Group 23 Report

Andrej Erdelsky — Dean Polimac — Mateja Zatezalo — Lucas Fatas

5 March 2021

1 Introduction

This report discusses the creation of an artificial neural network (ANN) which is classifying different products based on their input parameters.

2 Single Perceptron - Logic Gates

Figure 1 shows the rate of errors over epochs for logic gates and, or, and exclusive or, respectively. As it can be seen in the figure, the perceptron is perfectly capable of predicting the result of an and gate, as well as the or gate, but it requires two epochs more for the latter. None the less, it is unable to predict the value for the exclusive or gate as it relies on the step function. This is because if we were to plot all the possible results of the exclusive or gate on a graph, we could not separate them with a single line.

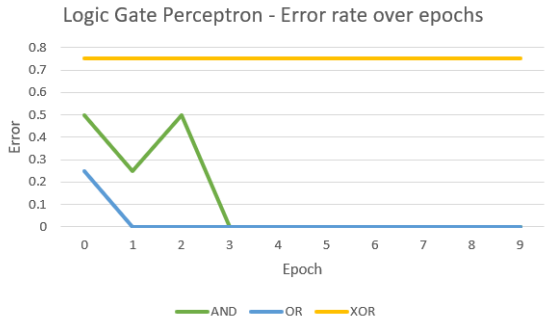

Figure 1: Errors per Epoch for the Logic Gate Perceptron

3 Architecture

3.1 Input neurons

The ten input parameters that the ANN is provided with are used to classify a products into one of seven possible types of products. In the input layer we will then have ten neurons, where each neuron represents a single input variable.

3.2 Output neurons

For the output layer the number of neurons will be with respect to the number of types into which we want to classify the products. Since the number of types is seven, we will have seven output perceptrons in the last layer.

3.3 Hidden neurons and layers

The number of hidden neurons and layers was initially arbitrary. Without doing any thorough research we opted for a single hidden layer containing 17 neurons, which is equal to the sum of input and output variables. Later this was changed during the optimization process.

3.4 Activation function

For simplicity we decided to use the sigmoid function as the activation function. We apply the sigmoid function to each predicted result of the perceptron and return it as a final result.

3.5 Diagram of network

The graph represents the neural network in its initial state. Starting from left to right, it shows the number of perceptrons in the input, a single hidden and an output layer, with 10, 17 and 7 perceptrons in each respective layer.

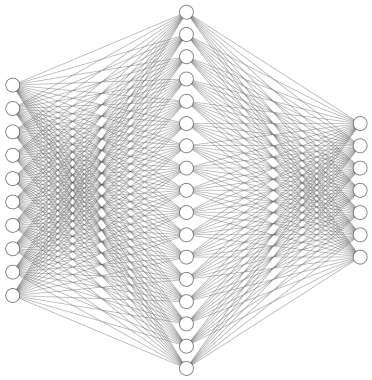

Figure 2: Representation of the neural network

4 Training

4.1 Dividing Data

The division of data is done by splitting it into the training batch, and the testing batch with the ratio 85:15. Cross validation is used to split the training data in order to reduce the chances of over-fitting, with k-fold validation where k is set to 10.

4.2 Evaluate Performance

In order to measure the performance of the neural network, we settled for the accuracy as the performance metric. Every time the predicted value equal to the expected value, the accuracy increases by a factor of one over the overall size of the test sample.

4.3 Stopping Training

While we fine tuned the hyper-parameters we realised that it was always best to stop training in between 10 and 20 epochs. The performance after this many epochs started to decrease steadily, as the neural network starts to over-fit on the training data. In the end the optimal number of epochs was set to 15.

4.4 Different Weights

Applying different weights to the network containing a single hidden layer with 23 perceptrons multiple times showed that when all the perceptrons have the same initial weights, the network struggles to learn. This often led to accuracy between 12 and 25 percent depending on the starting weight, for weight zero it was always within the bounds of 12 to 14 percent, as for weights such as one, two, etc. it deviated between 16 and 25 percent. On the other hand, when the network had perceptrons with random weights with the random seed 30, the accuracy peaked with approximately 89 percent.

