Assignment 2
Nora Mirabal
10/12/2022
CPSC 393-01

Introduction

In this assignment I built 3 different neural networks. The first is a binary classification NN which predicts whether a book has over 4 stores or not. The model was built using both scikit learn and keras packages. The model performed poorly with a 60% accuracy score. I tried to make different alterations to the model to see if it would help the model perform better and overall it only changed the accuracy rate by at most 1 percentage in the best case scenario. The second is a multiclass classification NN which predicts which category of pages a book falls into. The category 4 means that a book has more than 1000 pages, 3 means that a book has more than 500 pages but less than 1001, 2 means that a book has more than 300 pages but less than 501, and 1 means that a book has less than 301 pages. The model has a

Analysis

I got my data from Kaggle (good-reads book). The data set contains detailed information about the books, the original data set has 12 variables and 10,352 books. The first thing I did was look over the excel sheet of the book data to get a better idea of what my data looked like. I cleaned the data by removing the isbn and the isbn13 variables which provided no useful information. The language variable also had to be cleaned because eng and en-US and en-GB were both meant to mean that the book's language is in english (changed 1408). I created the variables review_bool which converted the ratings of each book to a binary variable, with 1 meaning the book's average rating is equal to or greater than 4.0 and 0 is a book with an average rating of less than 4.0. Additionally, I created the variable language_bool which converted the language code of each book to a binary, with 1 meaning the book's language is English and 0 meaning the book's language was a language other than english. I created a variable pages_category which converted the number of pages a book has to one of 4 categories, 4 means that a book has more than 1000 pages, 3 means that a book has more than 500 pages but less than 1001, 2 means that a book has more than 300 pages but less than 501, and 1 means that a book has less than 301 pages.

Methods

<u>Binary classification example:</u>

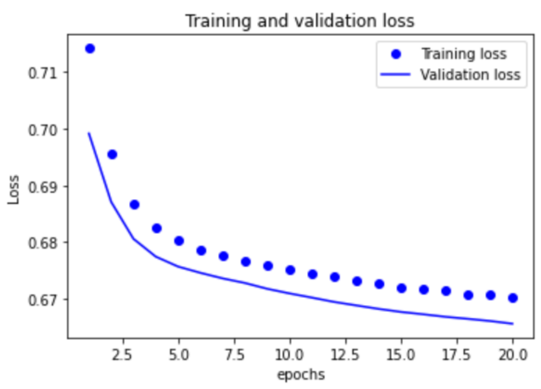
In this example I classified book ratings of books as above 4 or below 4 (1 or 0), based on the book's language, number of pages, the count of ratings, and the count of text ratings. The first thing I did was z-score all of my predictors so that they would be equally compared to one another and then I created a train and test split. I decided to go with 50% for the split to have equal sized train and test sizes. Next I made sure to convert my train and test sets to numpy arrays and convert all of the values within the array to float tensors. After I cleaned and formulated my data for the model, I created a Sequential model and added 3 layers to the model. The first layer (visible) has 16 nodes and uses the relu activation function, the second layer (hidden layer) has 16 nodes as well and also uses the relu activation function, and the third layer (the output layer) has 1 node and a sigmoid activation function so it will output either a 1 or 0. I chose 16 nodes because I didn't want my model to be too computationally expensive but I still wanted enough nodes in order to determine complex patterns. I then compiled the model with the rmsprop optimizer, binary cross entropy loss because it is the best option because our output is probabilities, and accuracy metrics. All three of these metrics are packaged as a part of Keras. To monitor during training the accuracy of the model I created a validation set by setting aside 1,000 which is roughly 10% of my data.

