Assignment 2, Artificial Intelligence 2

Vyron-Georgios Anemogiannis 1115202000008

1 Introduction

In this assignment we are tasked with developing a sentiment classifier using feed forward Neural Networks.
The data set consists of IMDB reviews, (URL, Review, Score). One review is positive if it has an attached score ≥ 5, otherwise it is considered negative.
For this assignment we will use the PYTORCH, SKLEARN frameworks and GloVe embedings.

2 Data Processing

Immediately after getting the data from the tab separated file we start the regularisation process.

- We remove all the apostrophes.
  By doing this step, sentences like ”I didn’t like” become ”I didnt like” helping the model distinguish positive sentences ”I did like the movie” from negative ones ”I didnt like the movie”.

- We make all the characters lowercase.
  Since the program distinguishes upper case from lower case characters, it is preferable to only keep one of them. So in the word vectors now, we have the ascii codes of only lowercase characters, decreasing the possible combinations.
- We remove <br /><br />
  Scrolling through the reviews, we often find the above mentioned sequence. We simply remove it since it has nothing to do with the actual review.
- We keep only alphabetic characters
  Keeping only the alphabetic characters, gives us word vectors of 32 possible ascii codes.
- We remove the stopwords
  Those are words that appear often in texts, removing them will help the model distinguish words that acrually make an impact on the sentiment of the review.

3 Data Vectorisation Using GloVe

As requested by the assignment instructions, we use GloVe word embeddings in order to vectorize our reviews. For each word, if it is in the dictionary, we match it to a 300D word embedding. We believe that the greater the vector dimension, the better will be the representation for each word. Afterwards we take their mean in order to reduce the size of the data we have to pass to the model.

4 Creating the NN model

For this assignment we will test models with 1 or 2 hidden layers. We believe that since the Logistic Regression Model did a descent job calculating the sentiments, we don’t need to have a very complex model. In addition we have implemented dropouts in order to reduce over fitting. All layers are linear. The input layer has the size of the input and the output neurons are 2 to show the probability of a review belonging in each class.

5 Train and Test functions

For the loss function we use Cross Entropy Loss. It is the best option since both the outputs of the NN are probabilities of the review being positive or negative.

In the training function, for each epoch, we train the model with all out batches, and then test a validation set. If the result of the validation set, worsens from one epoch to another, we end the training and return the previous epoch. We basically implement early stopping.
The test function simply produces a classification report for the test set using SKLEARN's classification report function.

6 Neural Network Parameters Tuning

For the parameter tuning, we try various combinations of all the different hyper parameters we can use. In the selection of the possible values of the hyper parameters we use previous knowledge from the class Pattern Recognition and Machine learning where we had to guess the genre of 1 second music clips.

We start the testing with a 1 hidden layer model, using various powers of 2 as neurons for the hidden layer. The activation functions we test are the onew that gave the best result in the above mentioned assignment. The dropout rates are increased each time by 0.1, we don't believe a drop out rate more than 60% is reasonable. If that gave better results, then it would be better to just simplify the model. Same for the optimisers, we use the best ones from the previous assignments. Last we try various learning rates, starting small and going bigger and bigger.

