Artificial Neural Networks and Deep Architectures, DD2437

Lab 1 : Learning and generalisation in feed-forward networks — from perceptron learning to backprop

Jesús Ventas Muñoz de Lucas, Willem van der Schoot

September 1, 2018

1 Main objectives and scope of the assignment

Our major goals in the assignment were:

- Implement Neural Networks from scratch.
- Differences between Delta rule & Perceptron learning and Sequence & Batch learning.
- Analyse performance of 2-layer and 3-layer Feedforward Networks.
- Study the effects of number of hidden nodes, regularisation and noise in Neural Networks.

2 Methods

All code was written in Matlab, utilising the Deep Learning Toolbox for the latter parts of the lab. For Part I, we have assumed that our algorithms have converged when the error increment between two consecutive iterations is small enough (1e-7 typically).

3 Results and discussion - Part I

3.1 Classification with a single-layer perceptron *(ca.1 page)*

3.1.1 Classification of linearly separable data

Perceptron learning guarantees convergence if the data is linearly separable, the algorithm terminates when it can classify all training samples correctly. The Delta rule algorithm terminates when it finds the weights which satisfy a certain minimum error, not guaranteeing to classify all training data correctly though. Figure 1 displays these findings with the step size in both cases, 0.01.

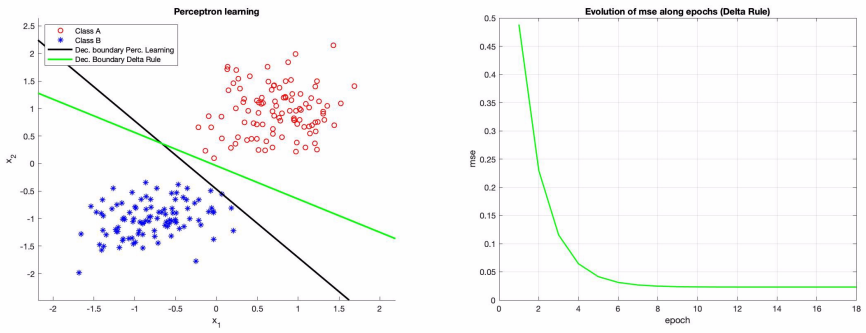

Figure 1: Perceptron Learning vs Delta Rule (left) and Learning curve for Delta Rule (right)

It was found using the delta rule, sequential learning converges much faster than batch learning on average (Learning rate : 0.01, Simulations : 100), Figure 2 displays the respective histograms of these results. Sequential mode needs fewer epochs on average to converge since it uses a stochastic gradient descent method to find the minimum error, whereas in batch mode is a gradient descent method. The variation of required epochs is influenced by the random initialization of weights as seen in Figure 2, especially in batch mode. The Mean Squared Error (MSE) is a little higher in batch mode due to the selected stopping criteria.

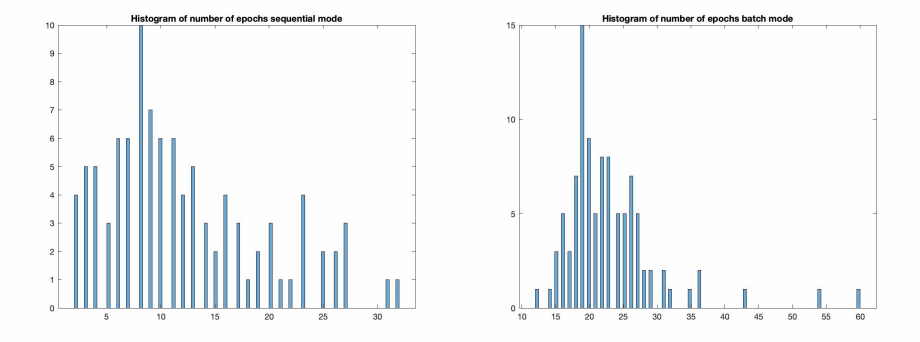

Figure 2: Histogram of epochs until convergence in sequential and batch modes

If bias is removed, the decision boundary always crosses at the point (0,0). Therefore, not all patterns can be separated (for example, two multivariate Gaussian's with means (1,1) and (3,3) could not be). However, even without bias, data can be classified if we do preprocessing such as a non-linear transformation.

3.1.2 Classification of linearly non-separable data

When data is not linearly separable, perceptron learning does not converge since the error is never zero. The Delta rule also cannot separate all data as well, however it does converge as it focuses on reducing the MSE to a minimum.
Training the networks on bias datasets (higher number of samples from one class than the other) creates a bias in the decision boundary towards the class with more samples.

