cs577 Assignment 5

Sahil Riyaz Sheikh(A20518693)

Department of Computer Science
Illinois Institute of Technology

November 22, 2022

Abstract

In this assignment, I 'll work on two datasets, by building a binary class classifier for first dataset and multi-class classifier for the second dataset. I'll make use of embedding layers and glove to create a embedding matrix

1. Problem Statement:

I have been provided with two datasets. The first one being the IMDB dataset. I need to perform sentimental analysis on this dataset. It consists of 25,000 entries and I need to correctly classify whether the sentiment of the given entry is positive or negative.

For the second dataset, I'll work on Reuter Newswire dataset. This dataset consists of 11,228 newswires from Reuter. I need to create a multiclass-classifier to correctly classify the entire of the dataset to the corresponding class. This dataset consists of 46 classes.

2. Proposed Solution:

I will implement three models for each dataset, one consisting of Embedding layer, second one consisting of loaded Embedding matrix while using Glove and the last model will consist of LSTM layers. I'll compare the results of these models and will report my conclusion.

3. Implementation Detail:

For first dataset, IMDB, I loaded the dataset directly from keras.datasets using the load data () function. This made the loading of data a fairly simple process. Then I had to split the data into training, validation and test sets as well as I process the data by making use of pad_sequnces as mentioned in the class lectures. This is essential to set the dimension of the data. Then I created a moderately deep network for this dataset. I made use of Embedding layer. I had to make sure to keep the input dimension more than the vocabulary, I set input dim =10,000, output dim as 64 and input length as 300 because I had used 300 for max length in the padding sequence. I trained the model, plotted the graph, evaluate the model on test set and saved the weights. For this dataset my choice of activation function in output layer was sigmoid.

Now to incorporate Glove, I referred to the class notes and tried to follow the given notes. I was able to create embedding matrix and use in the model with the weights frozen. I recorded the performance through graphs, evaluated the model on test set and saved the weights.

To incorporate LSTM layer was fairly less complicated then previous task as I just needed to import the layer from keras.layers and add in into the model. I observed the performance on test set and saved the weights/

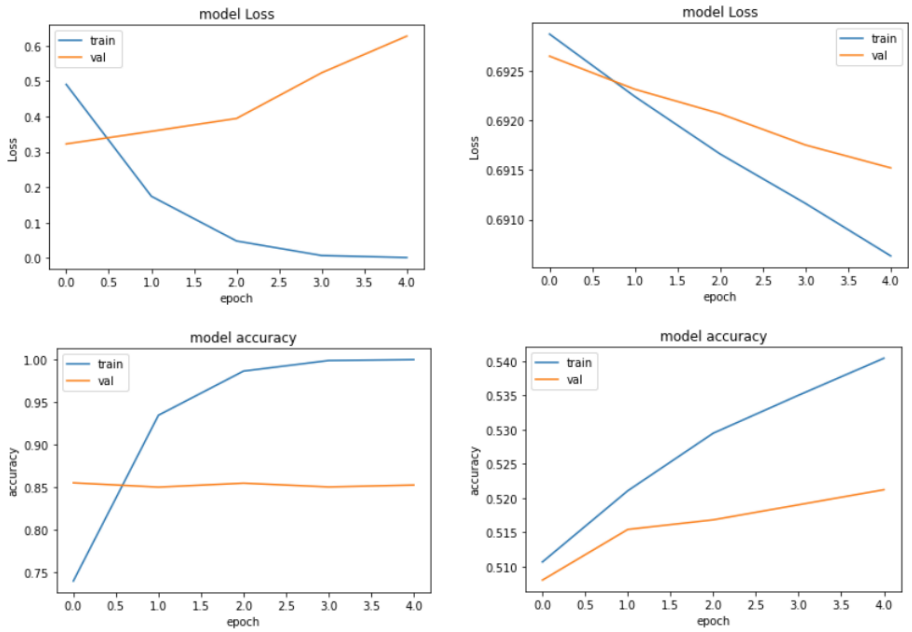
For the second data set, I was again able to load it directly from keras.datasets. I was able to one-hot encode the label/target of the dataset as they were present in numerical form rather than consisting of words making the first model was not a very challenging task but the next task was the one which I found out to be very challenging. As I had a reference for the IMDB dataset in notes, there was no reference for this dataset. I downloaded the dataset, untar it, extracted it but the dataset consisted of .sgm files which I have never worked on. I tried to find a way to convert them into text but was unsuccessful. I was stuck at this point. Then after searching for few hours I found a code on reference which I found a little helpful. This is the link: https://towardsdatascience.com/text-classification-in-keras-part-1-a-simple-reuters-news-classifier-9558d34d01d3 . For my perspective to create a embedding matrix, I needed a "text" holding all the text of the dataset. It was impossible to extract the text from sgm files. On further research I was able to make out the other thing which was essential to make the Embedding matrix was word_index of the dataset. I had converted the text of IMDB data to sequence, then tokenize them to create the word_index which made it possible to create the embedding matrix. For the link I was to create the