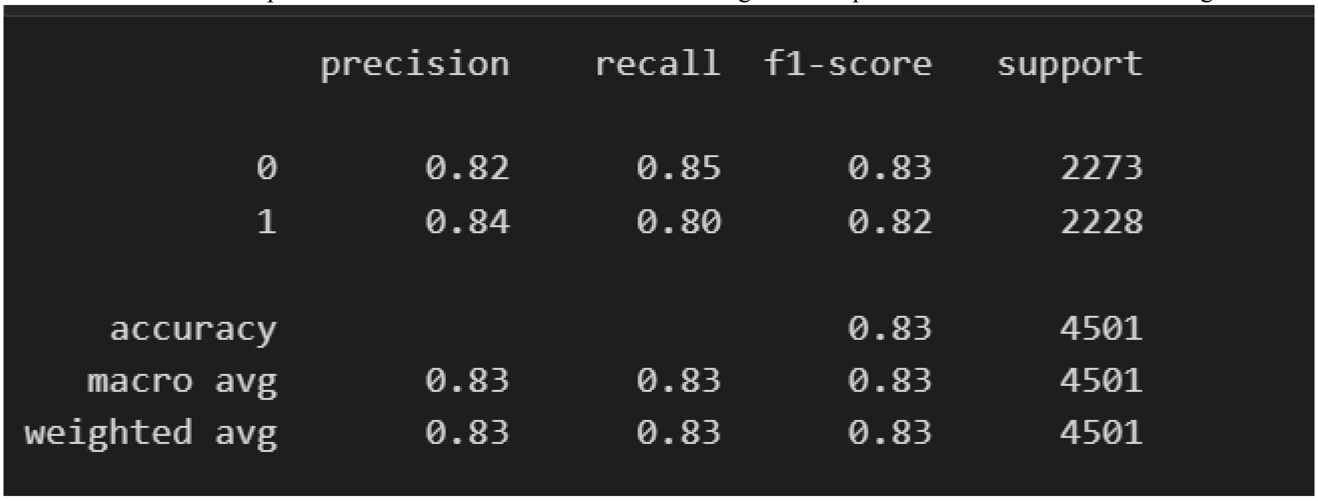
We see that the best results were given from the smallest neural network (with 1 layer and 128 neurons) and the Adam family of optimisers. The dropout rate and lr didn't make much difference. It is notable that a network with 2048 neurons and 1 hidden layer, also achieved this score. We will stick to the smaller models tho, since often simpler is better. We must also note that we were surprised that a learning rate scheduler, didn't make much difference tho this is most likely a result of the early stopping, not letting the model train for more than 6 epochs in average.

Afterwards we tried a NN with 2 hidden layers, using the best hyper parameters from before and only trying diffrent neuron combinations. This did not yield better results.

Last we tried some learning rate schedulers. Those were StepLR and ExponentialLR that also had the best performance in the previous assignment. Those also did not improve our scores.