5 Optimization

During the optimization process the network is trained four times, each time with different number of neurons: 7, 15, 23 and 30. These neurons were put in a single hidden layer. The results show that the performance increases as the number of neurons increases, it peaks at 23 and then drops. The probable reason for this increase, and then decrease is that as more neurons are added, the model becomes more complex which allows it to make a better predictions, but when the model gets too complex it begins to over-fit.

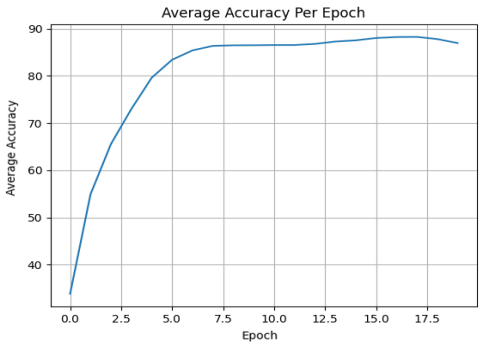

Figure 3: Best Performance With Different Weights

5.1 Best Result

As shown from the optimization process the best results were achieved using 23 neurons in a single hidden layer, combined with alpha that is equal to 0.001 as it provided the best performance for our network. This can be seen in Figure 4, the performance increases as training continues then at some point plateaus, and then finally decreases again as it begins to over-fit.

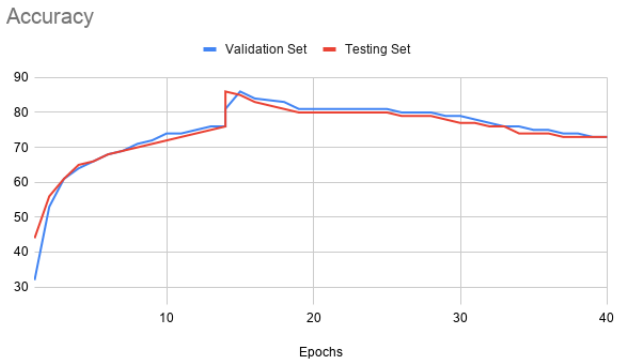

Figure 4: Validation and Test set Performance over Epochs

6 Evaluation

6.1 Success Rate

As mentioned in section 5.1 with 23 hidden neurons it peaked at a success rate of 89 percent accuracy on the test set. During other test - with a fixed 15 epochs

- it reached in between 70-89 percent accuracy as displayed in Figure 4.

6.2 Confusion Matrix

Every entry in the matrix represents a percentage of incorrectly predicted values by our network. As shown in Figure 5, our network made the most mistakes in entry [7,4] with 4.665% of incorrectly predicted values.

0	0.3343	0.3343	1.866	0.0848	0	0
0	0	0	0.0848	0.0848	0	0
0.334	1.7811	0	0.5089	0.5089	0	0.2545
0	0	0	0	0	0	0
0	1.0178	0.0848	0.7634	0	0.0848	0.0848
0.1696	2.8838	0	1.6964	1.4419	0	0.0848
0.0848	2.8838	1.6964	4.665	0.0848	0	0

Figure 5: Confusion matrix of the test set

7 Scikit-Learn

Using the optimized hyper-parameters from the notebook, our neural network learns much slower even though the alpha is larger by a factor of ten, but there are fewer than in our neural network. Another flaw is that the accuracy seems to be around 51 percent when using these parameters, which also falls short to the accuracy of our network that is in range from 70 to 89 percent.

8 Discussion and Conclusion

The neural network has best performance in terms of raw accuracy when the alpha hyper-parameter is set to 0.001, with a single hidden layer that contains 23 perceptrons, and runs over 15 epochs. The cross-validation technique used has contributed highly to the accuracy as it decreases the chances of the network over-fitting. Another important attribute that contributes to the performance - as seen in section 4.4 - is the random weights per perceptron. This allows each perceptron to train differently than the other perceptrons. Even though this may initially lead to lower accuracy, over time it should yield better results as the tests have shown. A few ways of increasing the accuracy might be tweaking the hyper-parameters and the alpha value, as well as changing the number of layers and perceptrons per layer. To conclude, the overall accuracy as well as the speed of the neural network are surprisingly good, and have far exceeded any expectation.