Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

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

Part 1

3.1.1 and 3.1.2

We generated data which is visualized in the plots in appendix 3.1.2 and 3.1.3. Note that the error for the perceptron algorithm goes to zero while the error for the delta rule goes to some non zero values. This makes sense since the sigmoid function cannot output exactly 0 or 1. Deltarule manages to converge pretty quickly, around 30 epochs (seen in appendix 3.1.2)
this is also illustrated in the appendix where it starts from the green line and ends at the black line. The perceptron algorithm converge in around the same number of epochs.

3.1.3

We note that the perceptron rule does not manage to fit the data well since the error is non-convergent. The delta rule manages to minimize the mean squared error since it does not require the classes to be linearly separable.

3.2.1

The error of the testing curve is a little bit higher in the beginning but reaches the same value at the end of the epochs Both the training and the test classification manage to classify the data but it seems like adding more neurons to the hidden layer the algorithm converges faster. The average mean squared error for 2 neurons is about 2.5 at epoch 50 but for 9 neurons, the error closed was the same at 40 epoch. It seemed that the 100 epochs were sufficient to converge.
The difference between the batch and the sequential learning was that the batch approach was slightly quicker at classifying and had less average error after 150 epochs. It also seemed that in the first epochs of the sequential run the error between the training and the testing data rose much faster. The training for the sequential took a bit longer, since back propagation is done more frequently (Appendix 3.2.1)

3.2.2

Looking at the findings in Appendix 3.2.2, where the first matrix is the output from the hidden layers, the second one is the weight matrix and the third is the final output; It managed to capture the expected outcome after around 7000 epochs. By examining the weights, it seemed like one of the nodes always fired a constant value (always close to -1) so the auto encoder might work with two neurons, we tested this and the error was almost the same. So in the case with three neurons in the hidden layer, one might just have the role as adjusting the bias. The code worked for various numbers of learning rates and it always managed to find a good representation.
It seemed like there was a strong connection between two of them.

3.3.1 and 3.3.2

Created the data, had problems plotting the 3D but managed to do so with the mplot3d library.

Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

The network used 10 neurons in the hidden layer and it seemed to capture the plotted graph (see graph 3.3.2 in appendix) except the lowest points. By adding 10 more neurons the image plotted perfectly. Learning rate of 0.001 was used.

3.3.3

Three different parameters were used, 4 16 and 28, for neurons in the hidden layer. For the model trained on the complete data we saw that the error went down as we increased the number of neurons. With subsampling however, 28 neurons did not really perform better than 16 neurons, four neurons performed poorly.

Part 2

4.3.1

Using one hidden layer we're able to get a very low valuation error, same with two layers but it happens faster so from a computational perspective, two layers is better. We implement early stopping as a limit for the error (0.3) because the validation error seems to be decreasing for a computationally intractable number of epochs.

Higher momentum seems better because it seemed to allow for reaching the stopping criteria earlier so we set it to 0.9. Increasing the number of neurons in the hidden layer also has this effect, thus our selected model has eight neurons in the hidden layer.

Momentum\Nodes	**2**	**4**	**8**
0.5	>20000	>20000	>20000
0.9	12842	8361	5669

Table of #iterations before stopping (error is 0.3)

4.3.2

When the standard deviation is 0.03 the validation error is decreasing in the same way as in the earlier part. Changing the amount of nodes then has no significant effect on the performance. When it is 0.09 for some runs the validation error had a local minima and for others not, one can suspect this has to do with the stochasticity of the data and initiation of weights. The errors for different runs are collected in the table below for different number of nodes in hidden layer 2. To account for stochasticity, three run averages was taken.