word-index without downloading the dataset and this it possible for me create the model with embedding matrix and record its performance and save its weights. The implementation of LSTM layer was less complex. I created two models one with only one LSTM layer and other with two LSTM layers. I used softmax as activation function in all output layer of all models for this dataset.

4. Results:

FOR IMDB:

Model with Embedded layer:

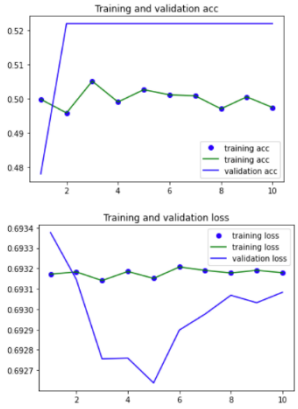


Both model were trained for:
Epochs = 5, Batch size = 128.
The model on left has rmsprop as optimiser and one on right as SGD as optimizer.
I can concur the model on left performs better than one right, eve thought left one over fits after 2 epoch but provides better results when considering loss and accuracy.
Evaluating on test set:

```
: 1 loss,acc = model1.evaluate(test_x,test_y, verbose=1)
  2 print('Test accuracy of model is ', (acc)*100,'%')
```

```
782/782 [==============================] - 2s 2ms/step - loss: 1.4475 - acc: 0.8308
Test accuracy of model is  83.0839991569519 %
```

Model with Embedded Matrix:

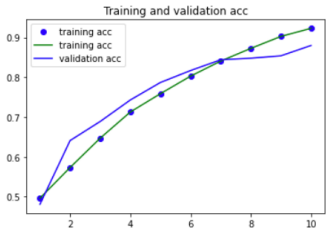


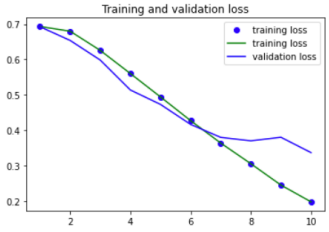
Model was trained for:
Epochs = 10, Batch size = 32.
The model overfits after 5 th epoch, but has a constant training accuracy and training loss. The doesn't perform abysmally but can definitely improve with further fine tuning. I used rmsprop as otimizer.
Evaluating on test set:

```
1 loss,acc = model_Em.evaluate(x_test,y_test, verbose=1)
2 print('Test accuracy of model is ', (acc)*100,'%')
```

```
125/125 [==============================] - 0s 3ms/step - loss: 0.6932 - acc: 0.4988
Test accuracy of model is  49.87500011920929 %
```

Model with LSTM layer:



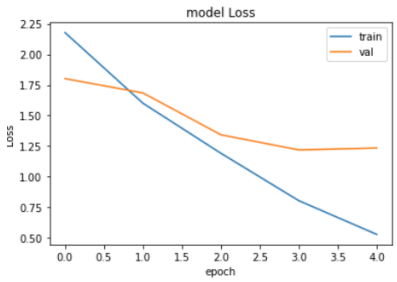


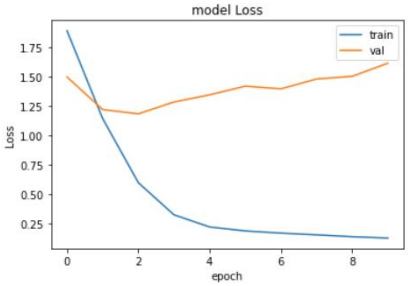
Model was trained for:
Epochs = 10, Batch size = 32.
This model doesn't overfit easily. It performs the best wen compared with the other models
Conclusions:
The model with embedded matrix has the worst performance of 49% whereas model with embedded layer reached a score of 83% and the model with LSTM was able to achieve an accuracy of 87%. All these results were recorded on test set. I can conclude model with LSTM layer perform better than the ones with othr layer. This is due to the fact LSTM has four gates which help to take care of vanishing gradient problem and long – term dependencies.

