

**ASSIGNMENT COVER SHEET**

**Student Name: Christopher Slattery**

**ID Number: B00092939**

**Course: Bachelor of Science (Honours) In Computing**

**Year: 4**

**Lecturer: Stephen Sheridan**

**Title of Assignment: Build a Multilayer Neural Network Using TensorFlow Library**

**Due Date: 14/04/2019**

**Date Submitted: 14/04/2019**

The material contained in this assignment is the authors original work, except where work quoted is duly acknowledged in the text. No aspect of this assignment has been previously submitted for assessment in any other unit or course.

Signed: Christopher Slattery Date: 14/04/2019

**Introduction**

The objective of this task is to build a multilayer neural network using Google’s TensorFlow library. The goal is to design and build a neural network for a classification problem. The network will try to solve the problem by adjusting its weights. The data that the network is to be trained on is already provided in the form of two Microsoft excel files with data entries. The Training data has 4000 examples, 2 inputs and one hot encoded label. The Evaluation data has 1000 examples, 2 inputs and one hot encoded label. There will be 2 phases involved in this objective. The first phase is the constructing and training of the network. This phase will involve extensive training of the network by changing the value of 3 variables to see whether they have a good or bad effect on the accuracy of the training. This phase also involves trying to reduce the overall error and the time it takes to train the network. The second phase is the network evaluation phase. This phase involves using the output model from the training network and evaluating it against the evaluation dataset.

**Network Construction and Training**

The network construction and training phase consisted of me experimenting with 3 different variables in the python training script to see what effect they have on the accuracy of the training. These variables were NUM\_HIDDEN (The number of hidden/middle layer nodes), LEARNING\_RATE (The speed that the network learns at) and NUM\_EPOCHS (The number of times the network will run through the target data). These variables have a big impact on how effectively the network trains. When the variables have been adjusted, the network is run ten times to get an average from the ten times instead of just using one pass. The first change I made to the variables was to set the learning rate to 1.5, hidden layers to 10 and epochs to 2000. Figure 1 below shows the results for ten runs. It also shows the figures that the variables were changed to.

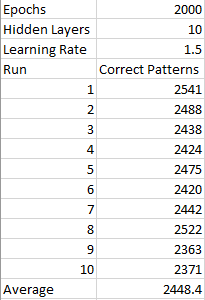


Figure 1

As the target data has 4000 entries and these parameters give an average of 2448.4 predicted patterns correct, it is safe to say that these parameters are not optimal. I decided to bump up the learning rate parameter to 5 and used the same parameters for hidden layers and epochs. As you can see from figure 2 below, the patterns predicted have jumped up the average to just under 1 thousand more than when the learning rate is at 1.5.

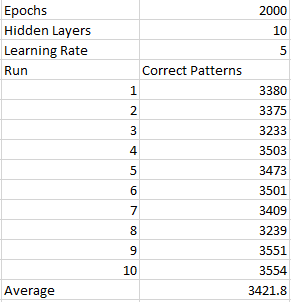


Figure 2

Figure 3

Figure 3 above shows a line graph for the number of patterns predicted correctly across ten different runs with the parameters from figure 2.

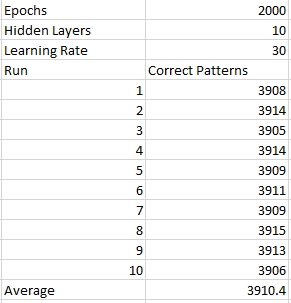


Figure 4

The next thing I did for the training phase was change the learning rate to 30 and do a further ten runs through the training data. figure 4 above shows the average for the ten runs and the parameters used. The average here bumped up to 3910.4. As there are 4000 data entries and the average Is 3910, you can see that the network is getting closer to optimal training. I continued this trend by incrementing the learning rate by 5 and keeping the other two parameters the same.

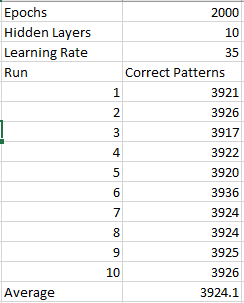
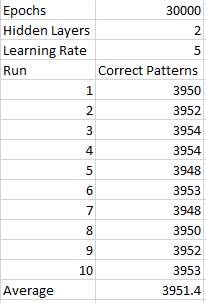


Figure 5

Figure 5 above shows the average when the learning rate is set to 35. As you can see, the network is getting closer to predicting the optimal prediction rate for the data set. There is now a problem though as setting the learning rate too high can cause the network to do unsatisfactory training and become unstable. If the network is doing this, it is very possible that we are getting false values being produced by the network.

To combat this problem, the number of epochs will have to be made higher but again with the epochs, if the number of epochs is too high. The model could succumb to overfitting which will also show the wrong values that are output from the training.

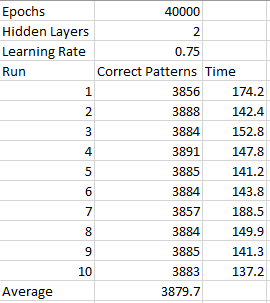


Figure

Figure 6 above shows the average of ten runs when the epochs are set to 30000(30 epochs), hidden layers set to 2 and learning rate set to 5.

Figure

The line graph from figure 7 shows the average of ten runs with the parameters from figure 6. As you can see, the results are getting closer to 4000. Even though this is getting pretty good results, it is time to move the learning rate down as its more appropriate to have a learning rate between 01. And 0.75.



Figure

Figure 8 above shows the learning rate set to 0.75, hidden layers to 2 and the epochs at 40000(40 epochs). The average is pretty good for these parameters.

Figure

Figure 9 is a line graph plotting the averages for the afore mentioned parameters. After trying different learning rates below 1, I decided that 0.75 would be optimal for what I am trying to achieve with the network. Once I had the optimal parameters, it was time to move onto the network evaluation phase.

**Network Evaluation**

After deciding on optimal parameters for the evaluation, it was time to save the model to disk and run the model on the network evaluation.

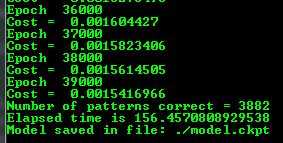
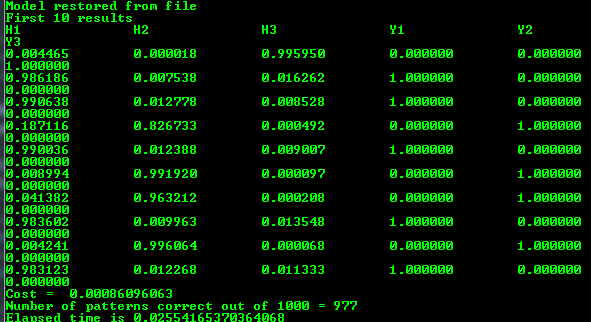


Figure 10

Figure 10 shows the number of patterns predicted correct after 40 epochs. The number of predicted patterns is quite high and should yield a good percentage when evaluated with the evaluation script.

Next it was time to run the evaluation. The image below shows the results of the evaluation. As you can see from the image, the network is predicting 977 out of 1000 values. As it is only 23 from the target, I think it is acceptable to use these parameters for the training network to help with the speed the network trains at and the accuracy that the network Is able to achieve.



Figure

**Conclusion**

To conclude, the model received two inputs, one hot encoded label and 2 hidden layers. The network was trained using 4000 examples in the training dataset that came from a csv file. The model performed well when training with different values for different parameters but mostly failed to be what I would have considered to be optimal. Because of this, further experimenting was conducted in order to find the optimal parameters for the network. The optimal parameters were then found to help the assignment be successful.