#Nodes layer 2	**2**	**4**	**6**	**8**
Stopped error Or error after 10k iterations	1.89	1.79	1.8	1.92

Standard deviation = 0.09

Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

#Nodes layer 2	**2**	**4**	**6**	**8**
Stopped error Or error after 10k iterations	6.1 Stopped twice	6.11 Stopped at 2k,7k iterations	5.96 Stopped at 2k,4k ,3.5k iterations	6.0 Stopped at 2k,7k 6k iterations

Standard deviation = 0.18

From these tables it is apparent that it does not make a big difference how many nodes one uses in the output layer the test errors are of the same order. The only difference is that when the standard deviation is 0.18, the test error seems to have local minimas and thus over fitting becomes an increasing problem when data is more stochastic.

How the regularization affect the performance, we check for nodes equal to 8 and get a two run-average of test errors in the table below

Mom\St.dev	**0.03**	**0.09**	**0.18**
0.9	0.62	2.04	5.80
0.4	0.9	2.27	5.89
0.1	1.26	2.67	5.85

Test Error after 10k iterations or early stopping

The momentum term seems to decrease the error when the standard deviation is 0.09 or lower.

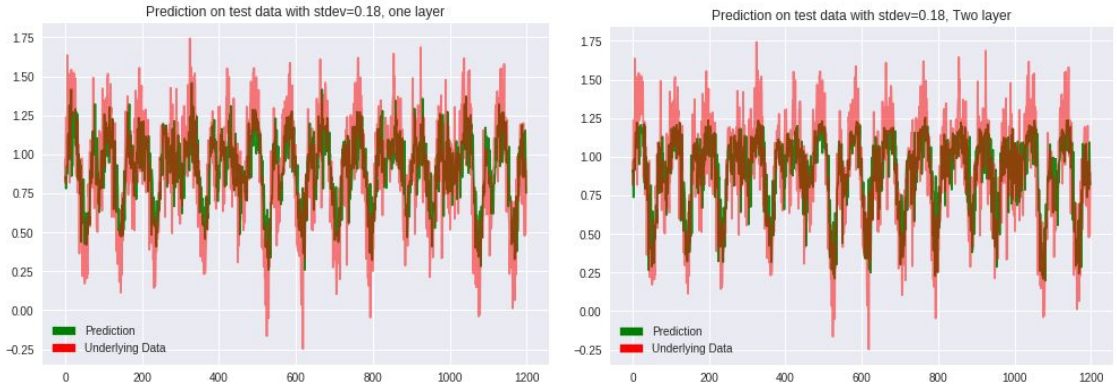
The best models are when the momentum is 0.9 and the number of nodes in the hidden layer is 8. We train on one layer model on the noisy data and obtain these generalisations

Layers\St.Dev	**0.03**	**0.09**	**0.18**
One hidden layer	1.27	3.10	6.97
Two hidden layers	0.62	2.04	5.80

Best model comparison, Test Error

We see that the two layer network outperforms the one layer network so having two layers generalizes better when there is noise.

Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)



By inspection when there is lots of noise the networks are having trouble predicting the spikes.

Hidden nodes in each layer	**2**	**4**	**6**	**8**
One hidden layer	avg 0.0036 max 0.0503 min 0.0030	avg 0.0038 max 0.0488 min 0.0031	avg 0.0037 max 0.0735 min 0.0025	avg 0.0038 max 0.0655 min 0.0029
Two hidden layer	avg 0.0044 max 0.0528 min 0.0036	avg 0.0044 max 0.0421 min 0.0033	avg 0.0044 max 0.0383 min 0.0034	avg 0.0044 max 0.0309 min 0.0031

