Assignment 3 - Report

CSE2530 Computational Intelligence

1. Develop (program) a single perceptron first. Show With three graphs (error over epochs) that the perceptron is able to learn the OR and AND function but not the XOR function.

We decided to merge the three graphs in the graph below. As we can see, the perceptron converges to an error of 0 after 5 epochs for the OR and AND functions, but the error stays at around 0.25 for the XOR function since it doesn't manage to learn for this function.

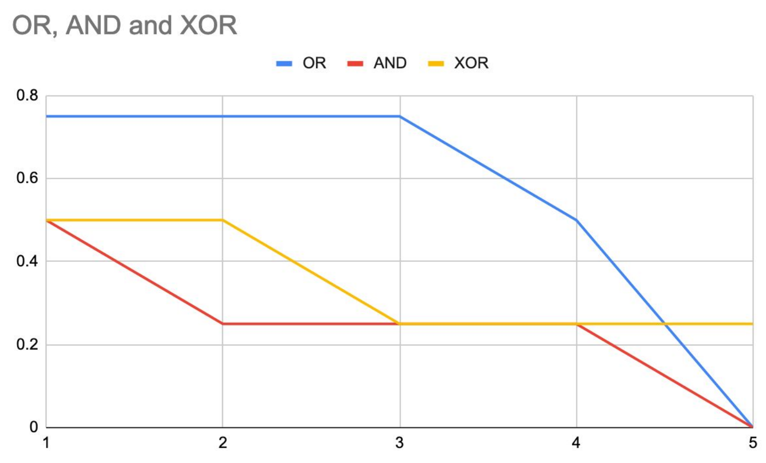


Figure 1
Graph representing the error over epochs for a perceptron for the OR, AND and XOR functions.

An ANN is much more powerful than a single perceptron. You will now design the topology of your network. For every question, briefly discuss how you arrived at your answers.

2. How many input neurons are needed for this assignment?
10, since there are 10 features for every input item.

3. How many output neurons do you require?
7, since there are 7 classes in which the given input can be classified. (7 outputs)

4. How many hidden neurons and layers will your network have? (Give an initial guess, later you will try a different number of hidden neurons and analyse the network's performance).
One hidden layer with 15 neurons is required on the initial count.

5. Which activation function(s) will you use?
We will be using Sigmoid and ReLU (rectified linear) as our activation functions. The main advantage of using Sigmoid is that it doesn't blow the activation function. The main advantage of using ReLU is because, in practice, networks with ReLU tend to show better convergence performance than sigmoid.

6. Give a schematic diagram of your complete network.

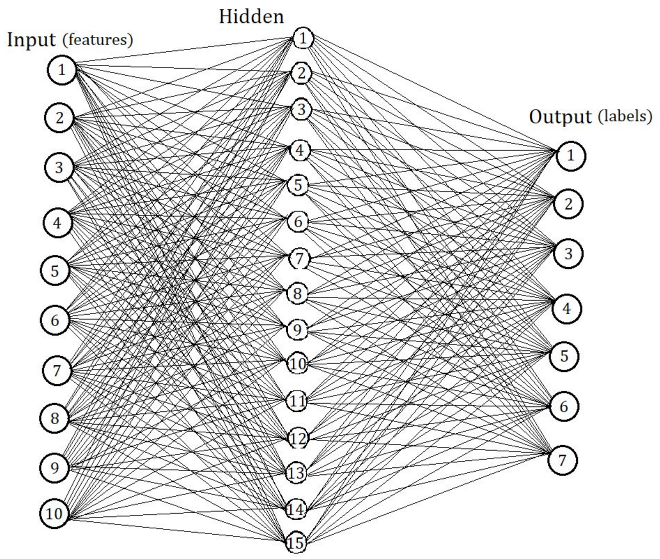


Figure 2

Schematic overview of our complete network

2.2 TRAINING

7. How and why did you divide your data into a training, validation and test set?

According to this article,[1] it's easier to validate and tune models with few hyperparameters. Leading to it being acceptable to reduce the size of the validation set.

We will divide the data into 70% training data, 20% validation data and 10% test data.

