Assignment One MultiLayerPerceptron Model (Newsgroups DataSet)

DTSC13-301

BOND UNIVERSITY

BOND BUSINESS SCHOOL

Muiz Murad

13599863

Table of Contents

Executive Summary

The task was to build a model that had a high testing accuracy but did not overfit or underfit. The desired outcome was to have a model with a small number of parameters, high testing accuracy but a balance between its training and validation loss. The process in which to find the best model was based off a logical but sequential method in which each model was refining the model built before it to eventually find the best fit. The problems faced within the task are based around infrastructure and resource availability, easier and more processing intensive methods are not able to be used as the machine being used is not able to process the code quickly and efficiently but rather take several hours and become a more exhaustive and time inducive method. This will be overcome by using a manual Grid Search technique in which the user will manually refine the model based on logical and educated thinking and methods. The final and best model to be used will be a Dropout Model with 256 nodes, 20 epochs, batch size of 480 and dropout rate of 0.1.

Problem Definition and Data Assembly

The input data for the model was the 20 Newsgroups data set which is a collection of approximately 20 thousand newsgroup documents split up evenly across 20 newsgroups. It is organized in 20 different newsgroups and each group is linked to an associating topic. The aim is to predict the accuracy of the model to classify the newsgroup with its associating category. The problem at hand is a supervised learning problem as we are categorising data. We also know the output of the data which further reinforces the supervised learning.

Evaluation Protocol Decision

As we have a large dataset it is reasonable to maintain a hold-out validation set. With the hold-out method we simply split the data up into testing and training sets to provide to randomized sample groups. We then train the model on the training dataset and validate the model on the testing set. The model evaluation technique we are using is the accuracy metric in the keras package. We can also use the loss function in keras to help us guide our optimization of the model. The loss function is a method of evaluation that tells us how well the algorithm can build a model from the given data. However, if the loss function is too large then it would indicate that the predictions are too far off from the actual result. For this dataset we will use the categorical cross entropy loss function as we are trying to classify data to one specific category out of many categories.

Preparation of Data

The data needs to be formatted as tensors and scaled to small values ie. [0,1]. As our data doesn't take value in different ranges, we do not need to normalize it.

Development of Model To Beat Baseline

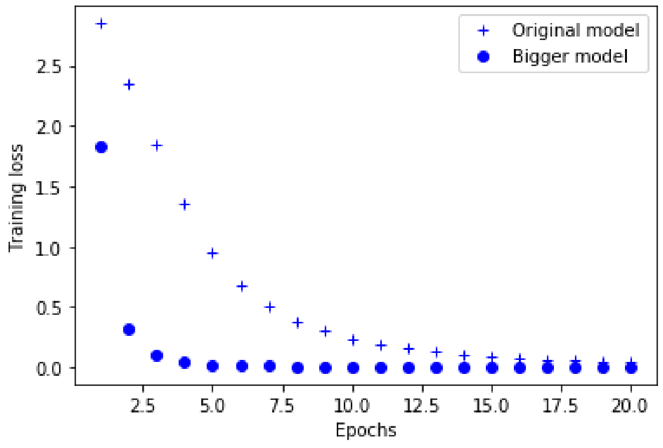
This is a crucial part of the process as we determine the best last-layer activation, loss function and optimization configuration for which the model will run. As we are dealing with a multi-class, single label classification problem we will need to use the soft max as our last-layer activation and categorical cross entropy as our loss function. We use softmax as a it will return probabilities of each target class over all possible target classes. It also summated all probabilities to one and calculated probabilities can easily be interpreted as they are between the range of 0 to 1. Our optimizer will be adam as it is the easiest to use and combines the most valuable properties of the exisiting RMSProp and AdaGrad optimizers allowing for it to use an optimization fitting process even with noisy data.
To make a model to beat the baseline we will use a simple three-layer mlp model that will have a larger set of parameters. As we can build a model that can beat the baseline, we know it's likely that the inputs are useful in predicting the outputs.

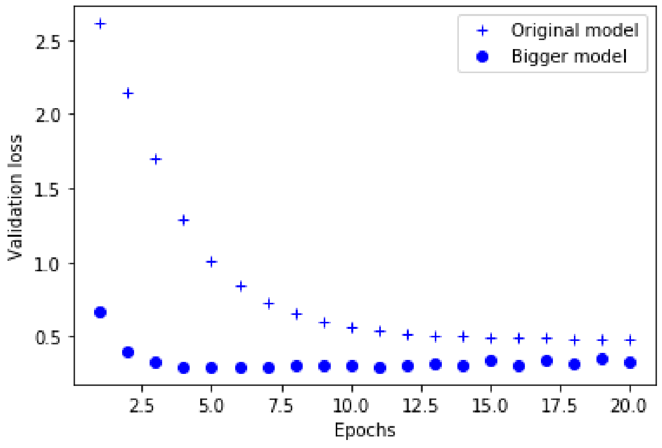
Scaling Up: Developing A Model That Overfits