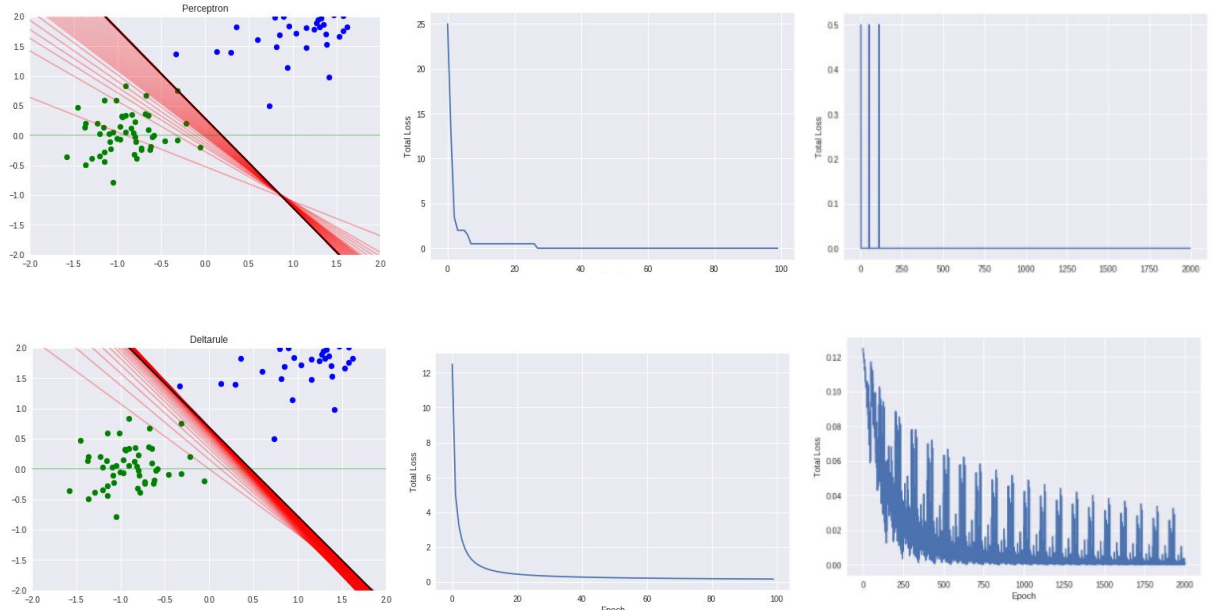
Batch training and momentum are used in time calculations

It is clear that the backpropagation is slower with three layers since it takes more time to feed the delta of weights backwards for each layer. The average time is about 18% slower with three layers and the max and min time is also greater than in the two layers. The number of nodes don't seem to have any effect, since the computation for all the weights update is done in a single linear algebra function. Computation time varies between runs on different CPU's and GPU's. These results ran on Google Collaboration.

Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

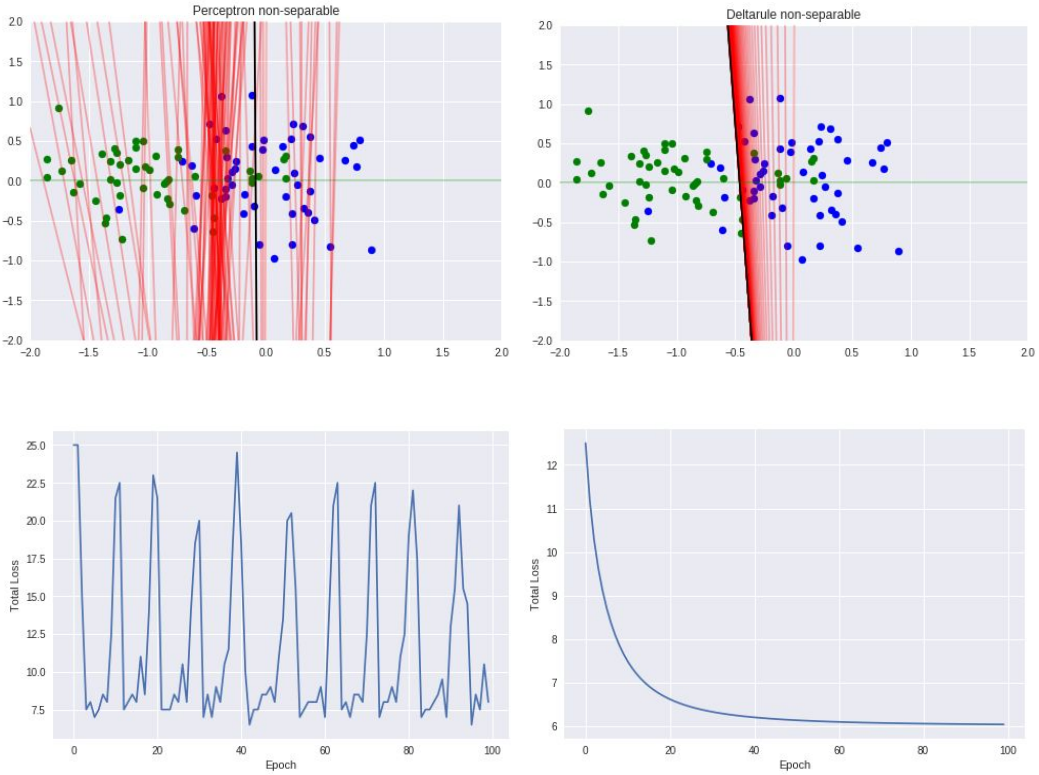
Appendix 1

3.1.2 (Linearly separable data - Activation vs Sigmoid)



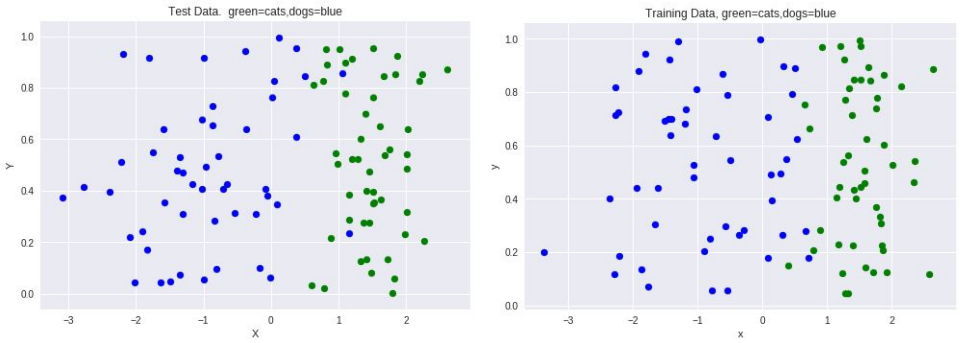
Top: Perceptron converge(left) with activation via batch(middle) and sequential learning (right).
Bottom: Perceptron converge(left) with sigmoid via batch(middle) and sequential learning (right).

3.1.3 (Non linearly separable data - Activation vs Sigmoid)

