Artificial Intelligence II

IMDB Sentiment Analysis - Deep Neural Networks

Georgios - Alexandros Vasilakopoulos
A.M. 1115202000018

Data Preprocessing

- The entire preprocessing step is implemented within a custom-made vectorizer.
- Similarly to the first assignment, all alphabetic characters are converted to lower case, all non alphabetic characters are removed and the english stopwords are also removed.
- The inputs of the model are GloVe embeddings that are acquired as follows: each review is tokenized and, for each token that exists in the w2v vocabulary, the corresponding vector is fetched. The final vector corresponding to the review will be the average of the token vectors. It is logical to assume that the average of the token vectors represents the ”semantic average” of the words of the review.
- The custom vectorizer also standardizes the data and stores the mean and std values in order to standardize incoming test & validation instances. This step actually doesn't seem to affect the performance significantly.
- After many trials, it was observed that the maximum f1 score that could be achieved with the input of size 50 was about 0.78 on the test set. On the other hand, the size 300 input managed to reach f1 scores over 80 percent on the test set. Since the input of size 300 is very much affordable in terms of computation time, it was preferred over the other candidate sizes.

Model Options

- Generally, the total number of hyperparameters to consider in the making of a DNN can be large.

In this case, however, we can make a few assumptions to simplify the selection:

We observed in the 1st assignment that a simple logistic regression model can perform quite well in this classification task. This implies that the task itself is not very complicated. Therefore, we should avoid having too many layers in our network, because they increase (unnecessarily) the complexity of the model and there is a risk of overfitting.

Therefore, in order to keep the model simple, we will start by examining networks of one hidden layer and afterwords we will examine whether two-layered networks perform better.

- The hyperparameters to consider in a feedforward NN with one hidden layer are the following: hidden layer size, Optimizers, learning rate, Activation Functions. These hyperparameters can be evaluated by exhaustive (or at least, heuristic) searching.

Optuna

- In order to determine the optimal hyperparameter selection in the simple one-hidden-layered network, the optuna framework was used.
- The candidate optimizers that were examined were: Adam, Adamax, NAdam, SGD.
- The learning rate was picked from within the interval $[0.0001, 0.01]$. This generous interval examines both (relatively) small and (relatively) large values.
- The activation functions that were examined were ReLU, LeakyReLU and SELU.
- The size of the hidden layer was selected from the interval $[20, 128]$. Again, the idea is to keep the model as simple as possible.
- Since we only have one hidden layer of varying size, I found it unnecessary to use dropout in the hidden layer.

- The default sampler TPESampler was used in order to search for the hyperparameters

Evaluation of the Produced Model:

- The model produced by the searching algorithm is defined by the following hyperparameters:

```
learning_rate = 0.0004
optimizer = "Adam"
activation_function = "ReLU"
d1 = 121
```

- After plotting the learning curve[1] for the previous network, it seemed to be a bit unstable. After sequential epochs, the score fluctuated, but the overall trend was upward[2]. In order to avoid these unwanted fluctuations, I manually decreased the learning rate down to 0.0001. The learning curve [3] now seems to be much more stable, and the performance tradeoff is not that significant.

Neural Networks with more than one hidden layer:

- After following a similar process with optuna, using two hidden layers, we can conclude that the performance doesn't increase significantly with more than one hidden layers.
- Notice that, in this case, the learning curve [4] is unstable and it shows that the model overfits relatively quickly.
- Therefore, once again, in an effort to keep the model simple, we will avoid more than one hidden layers.

[1]plotted under "plotted under "Hyperparameter Selection" -> "Evaluation"

[2]The training algorithm that I implemented trains the network for a fixed number of epochs and then returns the model which scored the highest f1 score on the val. dataset

[3]plotted under "Learning Curve & ROC curve (Final Model)"

[4]plotted under "Hyperparameter Selection" ->"More than one hidden layers"

Final Model and Results on Test Set

- The final model is the same as the one produced by the searching algorithm for 1 layer, with the exception of the learning rate, which we decreased manually to increase the stability:

```
        learning_rate = 0.0001
          optimizer = "Adam"
    activation_function = "ReLU"
               d1 = 121
```

  Precision: 0.8305

  Recall: 0.8346

  f1 measure: 0.8325

- The learning curve[5] of the Final Model shows that the model does not overfit within 14 epochs and that the training and validation scores are close to each other.
- The form of the ROC curve implies a good performance and the auc of 0.83 corresponds to a high percentage of correct predictions .

[5]plotted under "Learning Curve & ROC curve (Final Model)"