For this part of the process we need to investigate what parameters provide us with a model that has a low validation accuracy but high train accuracy. This would mean that the model is starting to recognise the random error in the data rather than the relationship between the 20 categories and 20 thousand newsgroup sets. We can do this by three easy pointers:

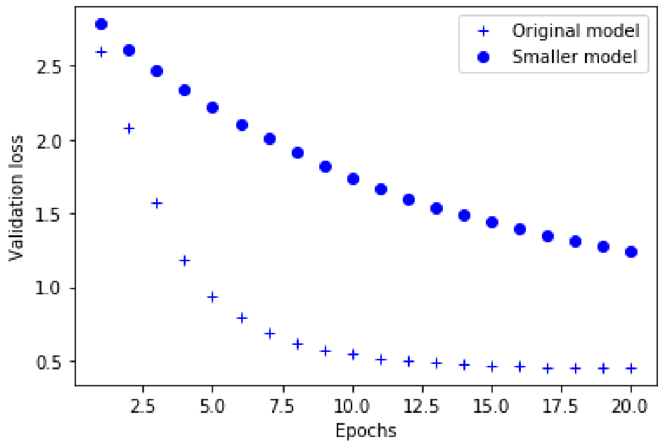
1. Adding Layers
2. Enlarging Layers
3. Increasing Epochs

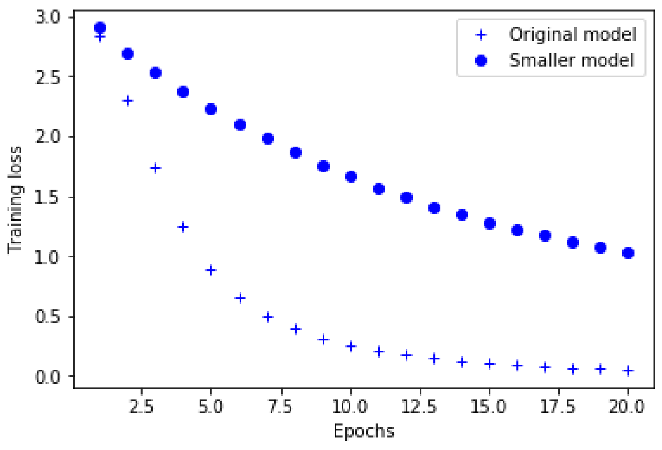
We can use the plots from our code to help us explain how the model was overfitted and describe any patterns or trends observed.



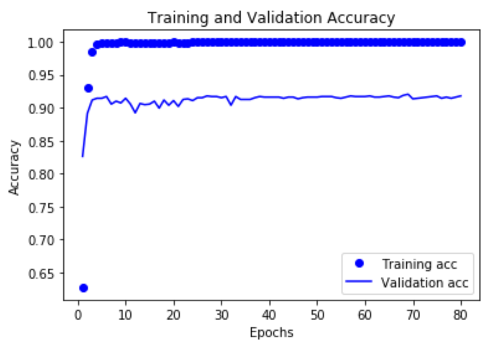


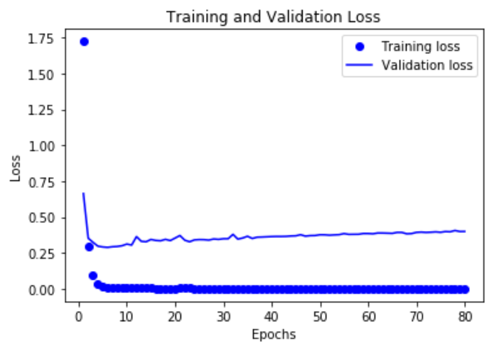
From the plots above we can see that the training loss is less than the validation loss indicating overfitting for our larger model. However, when we look at our smaller model, we can tell it is underfitting as the training loss is much larger than the validation loss as we can see below.





As we now have two distinct models with large and small parameters we are able to start looking inbetween the two model's parameter ranges to find the best model that is the best balance between over and under fitting. It also needs to be determined when the plots start converging as this is a good indication of when the model has reached its optimal point This will be done by increasing the number of epochs to a significantly large amount, this can be seen in the plots below.