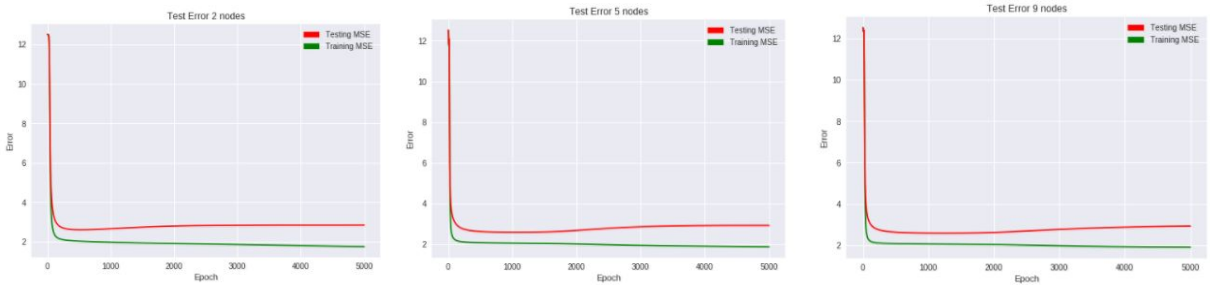


Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

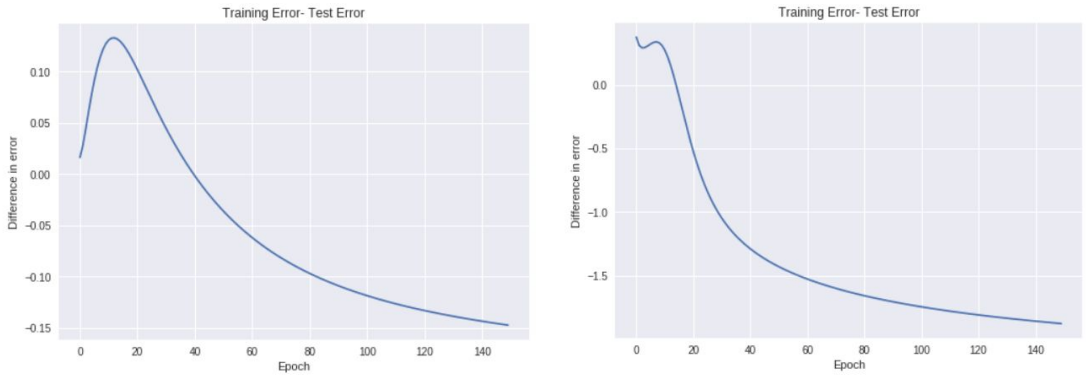
3.2.1 (Data - Batch - Sequential)



Data used for training (left) and for testing (right)



MSE of training (green) vs testing (red) error with 2,5 and 9 neurons in hidden layer

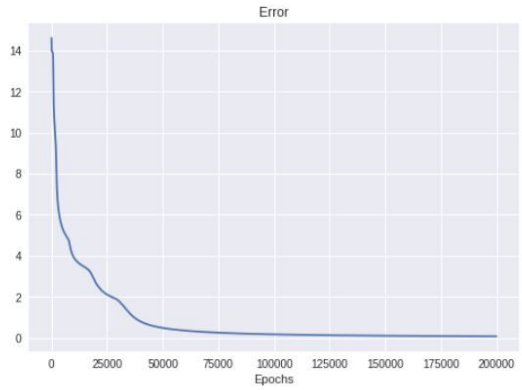


Difference between training and testing MSE in Sequential (left) and Batch (right) of the first 150 epochs with 2 neurons.

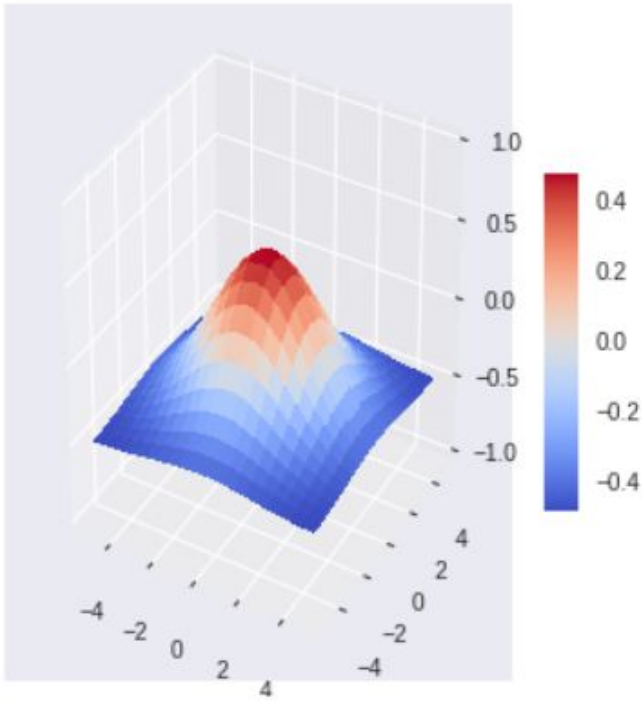
Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

