Work 2: Neural Networks

Computational Intelligence

Nikita Belooussov Roberto Ariosa

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Barcelona, November 25, 2021

1 Description of the runs with the different configurations that you have performed.

We ran 18 different network configurations. The possible configurations can be seen in Figure 1. The configurations that were tested were specifically asked for in the assignment.

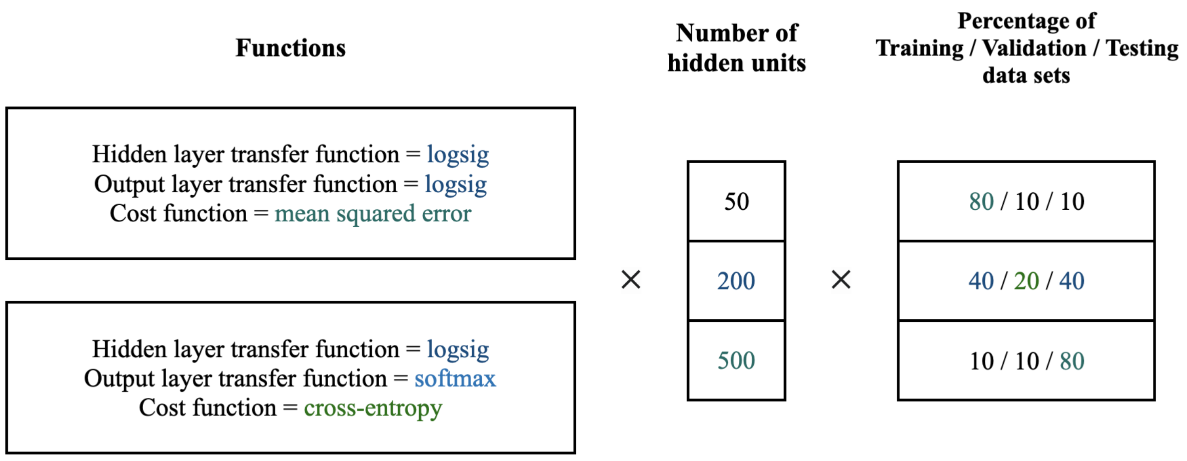
Functions
Hidden layer transfer function = logsig
Output layer transfer function = logsig
Cost function = mean squared error
Hidden layer transfer function = logsig
Output layer transfer function = softmax
Cost function = cross-entropy
×
Number of hidden units
50
200
500
×
Percentage of Training / Validation / Testing data sets
80 / 10 / 10
40 / 20 / 40
10 / 10 / 80

Figure 1: Neural Network Configurations

The main change is the shape of the output layers transfer functions, which is how the input to the layer is mapped to the output of the layer, and the number of hidden units. Next is the cost function, in which we tested two different popular cost function types. The equations for them can be seen below:

$$\text{MSE} = \frac{\sum_{i=0}^{n}(y - y')^2}{2}$$

$$\text{Cross-entropy}(y,p) = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix} \begin{bmatrix} log(p_1) \\ log(p_2) \\ \dots \\ log(p_n) \end{bmatrix}$$

An important thing to note for this is that mean squared error (MSE) is a popular cost function for regression type of problems, while Cross-entropy is a popular cost function for classification problems. The hidden units is the number of neurons per hidden layer. Clearly, the more neurons there are in a layer the more complex the system becomes. The last configuration that is changed is how the given data is divided between training, validation, and testing data sets. This shows the percentage of the total data set that each category gets. The training data set is used to train the neural networks. The validation data set is used estimate the performance of the neural network on new data, preventing overfitting. Lastly, the test data set is what is used to actually test how well the neural network is performing.

2 Explain how you have selected the rest of parameters.

There are several parameters that could be tested when trying to optimize the network, such as *minimum gradient*, *learning rate*, etc. The ones set specifically were the *divideFcn* and *trainFcn* and the *minimum validation checks*. The *minimum validation checks* to stop the training process was set to 6, this was to try to make sure that the solutions would not stop at the first local minimum. There is still a possibility that there is a better solution, and this would be overcome by setting the *minimum validations* to a higher value. The *trainFcn* was set to 'traingdx'. This training function allows for the neural network to adjust its learning rate, which is why it was not set for this report. Lastly, the *divideFcn* was set to 'dividerand'. This was because the data sets given were already sorted. As result it is necessary to randomize the data when splitting it, otherwise it is possible that the network would receive a data set with imbalanced output classes. The rest of the parameters were set to default. One reason is because the train function would change them, as in the case for the learning rate. The *minimum gradient* could also have been changed, but during training the minimum gradient was never reached before the number of validations was met. The other reason is that we assumed that most of the default parameters would be fine, since they are typically set to be functional in a wide set of scenarios.