7 Results

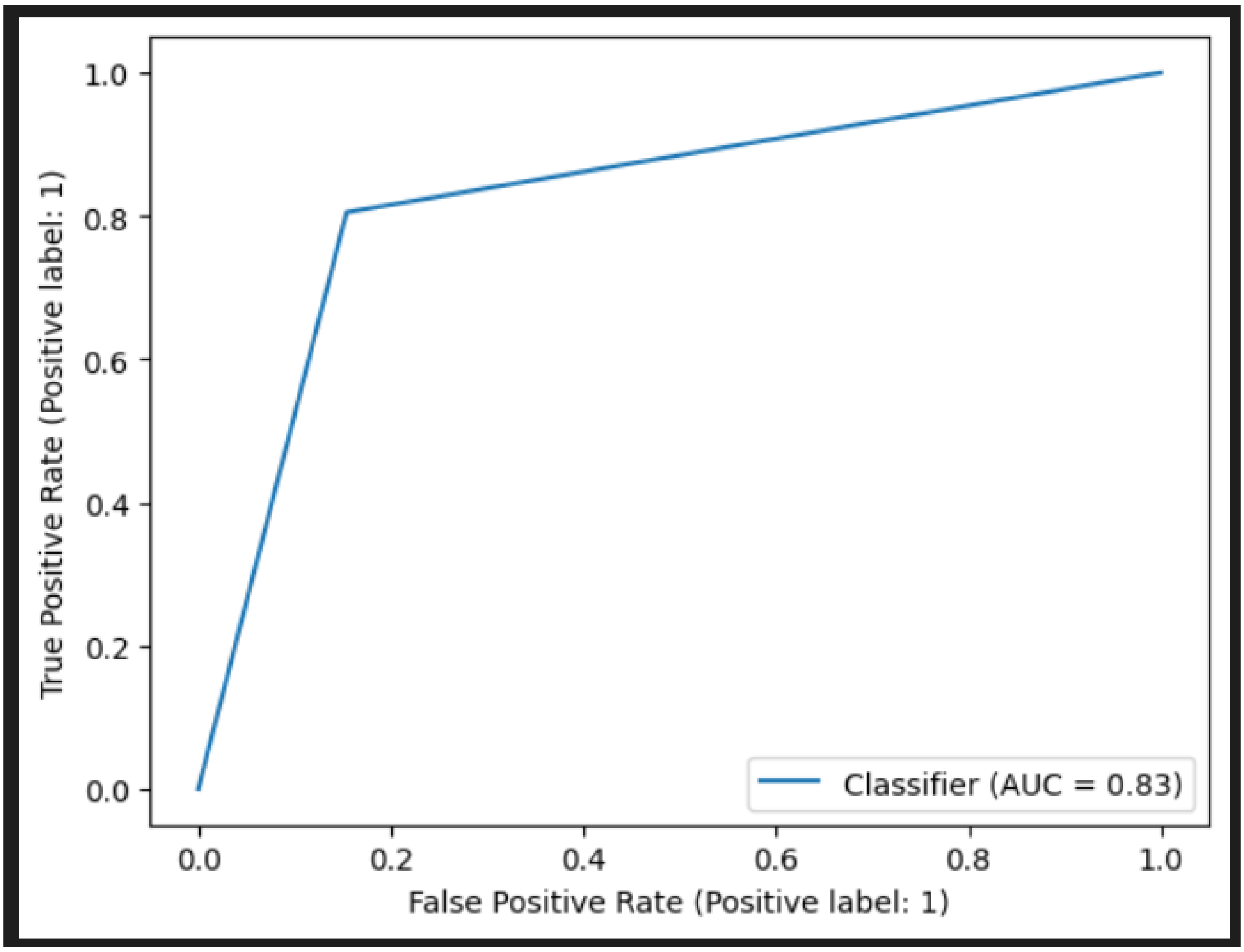
Please note that the results were done locally.
In the end we use one of the best models to present our final results.
We have a classification report that has all the metrics requested in the instructions. This gave us an average score of 0.83. We believe this could improve with TFIDF vectorisation but this assignment requires us to use GloVe Embeddings.