3.2.2

```
[[ 0.42 -1.   -0.87]
 [-1.    0.42 -0.81]
 [-0.42 -1.   -0.86]
 [ 1.    0.42 -0.95]
 [-1.   -0.47 -0.92]
 [ 0.44  1.   -0.95]
 [ 1.   -0.49 -0.96]
 [-0.49  1.   -0.88]]
[[ 0.4  -3.51  0.71]
 [-3.42  0.45  0.9 ]
 [-0.49 -3.86  0.75]
 [ 3.61  0.45  0.15]
 [-3.87 -0.5   0.44]
 [ 0.42  3.71  0.22]
 [ 3.66 -0.53  0.06]
 [-0.58  3.58  0.66]
 [ 0.18  0.22  0.17]]
[[ 0.91 -1.   -0.93 -1.   -1.   -1.   -0.92 -1.  ]
 [-1.    0.91 -1.   -1.   -0.93 -1.   -1.   -0.93]
 [-0.93 -1.    0.91 -1.   -0.93 -1.   -1.   -1.  ]
 [-1.   -1.   -1.    0.91 -1.   -0.93 -0.93 -1.  ]
 [-1.   -0.94 -0.93 -1.    0.91 -1.   -1.   -1.  ]
 [-1.   -1.   -1.   -0.93 -1.    0.91 -1.   -0.94]
 [-0.93 -1.   -1.   -0.94 -1.   -1.    0.9  -1.  ]
 [-1.   -0.93 -1.   -1.   -1.   -0.94 -1.    0.91]]
```