3 Those tables that you consider necessary to describe the results obtained for the different network configurations. Explain and reason the results presented in the tables.

An important note to make about our results is that for each test of the configurations, the data set selection of the data set was randomized. While this will not affect the outcomes made based off of the results of these runs, future tests will likely produce results with different values. The main reason of why this not a concern is that the data set is fairly large and the configuration was tested multiple times with different data sets.

The results can be seen in Table 1. The table shows the mean accuracy of the different configurations tested. The MSE configuration refers to having the cost as MSE, the hidden transfer function as logsig, and the output transfer function as logsig. The Cross-entropy configuration is when the neural network was set to having a cross-entropy cost function, logsig hidden layer transfer function, and a softmax output layer transfer function.

Table 1: Parameter Values Tested

Data Split(Train, Validation, Test)	Number of Neurons	Configuration	Accuracy(%)
80% 10% 10%	50	MSE	28.84
		Cross-entropy	55.98
	200	MSE	35.91
		Cross-entropy	53.02
	500	MSE	36.37
		Cross-entropy	53.71
40% 20% 40%	50	MSE	28.27
		Cross-entropy	48.19
	200	MSE	35.41
		Cross-entropy	50.94
	500	MSE	35.62
		Cross-entropy	50.41
10% 10% 80%	50	MSE	30.32
		Cross-entropy	40.12
	200	MSE	35.26
		Cross-entropy	42.93
	500	MSE	33.92
		Cross-entropy	39.85

The table 1 clearly demonstrates that Cross-entropy configuration was better than MSE. This is likely due to the problem being a classification type of problem and cross-entropy is better suited for these conditions. It is also seen that the best data set split is 80% for training, 10% for validation, and 10% for testing. This is because the neural network will have a larger variety of data to test itself on with this type of split. Lastly, it is seen 500 neurons was the best for the MSE case, and 50 neurons was the best for the Cross-entropy case.

For the case of the cross-entropy configuration. The 250 and 500 neurons may have been too many neurons and have started to overfit the network. It is also possible that there is a deep local minimum for the 500 and 250 neurons, and not enough training was done for them to learn the complexities of the data set. If this is the case, more training on the data set is required, which would be fixed by increasing the *minimum validation checks*.

In the case of the MSE configuration, the 50 and 250 neurons may not have been enough to be able to properly analyze the data with all of the detailed patterns, and as a result perform worse. This is likely due to the cost function being used for a type of problem that it is not well suited for.

4 Your own conclusions with respect the results obtained.

The lab showed that the configuration of the neural network and the way that the data is split is vitally important to the performance of the neural network. As expected, the best way to split the data was to have 80% in training, 10% in validation, and 10% in test. Overall the best configuration for the network was the cross-entropy configuration with 50 neurons. The cross-entropy configuration performed better than the MSE configuration in all cases when compared with similar conditions.

In Table 2, we can see the average across all tests, which backs up the claim that overall the cross-entropy configuration was better than the MSE configuration. As mentioned previously, this is likely due to the problem lending itself to being a classification type of problem which the cross-entropy configuration is better at handling.

It is also important to note the fact that, on average, the 200 neurons performed best in both configurations. This may be due to it being a good compromise size for the network. This is due to it being able to deal with a smaller training data sets better than a 500 neuron network. While also being able to fine tune itself better than the 50 neuron network when training on the larger training data set.

Table 2: Mean of Configurations Accuracy (%)

	50	200	500	Average
MSE Configurations	29.14	35.52	35.30	33.32
Cross-entropy Configurations	48.10	48.96	47.99	48.35

Overall, the lab has shown that in order to have a neural network work properly, it is important to get the configurations correct. Choosing the correct cost function for the correct problem will greatly help the performance of the neural network. The networks created for this lab clearly are very functional, it is still possible that there are better configurations possible. This would need to be determined with further testing, and it may be more beneficial to perform some prepossessing to the data or to use other neural network architectures.

In conclusion, searching in the space of possible neural network configurations is a complex yet rewarding exercise. A deep understanding of the problem that we are trying to solve can guide the search and help finding the optimal set of hyper-parameters.