3.2 Classification and regression with a two-layer perceptron *(ca.2 pages)*

When the number of hidden nodes is increased the validation performance improves. However, having more than 9-10 nodes does not improve the validation performance by much, having too many nodes does increase the complexity of the network though, leading to over-fitting. Therefore, the best option found was 8 nodes achieving almost 0 mis-classifications.

When separating the data set into training and validation sets, plots of Figure 3 have been obtained. The multi-layer perceptron cost function does not have to be convex as the training error can have different local minima. As expected, validation error decreases to a minimum before it begins to increase. This increasing is due to overfitting of the data, which deteriorates generalization properties of the network, so early stopping in the epoch of the minimum would be a good point. Figure 3 shows the training and validation MSE for a network of 6, 8 and 10 hidden nodes (with the same data set). It can be seen that the higher the complexity, the sooner the overfitting begins.

Input	1	2	3	4	5	6	7	8
h1	0.50	0.62	-0.99	-0.10	0.94	-0.99	0.46	-0.70
h2	-0.99	0.33	-0.99	-0.59	-0.40	0.74	0.97	0.99
h3	0.98	0.99	0.40	-0.99	-0.21	-0.77	-0.83	0.98
Binary	101	111	001	000	100	010	110	011

Table 1: Outputs of hidden nodes for each input and associated binary codification

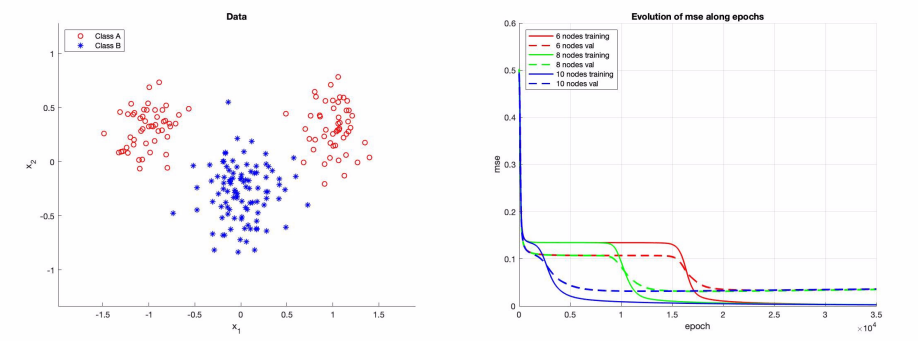

Figure 3: Data (left) and Comparison learning curves for 6,8 and 10 nodes (right)

In Figure 4 the batch and sequential modes have been compared using 6 hidden nodes. The training error in sequential mode oscillates as the error is estimated using only one sample (stochastic gradient descent). The validation error in sequential mode has the expected shape, reaching a minimum. However, it was found that the minimum is higher in the case of sequential mode than batch mode.

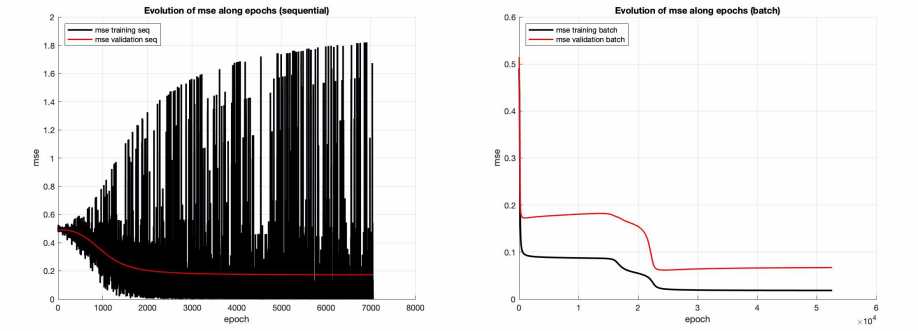

Figure 4: Training and validation error in sequential (left) and batch (right)

3.2.1 The encoder problem

Outputs of the hidden nodes for each input are shown in Table 1. If negative and positive values are thought of as binary numbers, the output of the hidden nodes corresponds to a binary codification with 3 bits, so 8 possible combinations are available (000, 001, 010, â , 111). Therefore, the purpose of the hidden nodes could be to map the input into a lower dimension space, which is useful, for example, to represent digital signals with the minimum possible number of bits. The network does not always converge, for example, if a very high step size is chosen, the input is not mapped to the output. When the number of hidden nodes is 2, the outputs of hidden nodes did not have any encoding information since it is not possible to represent 8 values with 2 bits.