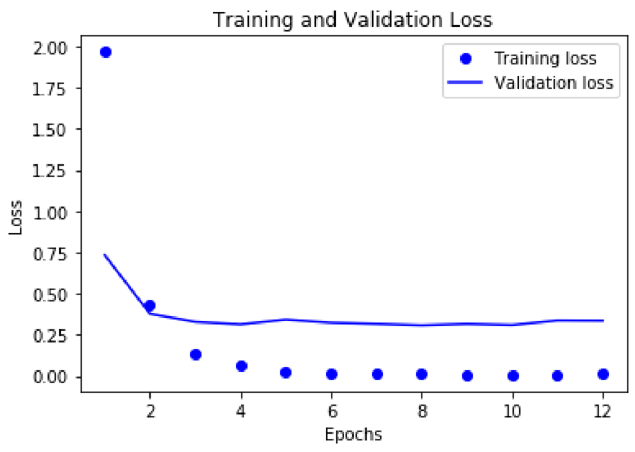


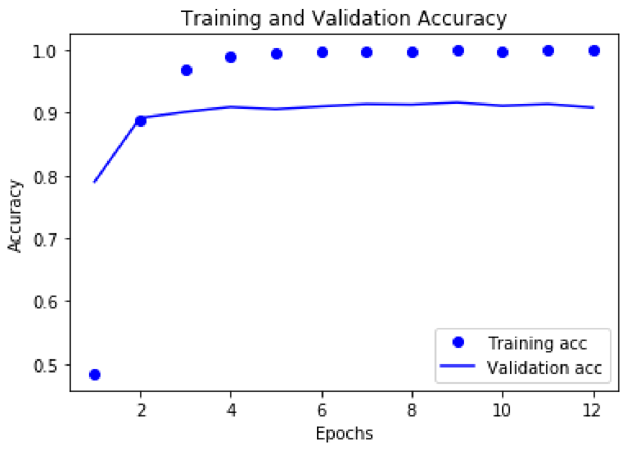
It can be clearly seen that when the number of epochs has been increased to 80 the model is flattening out extremely quickly. We can also notice that the training loss is far less than the validation loss by a significant amount and is an indication that the model is overfitting a fair bit. Another noticeable point on the graph is that the model's turning point is around the 10-epoch mark where it is flattening out constantly from the decline before it. This is a good indicator for us to decide about where we will be stopping the number of epochs at, to prevent a large overfitting as seen above. We can also draw the information from the smaller and bigger models to tell us that at 20 epochs the plot starts to plateau hence combining these two points together we can make a sensible decision to keep our epochs between 10 and 20. We also need to note that using a batch size of 480 has provided a consistently high validation accuracy and hence we will continue to use this throughout our process. Although we should be using a batch size that is a power of 2 as CPU and GPU memory architecture is often organized in powers of 2. However as for consistency we will remain with a batch size of 480 which lies close to 2^9.

We have also introduced a dropout rate to our model, this was not applied to all models as a comparison was needed between a model with and without a dropout layer. The dropout values are optimal usually between 0.2 and 0.5 as a high dropout rate will under learn the data and too little of a dropout rate will lead to overfitting. They are better for larger networks and hence we will be using it in our models with a larger node size.

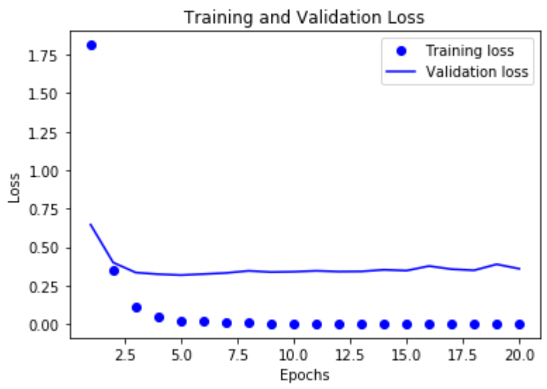
Regularizing Model and Tuning Hyperparameters

For this part of the process we need to start looking at the validation accuracy more closely as start to fine tune the model in detail. We will need to change architecture, add/remove layers and try different hyperparameters. We need to start looking for clear changes within the loss and accuracy plots where the training and validation plot starts to come closer to each other indicating that the model is becoming balanced. We need to also note that as we are manually doing this, it will not be as accurate as using a Grid Search Cross Validation Technique in which the model with automatically be fitted with provided values and validated to find the best model.