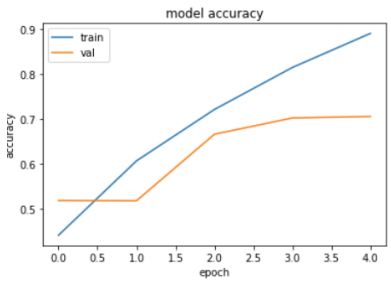
Evaluating on test set:

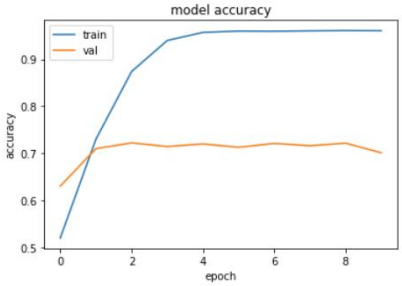
```
]:  1 loss,acc = model.evaluate(x_test,y_test, verbose=1)
    2 print('Test accuracy of model is ', (acc)*100,'%')

125/125 [==============================] - 1s 9ms/step - loss: 0.3800 - acc: 0.8712
Test accuracy of model is  87.12499737739563 %
```

For Reuters Dataset:
Model with Embedded layer:I created 4 models to view their performance.



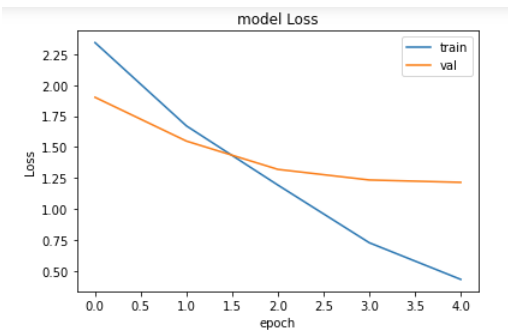


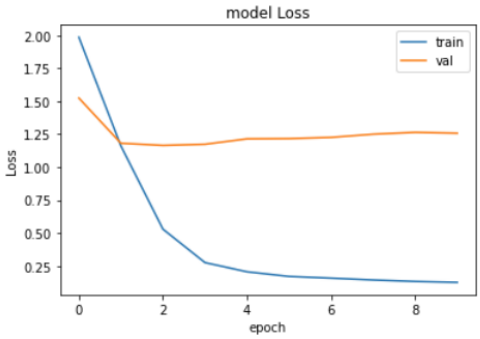


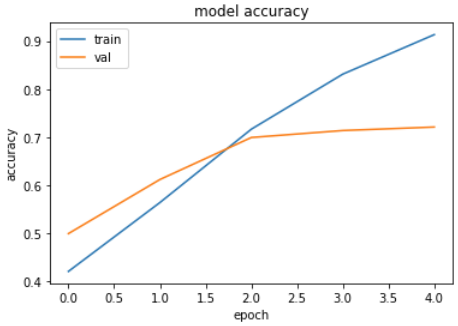


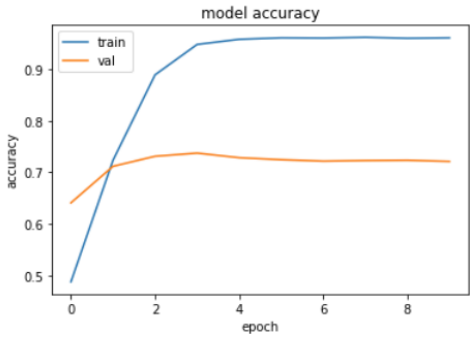
Model 1 and 2 (above) were trained for 5 and 10 epochs respectively. Batch size of model 1 was 128 whereas batch size of model 2 was 32. Both model made use of rmsprop as otmizer.

Model 3 and 4 (below) were trained for 5 and 10 epochs respectively. Batch size of model 1 was 128 whereas batch size of model 2 was 32. Both model mde use of adam as optimizer.