3.2.2 Function approximation

After testing with different number of hidden nodes (from 2 to 25), the function approximation with 2 nodes is very coarse compared to the 25 (see Figure **??**, which means that the MSE is high, as seen in Figure 5. With a large number of hidden nodes the performance does not increase, however the over-fitting is high. The best way is to choose around 8-12 hidden nodes, since it seems the approximation is good enough and gets a good trade off between bias and variance, and therefore it achieves good generalization.

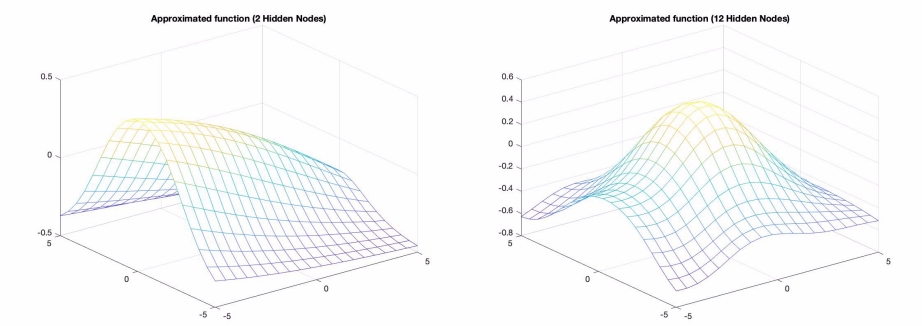

Figure 5: Function approximation with 2 Hidden Nodes vs 12 Hidden Nodes

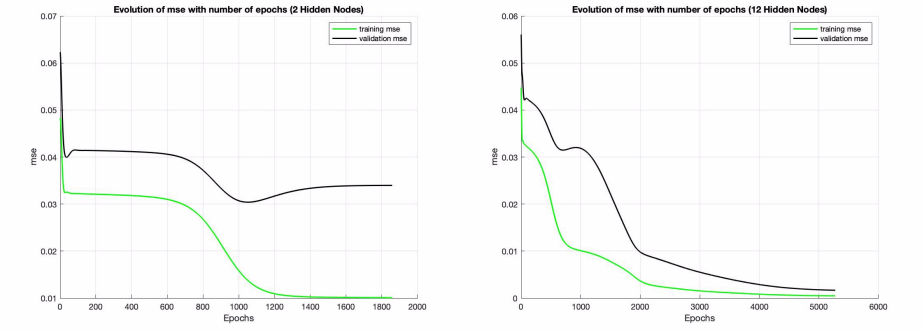

Figure 6: Learning curves (MSE) 2 Hidden Nodes and 12 Hidden Nodes

After training the network with a decreasing amount of training samples (from a 80% until around 30%), as seen in Figre 6, the validation MSE increased for a fixed number of hidden nodes. This shows that the number of training samples is very important in ensuring the network is trained well enough to generalise.

Finally, it was found that by increasing the learning rate slightly for our best model (the one with 12 hidden nodes) the convergence speed increased. However, if the learning rate was too high, the algorithm could not converge.

4 Results and discussion - Part II *(ca.2 pages)*

For this section the Matlab Deep Learning Toolbox was utilised. The network of choice was fitnet (function approximating feed-forward network). The training function used was 'traingdx' - Gradient descent with momentum and adaptive learning rate back-propagation, with early stopping. For regularization the built in NN parameter was (L2). For each network configuration, the evaluation was repeated 100 times.

4.1 Two-layer perceptron for time series prediction - model selection, regularisation and validation

The aim of regularization is to limit the over-fitting of a NN, the more complex the model the more over-fitting is likely for a given problem. In the case for the Mackey-Glass time series the number of nodes in the hidden layer were varied up to 8 along with the regularization strength, seen in Figure 7. For this problem, the network did not seem to overfit therefore the greater the regularization the greater the hold-out validation error. The optimal network configuration was 8 hidden nodes with zero regularization.

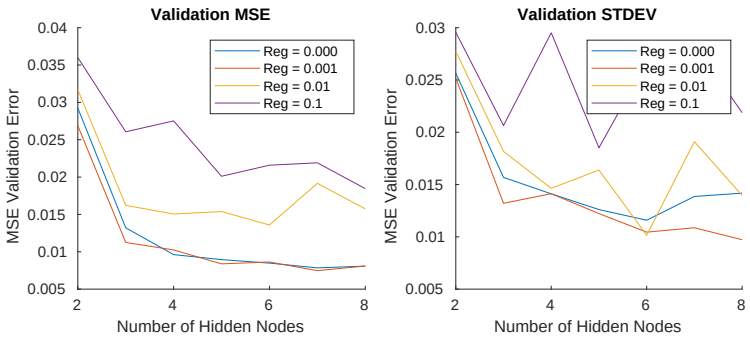

Figure 7: 2-layer Hidden Nodes vs Regularisation Strength

Regularisation was found to result in a smaller distribution of weights as seen in (Figure 8) and expected from the theory.