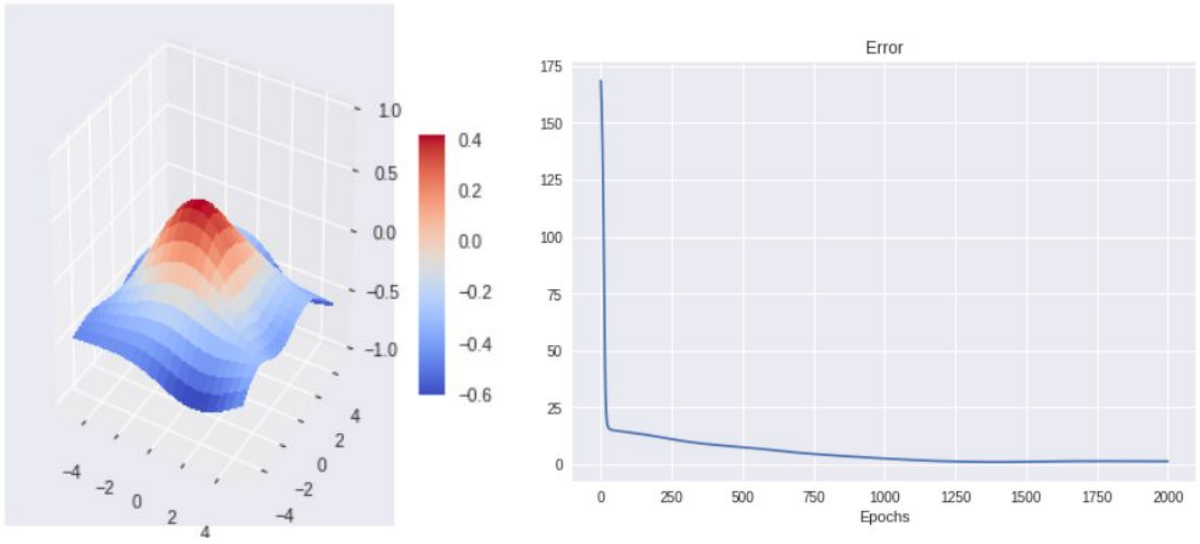


3.3.1

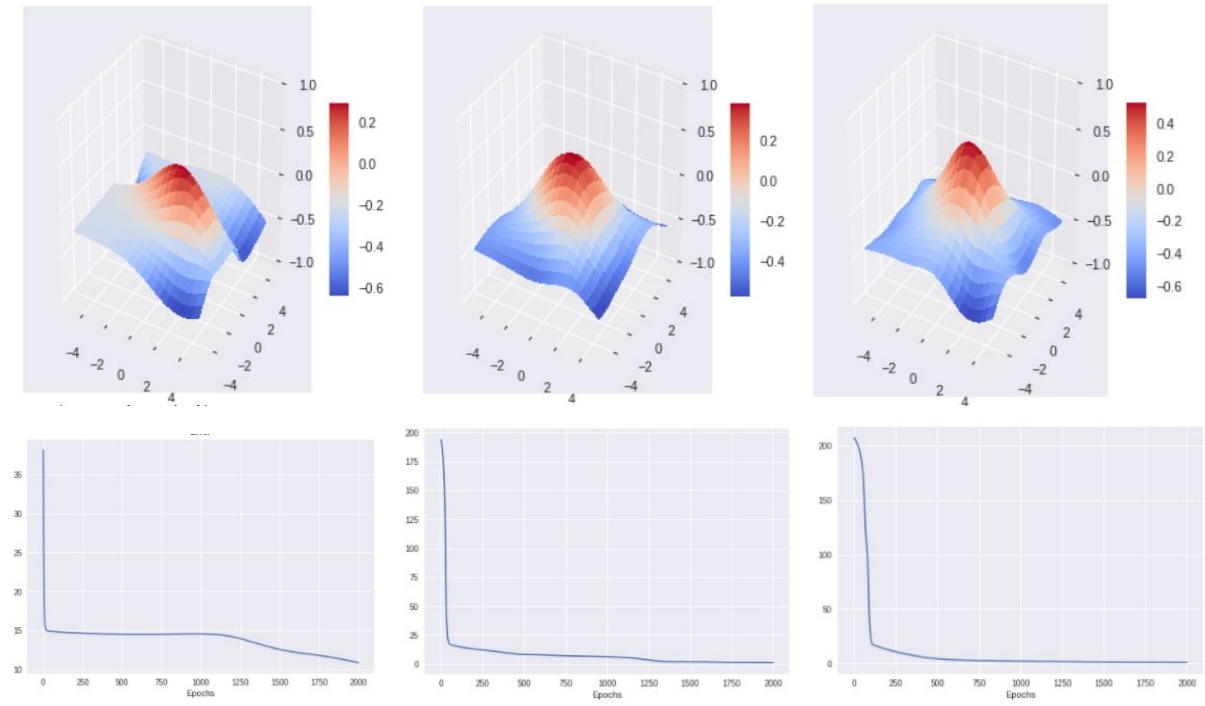


Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

3.3.2

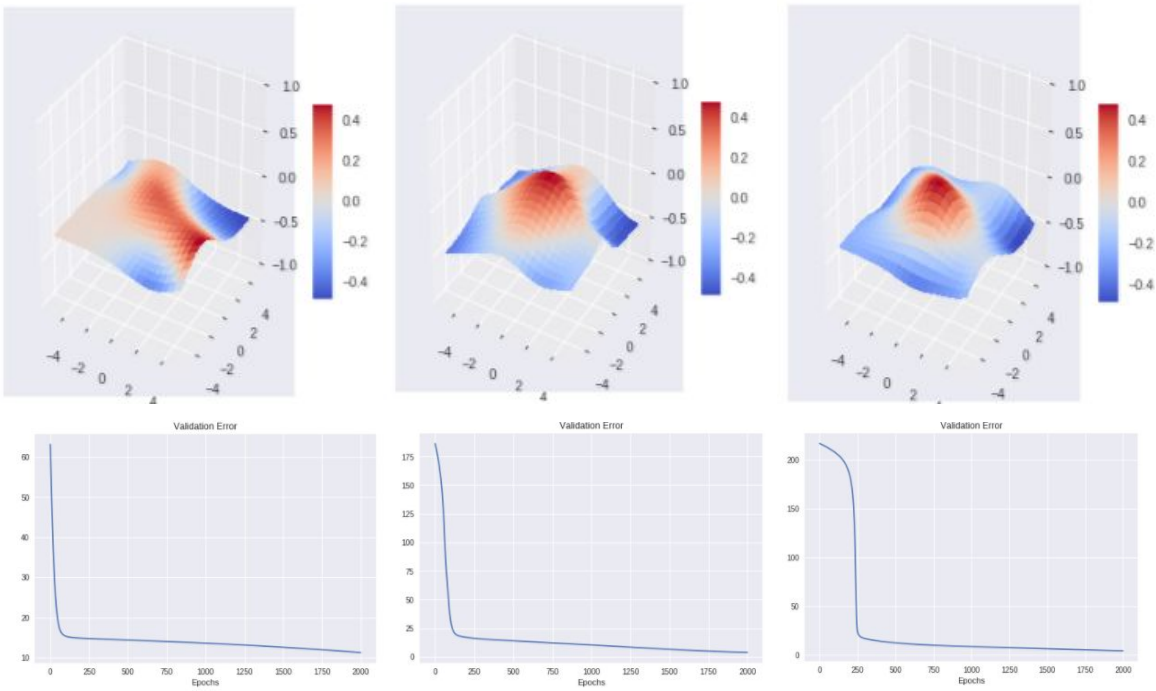


3.3



4,16 and 28 neurons with all points (top) with corresponding errors (bottom)

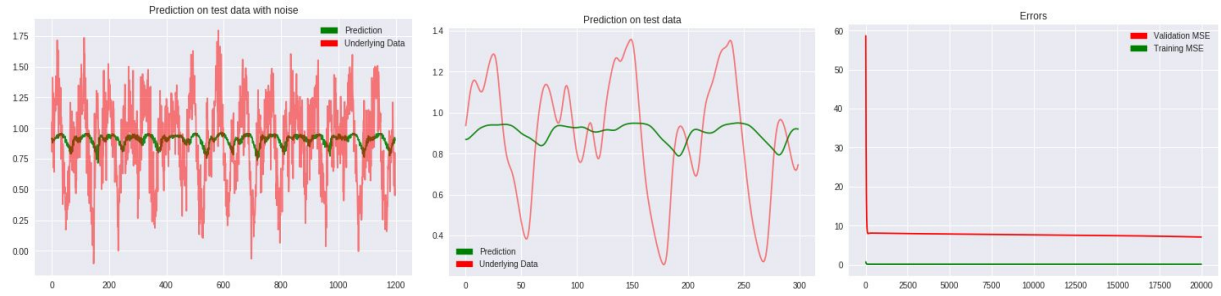
Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)