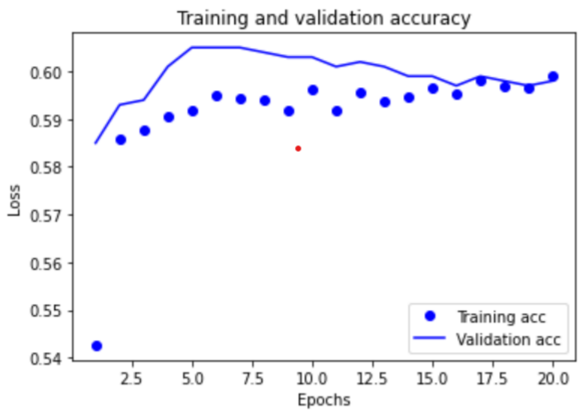
<u>Multiclass Classification example:</u>
The multilayer classification predicts a categorical variable therefore I used the pages category variable I created. For this example I decided to see if the model would work well on guessing the category of different page number categories using the book's language, the average rating of a book, the total number of reviews and the text review count. The first thing I did was z-score all of my predictors so that they would be equally compared to one another and then I created a train and test split. I decided to go with an 80% training set and 20% test set for the split. Next I made sure to convert my train and test sets to numpy arrays and convert all of the values within the array to float tensors. I created a Sequential model and added 3 layers to the model. The first layer (visible) has 16 nodes and uses the relu activation function, the second layer (hidden layer) has 16 nodes as well and also uses the relu activation function, and the third layer (the output layer) has 5 nodes because I have 5 categories and uses the softmax function. I chose 16 nodes because I didn't want my model to be too computationally expensive but I still wanted enough nodes in order to determine complex patterns and because I had few categories to choose from I didn’t need any more nodes. I then compiled the model with the rmsprop optimizer, mean squared error loss, and mean average error metrics. All three of these metrics are packaged as a part of Keras. To monitor during training the accuracy of the model I created a validation set by setting aside 1,000 which is roughly 10% of my data.

<u>Regression example:</u>
The regression example is different from the first two examples because instead of predicting a single discrete label it predicts a continuous variable. For this example I decided to see if the model would work well on guessing the number of total reviews a book has using the book's language, the average rating of a book, and the text review count. The first thing I did was z-score all of my predictors so that they would be equally compared to one another and then I created a train and test split. I decided to go with an 80% training set and 20% test set for the split. Next I made sure to convert my train and test sets to numpy arrays and convert all of the values within the array to float tensors. I created a Sequential model and added 4 layers to the model. The first layer (visible) has 128 nodes and uses the relu activation function, the second layer (hidden layer) has 128 nodes as well and also uses the relu activation function, the third layer (hidden layer) has 128 nodes, and the fourth layer (the output layer) has 1 node. I chose 128 nodes because I didn't want my model to be too computationally expensive but I still wanted enough nodes in order to determine complex patterns. I then compiled the model with the rmsprop optimizer, mean squared error loss, and mean average error metrics. All three of these metrics are packaged as a part of Keras. To monitor during training the accuracy of the model I created a validation set by setting aside 1,000 which is roughly 10% of my data.

<u>Results</u>
<u>Binary classification example:</u>
Next I trained the model for 20 epochs (mini-batches) of 512 samples and monitored the accuracy on the 1,000 samples I set aside. In order to see everything that happened during training I used model.fit() to

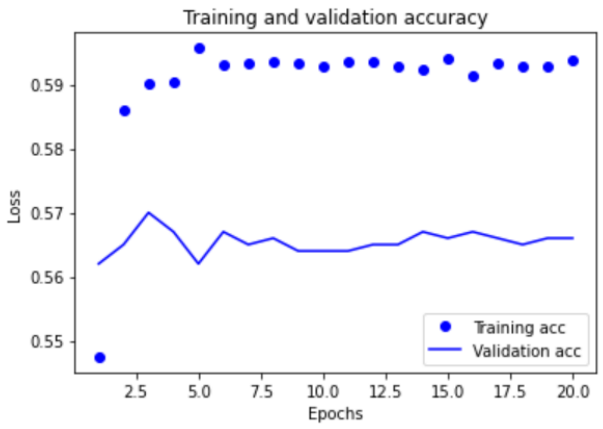
create a history object. Then to see the output of the validation loss and accuracy visual I created the two graphs below using matplotlib.