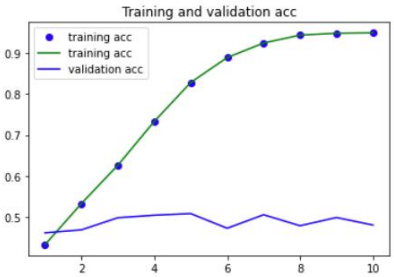




Their performances are quite similar but I selected model 4 as the better performing model as it has the graph with less deviations.
Evaluation on test set:

```
1]:  1 loss,acc = model4.evaluate(test_x,test_y, verbose=1)
     2 print('Test accuracy of model is ', (acc)*100,'%')

71/71 [==============================] - 0s 2ms/step - loss: 1.3946 - acc: 0.6968
Test accuracy of model is  69.67942714691162 %
```

Model with Embedded Matrix:



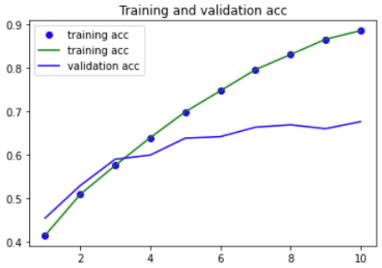
The model was trained for 10 epochs with a batch size of 32. The model achieves a good training accuracy and loss scores but it doesn't show much improvement in its validation accuracy as well as loss. Theo model also overfits after 3rd epoch.

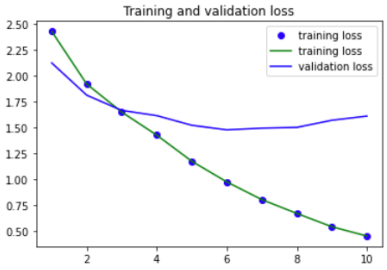
Evaluation on test set:

```
49]:  1 loss,acc = model1.evaluate(test_x,test_y, verbose=1)
      2 print('Test accuracy of model is ', (acc)*100,'%')

71/71 [==============================] - 0s 2ms/step - loss: 3.5838 - acc: 0.4800
Test accuracy of model is  47.996437549591064 %
```

The model achieves an accuracy of 47%. This is lower than what I expected. I could fine tune the model but I being to suspect Embedding matrix as suited for this type of task related to text classification.

Model with LSTM Layer:





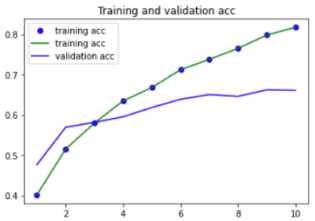

```
7]: 0
```

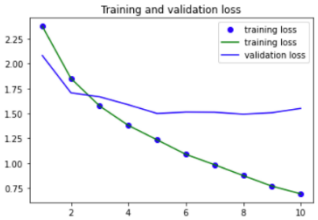
```
8]: 1 loss,acc = model3.evaluate(test_x,test_y, verbose=1)
    2 print('Test accuracy of model is ', (acc)*100,'%')
```

```
71/71 [==============================] - 0s 6ms/step - loss: 1.6868 - acc: 0.6549
Test accuracy of model is  65.49420952796936 %
```

The model above consists of 1 LSTM layer, I could see the significant improvement in performance when compared to the previous model.
Model was trained for 10 epochs with the default bach size of 32 and adam as optimizer.

Model with 2 LSTM layers






```
]: 0
```

```
]: 1 loss,acc = model2.evaluate(test_x,test_y, verbose=1)
   2 print('Test accuracy of model is ', (acc)*100,'%')
```

```
71/71 [==============================] - 1s 8ms/step - loss: 1.6617 - acc: 0.6273
Test accuracy of model is  62.73375153541565 %
```

:
Epochs = 10 batch size =32
Optimizer – adam.

Conclusion:
I can conclude the models with LSTM layer perform significantly better than rest. This is due the advantages of LSTM

The model with 2 LSTM layer witness a drop in accuracy by 3 % when compared to the model with one LSTM layer.

5. References:

Lecture notes
https://keras.io/api/layers/core_layers/embedding/
https://stackoverflow.com/questions/58352326/running-the-tensorflow-2-0-code-gives-valueerror-tf-function-decorated-functio

https://stackoverflow.com/questions/4674473/valueerror-setting-an-array-element-with-a-sequence
https://sahiltinky94.medium.com/word-embedding-glove-dd27f630c663
https://shiva-verma.medium.com/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e
https://keras.io/api/datasets/reuters/