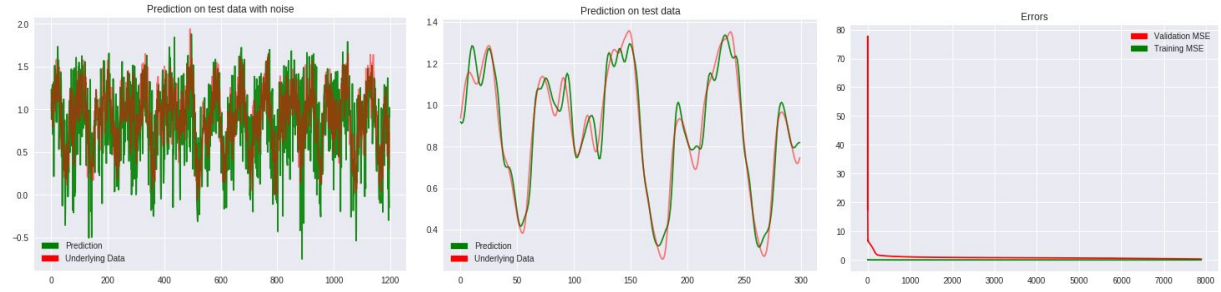
4,16 and 28 neurons with half points (top) with corresponding errors (bottom)

Part 2

4.1.1



No batch training, no momentum, one hidden layer and low set of neurons



Batch training, 0.9 momentum, two hidden layers and more neurons in layers. - Best model

Carl Ridnert (940325-0112)
Óttar Guðmundsson (910302-0450)

4.1.3