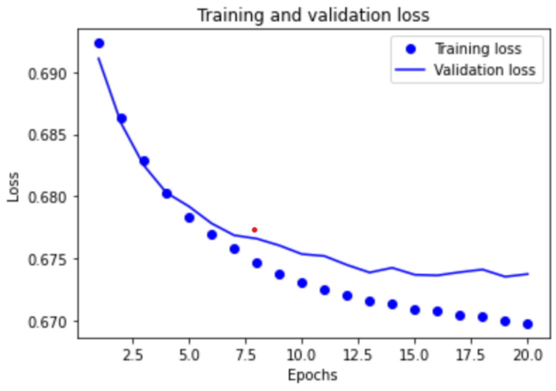




As you can see the epochs training loss decreased with every epoch run and the training accuracy increased as well. In both the accuracy and loss graphs you can also see that the training and validation set are parallel to each other and don't move in the opposite direction. This is great news because it means that our training set isn't extremely overfit and should work well with data that it hasn't seen before. Because of this I did not adjust the epoch size. The final results for my model were .67 for the test set and .594 from the train set. That means that this model only achieves a 59.4% accuracy which is not ideal for a model and would likely not be implemented for any sort of predictions.

<u>Multiclass Classification example:</u>
Next I trained the model for 20 epochs (mini-batches) of 512 samples and monitored the accuracy on the 1,000 samples I set aside. In order to see everything that happened during training I used model.fit() to create a history object. Then to see the output of the validation loss and accuracy visual I created the two graphs below using matplotlib.





I settled on training the model with 7 epochs based on the graphs because it seemed the most optimal. After adjusting the model with 7 epochs the accuracy was 50%. The model only was able to predict the category of number of pages a book had 50% which is not ideal.

<u>Regression example:</u>

I trained the model for 20 epochs (mini-batches) of 512 samples and monitored the accuracy on the 1,000 samples I set aside. In order to see everything that happened during training I used model.fit() to create a history object. I evaluated the model then using mean squared error and mean absolute error. The mse was 29968158720 which is horrible and a mae of 29103 which means that for every output the model is 29103 reviews off. Overall this model works terribly and should not be used to predict the number of reviews a book may have.

Remarks

Binary Classification Further Experiments:

1. I completed the first further experiment by adding 1 extra hidden layer to the model, so as opposed to 3 complete layers there are 4. After running the new model the accuracy of the model was only .5% better. I tried adding a 5th layer to see if the model would continue to gain accuracy and it did in fact lead to an accuracy of 61% which showed that adding layers to the model leads to a higher accuracy model.
2. The second experiment I tried was by tweaking the number of nodes in each layer to 32 instead of 16 from the previous model. And found that the accuracy results were once again better. Sometimes adding more nodes leads to the overfitting of the training set which ultimately leads to less accuracy however in this case it made our model better because the additional nodes didn't lead to overfitting instead it increased the accuracy to 60%.
3. The third experiment I tried was changing the loss to mean squared error instead of accuracy. This led to the models accuracy decreasing by .2% which tells us that there isn't a significant change between mse and acc.
4. The fourth experiment I tried was changing the activation function from relu to tanh. Tanh was the activation function used in the early uses of neural networks and as expected we can see why. The accuracy once again declined from 59% to 58% because of the function.

Multiclass Classification Further example:

I wasn't able to get the further examples to run correctly because it kept giving me an error that my x and y values were of a different length but I included the code in my google colab regardless. Regardless, for some reason this model is more intricate to code than the regression classification.

Regression example:

I wasn't able to get the cross validation code to run however I included the code.

Overall, all of the models had subpar accuracy but I don't blame the models, the main reason is most likely because the variables in the dataset are linearly correlated therefore to predict one based off of all the others is nearly difficult for th emodels.