8. How do you evaluate the performance of your network? Briefly justify your answer.

By judging how it performs on the test set. Since this set is filled with previously unseen data we can see how it performs on new data.

9. When and why do you decide to end the training?

We end the training when the validation error reaches its minimum (if we would stop training when the training error got to its minimum → overfitting)

10. Train your network 10 times, each with different initial weights. How does the initialization impact the performance?

According to the 2 plots below, we can see that the initial weights do not change the eventual accuracy and mean square error of the network. The network will converge to (approximately) the same values independently of the initial weights.

[1] Shah, Tarang. "About Train, Validation and Test Sets in Machine Learning." *Medium*, Towards Data Science, 10 Dec. 2017, <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>

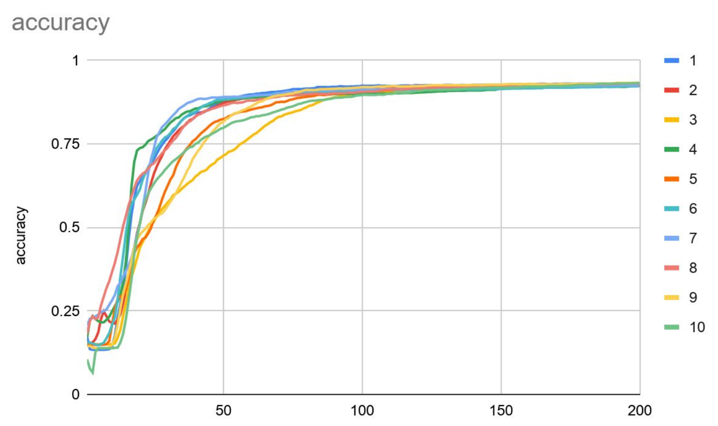


Figure 3
The accuracy of the network depending on the initial weights.

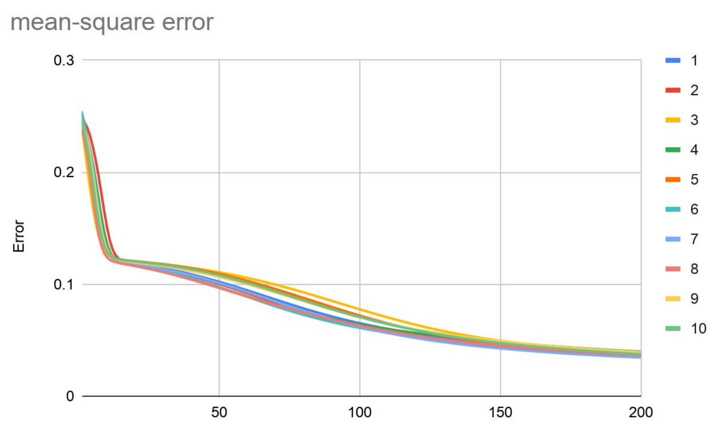
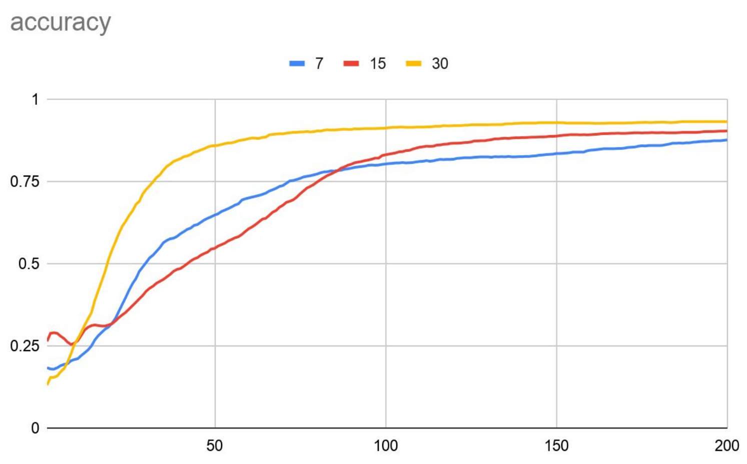


Figure 4
The mean squared error of the network depending on the initial weights.