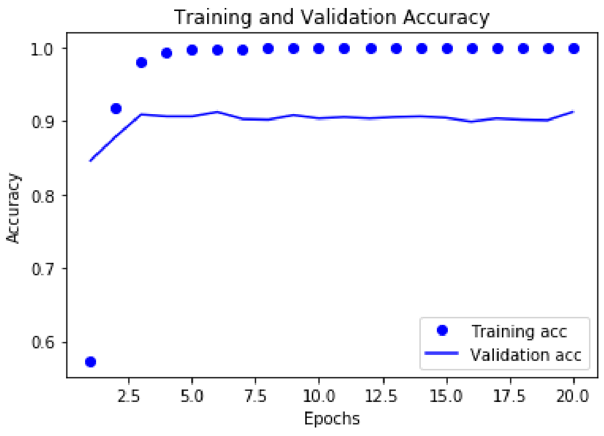




For the model above we have used 1024 nodes, 12 epochs, 480 batch size and 0.4 dropout rate. Upon fitting the model, we need to note that we receive a high validation accuracy indicating the model is being fit well with the data however there is still a difference between the training and validation loss. But we can see a smaller a gap between these validation and training losses indicating that the model is being refined better and reducing its overfitting. We now reduce our number of nodes and dropout rate to help us reduce our total number of parameters and balance out our training and validation losses. The model below represents a dropout model with 512 nodes, 20 epochs, 480 batch size and dropout rate of 0.1.

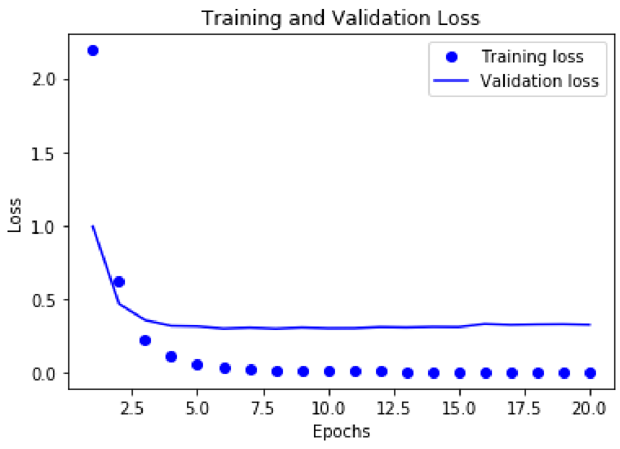


We can see that we have reduced our training and validation slightly and this can easily be seen by the reduction in the y-axis markings. As we have also increased our epochs we can see that that losses are also steadying out in a more smoother formation.

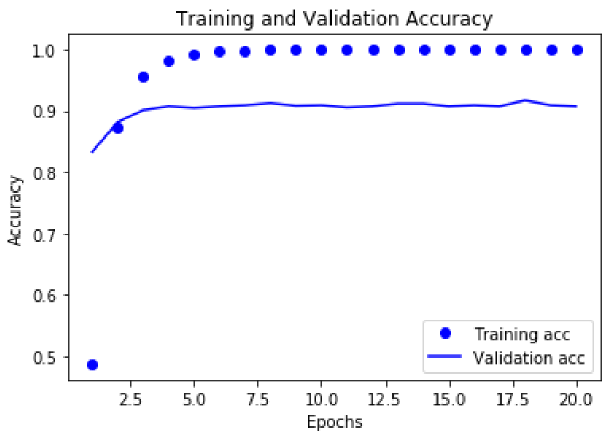


The validation accuracy has also increased slightly indicating that the model is performing better on the holdout set and the reduction in the training and validation loss is also indicating a more tuned model with a better balance.

We can now say that we have deduced a model that is providing good results however has too many parameters, we will now try and reduce the total number of parameters whilst still trying to keep the validation and training losses closer together and keeping the accuracies high. We will simply reduce the node size to 256.



We can see the model has performed well and has been able to hold the validation and training loss close together and provide a decent balance indicating there is still a bit of overfitting however a lot less than we started. We can also note the validation accuracies are higher from the diagram below informing us that our model has performed well and achieved a validation accuracy around 90-91%. The testing accuracy was 81.48% which in comparison to the validation accuracy is good and hence we have built a model that is optimal based on our process.



Conclusion

Our final model was with parameters of the following:

Epochs = 20
Nodes= 256
Batch Size = 480
Dropout Rate = 0.1
Total Parameters = 2 631 188

Based off the hardware and infrastructure constraints we were not able to build the most optimal model by checking the model over thousands of times with different parameters allowing us to choose the best model based on cross validation. This limited our optimization of the model to a manual grid search in which we had to logically build and recognize changes within the model to gradually work our way to a refined model that was not overfitting or underfitting but had the right balance. This was done through analysing the training and validation losses. Another key part was to determine when the model was converging and steadying out to ensure we were not building models with large parameters and making an exhaustive process. The max words was set at 10 000 to ensure that we had a large amount of data to train the model with to ensure that when the model was put to use it would be able to determine a wide variety of newsgroups and be able to link them to the most suitable category more accurately.