly used for training and testing.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Score Distribution before Processing | | | | | |
| Dataset | 1 | 2 | 3 | 4 | 5 |
| Training | 3200 | 3200 | 1600 | 0 | 0 |
| Validation | 320 | 320 | 160 | 0 | 0 |
| Testing | 0 | 0 | 160 | 320 | 320 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Score Distribution after Processing | | | | | |
| Dataset | 1 | 2 | 3 | 4 | 5 |
| Training | 2783 | 2830 | 1551 | 263 | 253 |
| Validation | 362 | 361 | 170 | 34 | 33 |
| Testing | 375 | 329 | 199 | 23 | 34 |

Note that after processing, the labels are more distributed. There is, however, still a partial class imbalance as there are fewer 4 and 5 star reviews in the dataset overall.

## Implementations

#### FFNN

A computer screen shot of a program code

AI-generated content may be incorrect.

No changes were made to the \_\_init\_\_ and compute\_Loss functions.

The forward function operates as follows:

1. Input vector is transferred to the proper device to ensure that the model will use the GPU if Cuda is available, otherwise it will use the CPU
2. Input is fed into the first hidden layer, with dimension h
3. Output of the hidden layer is passed through the Leaky ReLU activation function to introduce nonlinearity
4. Output is passed through the last linear layer to get the raw output scores
5. Softmax layer converts the raw scores into log-probabilities for classification

#### RNN

A screen shot of a computer program

AI-generated content may be incorrect.

The numOfLayer value in \_\_init\_\_ was changed to 8 so the RNN part of the network would have 8 hidden layers. The compute\_Loss function was not changed.

The forward function operates as follows:

1. Move inputs to GPU or CPU
2. Pass input through RNN layers and get hidden states for all time steps and final hidden state
3. Sum all hidden states to aggregate all information
4. Pass aggregated hidden state through linear layer to get raw output scores
5. Convert raw scores into log-probabilities

## Experiments and Results

#### Evaluations

The models were simply evaluated based on accuracy. This is done by making predictions on the testing dataset and then comparing the predicted scores to the actual scores. The percentage of correct scores to the total predicted scores gives us the accuracy.

#### Results

|  |  |  |  |
| --- | --- | --- | --- |
| FFNN | | | |
| Hidden Layer Size | Training Accuracy | Validation Accuracy | Test Accuracy |
| 4 | 0.9862 | 0.5573 | 0.5760 |
| 32 | 0.9904 | 0.5427 | 0.5625 |
| 128 | 1.0 | 0.5688 | 0.5708 |

In each case, the FFNN was trained in 40 epochs, as this is when the scores typically leveled out and continuing training would lead to overfitting. The optimizer used was SGD with a learning rate of 0.01 and a momentum of 0.9. As seen in the above table, the hidden layer size had little impact on the accuracy of the network’s predictions. The training accuracy is high in all cases while the validation and test accuracy remain low. This suggests that the model is incapable of generalizing when introduced to new data. This remained true regardless of the size of layers or epochs used in training.

|  |  |  |  |
| --- | --- | --- | --- |
| RNN | | | |
| Hidden Layer Size | Training Accuracy | Validation Accuracy | Test Accuracy |
| 16 | 0.5034 | 0.4875 | 0.4948 |
| 32 | 0.5113 | 0.5344 | 0.5219 |
| 64 | 0.5198 | 0.5156 | 0.5406 |

In each case, the RNN was trained in 16 epochs, as this is when the scores typically leveled out and continuing training would lead to overfitting. The optimizer used was Adam with a learning rate of 0.0001 and there were 8 recurrent layers. Once again, we see that the layer size has a minimal impact on the accuracy of the model. This model, however, can generalize as it achieves similar or better accuracy when making predictions on unseen data.

## Conclusion and Others

I (Brigham Thornock) completed this project on my own. I am typically never a fan of projects that use starter code, as I feel I spend more time wrapping my head around the provided code than developing the project for myself. I also feel there was not enough direction provided in the assignment. It was presented as if the only thing that needed to be done was implement the forward function and a small hint that the other code could use some changes, but the extent of the changes needed was not straightforward. I would have preferred that the data preprocessing be the only thing already implemented, but that is based on what I already know going in so it may be different for others. Overall, I think the hardest part was just figuring out what we needed to do.