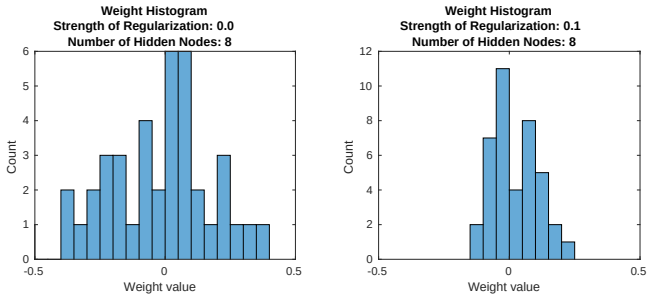

Figure 8: Weight Distribution Histogram

4.2 Comparison of two- and three-layer perceptron for noisy time series prediction

All input data had zero-mean Gaussian noise added with varying levels of sigma. The first hidden layer had 8 nodes. The addition of noise to a dataset is used to also prevent over-fitting of the data making the model more robust. Increasing the number of nodes in the second hidden layer was found to reduce the validation MSE up to a certain point, after which the model began to over-fit as it was too complex (Figure 9). When varying the number of nodes in the second hidden layer along with the noise, there was a large standard deviation in the results, making it difficult to come to a conclusion. One would expect that more noise in a more complex model would reduce the validation error.

	r=0.001	r=0.01	r=0.1
$\sigma = 0.03$	4.76e-4	1.12e-4	4.81e-4
$\sigma = 0.09$	8.65e-5	6.79e-4	3.74e-4
$\sigma = 0.18$	1.20e-4	2.02e-4	4.99e-4

Table 2: Regularisation Strength vs Noise, NN conf layers=[8,8]

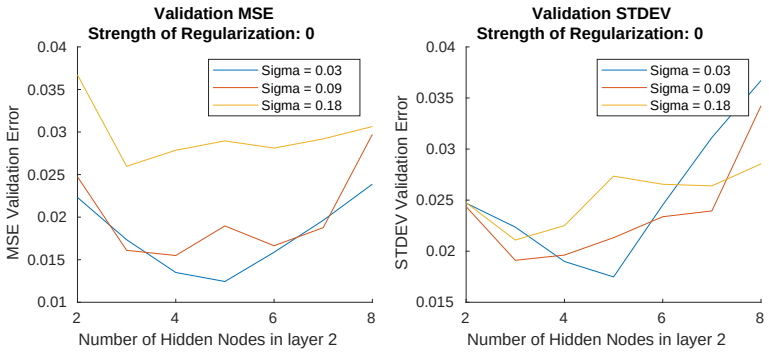

Figure 9: 3-layer Hidden Nodes vs Noise level

Both noise and regularisation contribute to reducing over-fitting, therefore adding both in increasing amounts enhance the regularisation for each. Table 2 displays the results for this, it hards to draw a definite conclusion here, but for the most complex network (layer nodes [8,8]) and with the largest noise $\sigma = 0.18$ it can be seen the combined effect is enough regularisation for r=0.001 and by increasing the regularisation strength the validation error just increases. A trade-off between the two parameters must be found for optimal generalisation.

The best 3-layer network was configuration [8,4] with $\sigma = 0.09$, retraining the 2-layer best network from the previous section [8] with noise added ($\sigma = 0.09$), the three layer network still performed better on the test dataset. The resulting accuracy of the two networks along can be seen in 10. This Figure also shows the CPU time for training networks with more hidden nodes, the greater number of nodes the more computations are required however the number of epochs can vary depending on when a sufficient minima is found.

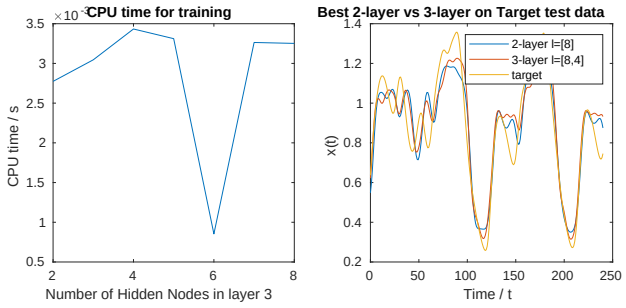

Figure 10: 3-layer Hidden Nodes vs Noise level

5 Final remarks *(max 0.5 page)*

Overall the lab satisfied the main goals listed at the beginning of the lab, with the majority of the theory easily seen in the results.