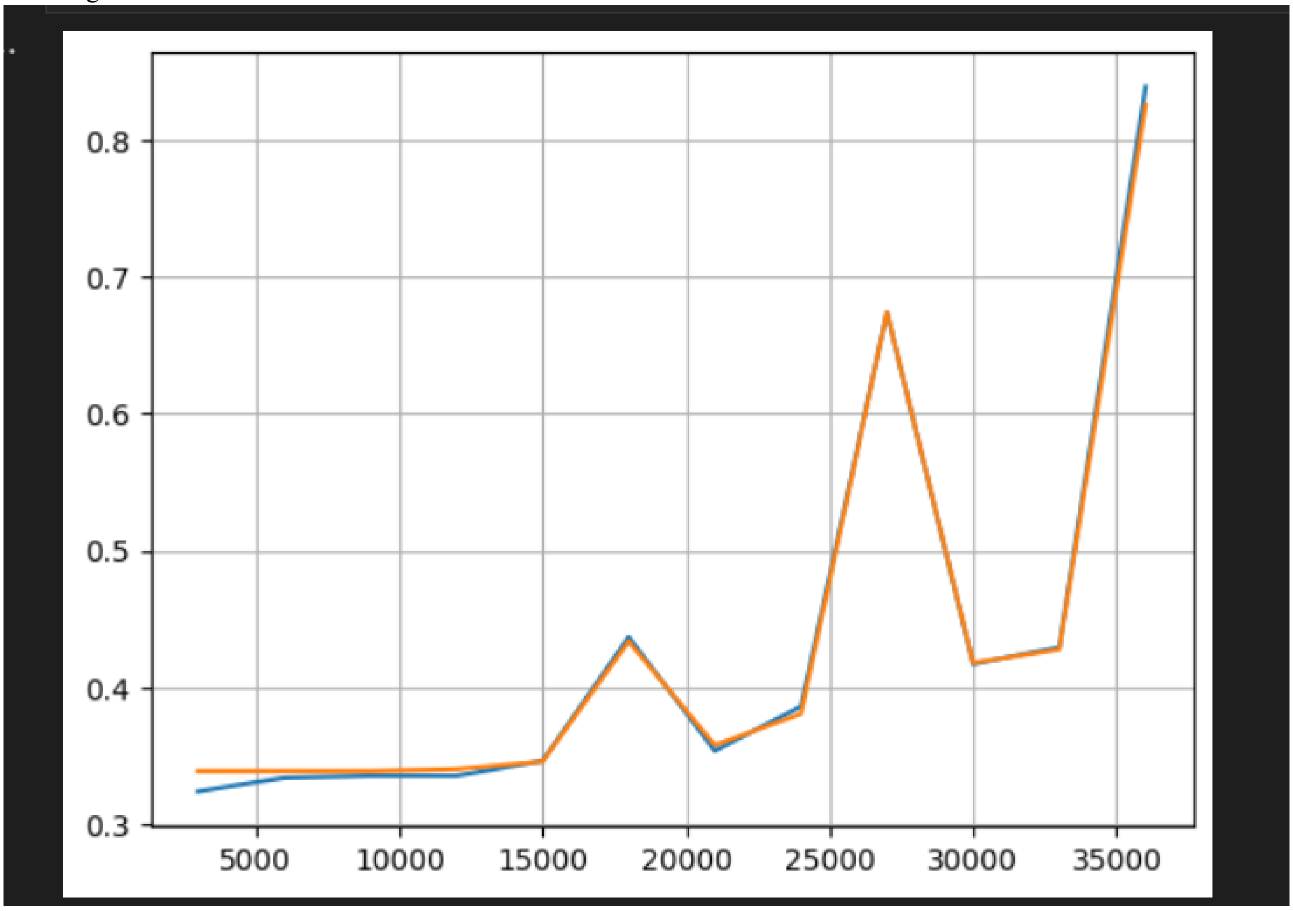
```
              precision    recall  f1-score   support

           0       0.82      0.85      0.83      2273
           1       0.84      0.80      0.82      2228

    accuracy                           0.83      4501
   macro avg       0.83      0.83      0.83      4501
weighted avg       0.83      0.83      0.83      4501
```

In addition we print the ROC curve and the learing Curve. The lines in the learning curve are as close to each other as shown in the graphics bellow due to the intense regularisation the earl stopping does. It will not let the model overfit.

ROC curve:



Learning Curve:



8 Test Set - What to run

For this assignment we have handed in 2 IPYNB files, HW2.ipynb and test.ipynb.
All the work was done on HW2.ipynb locally on our PC. This file is not compatible with google colab and cannot run there. Even tho you can't run it, all the cells have their outputs visible if you want to check them. In order to actually use the test set on google colab, we have uploaded the test.ipynb file. This file omits the model parameter tuning and by giving the train and test set path and running all the cells, you can test the model. Since the seed does make a big diffrence, if the results are not good, please try a few different seeds. This is something we cannot control. We have left there a seed that worked descently during the file creation but we don't have a test set to make sure we chose the right seed.

9 References Used or Studied

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
https://www.cs.toronto.edu/~lczhang/aps360_20191/lec/w06/sentiment.html
https://medium.com/analytics-vidhya/sentiment-analysis-for-text-with-deep-learning-2f0a0c6472b5
https://pytorch.org/docs/stable/optim.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.RocCurveDisplay.html