2.3 OPTIMIZATION

11. Train your network with different amounts of hidden neurons. At least 3 times chosen within the range of 7-30 hidden neurons. Generate a plot of the final performance versus the number of hidden neurons in the network. Explain what you observe and what might be the cause of these observations.



From this graph, we can see that having 30 hidden neurons is the optimal choice while a number of 7 hidden neurons is the worst in the long-term. This is potentially because using less than 30 neurons is too few neurons to properly detect the signals in the dataset and can result in underfitting.

Figure 5
The accuracy depending on the amount of neurons in the hidden layer

12. Pick the architecture with the best result and show a plot of the performance of the training set and the validation set during training, across epochs. Justify your choice.
We chose the architecture with 30 hidden neurons as this gave us the best performance. In Figure 6 below, there is a plot below of the performance of the training set and the validation set during training, across epochs. As one can depict from the plot, as the epochs increase, the accuracy tends to increase with it for both training and validation set.

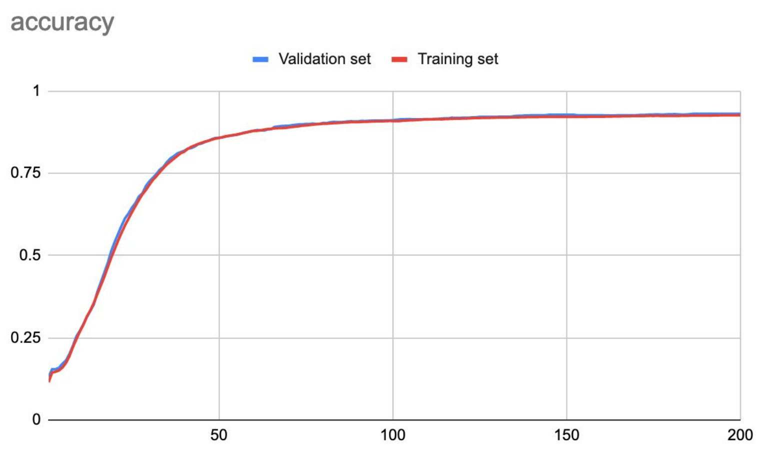


Figure 6

The accuracy of the network depending on the number of epochs.

2.4 EVALUATION

13. What is the success rate of your network on the test set? How does it compare to the results of the validation set?

Correct rate is 93.95% of the test set and 92.74% for the validation set.

14. Show and discuss a confusion matrix of your test set. How should it be read? Where did your network make the most mistakes?

The confusion matrix is the error matrix where the rows & columns of the matrix represent the instances of the predicted and actual classes, respectively. The amount of correct and incorrect predictions are shown with the count values which are broken down into classes.

0.914	0.000	0.037	0.000	0.000	0.000	0.049
0.000	0.959	0.010	0.000	0.000	0.010	0.020
0.038	0.000	0.900	0.000	0.050	0.000	0.013
0.012	0.000	0.000	0.964	0.000	0.012	0.012
0.000	0.000	0.022	0.011	0.957	0.011	0.000
0.000	0.010	0.000	0.030	0.000	0.950	0.010
0.022	0.000	0.022	0.022	0.000	0.011	0.925

$$\text{ACC} = \frac{TP + TN}{TP+TN+FP+FN} = 0.938$$

2.5 SCIKIT-LEARN

16. Download and open the Jupyter notebook. Tweak the settings of the grid search, and run all cells to find optimized parameters, according to scikit-learn. Are the values of these parameters different from the ones you chose? What differences can you see in the training behaviour and performance of the scikit-learn network, compared to yours?

Different parameters:

learning_rate_init:
Theirs: 0.01, Us: 0.0001

Activation functions:
Theirs: Only sigmoid, Us: Relu in hidden and sigmoid in output layer

hidden_layer_sizes:
Theirs: 0.01, Us: 0.0001

Validation_fraction:
Theirs: 0.127, Us: 0.2

17. Finally, take the optimal hyperparameters found by scikit-learn and plug them into your own network. Does this give you better performance? Why do you think these parameters are better / worse?
We plugged in the optimal hyperparameters found on scikit-learn and upon plugging them into our network, we got an accuracy of 19%. We think the performance is worse because our implementation is different and we don’t have certain elements like tol, solver, batch size etc. in our code.