ANNDA - Lab 2

Niels Agerskov, Lukas Bjarre, Gabriel Carrizo

December 2017

1 Batch Mode Training

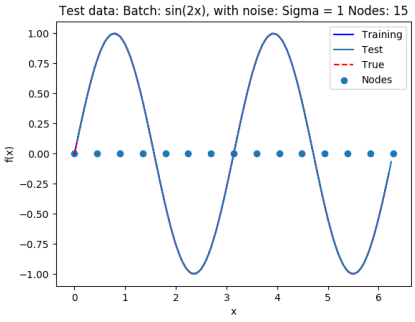
In this part of the lab we trained an RBF network to approximate two functions:

$$f_1(x) = \sin(2x) \tag{1}$$

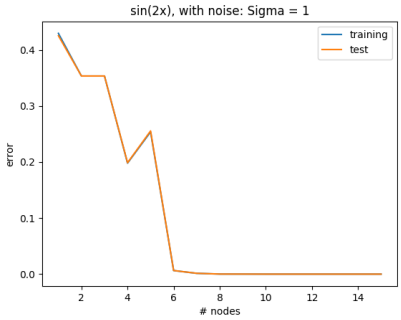
$$f_2(x) = \text{square}(2x) \tag{2}$$

Where $x\epsilon[0, 2\pi]$ which is sampled with a step sizeof 0.1. The function is consequently tested with $x_{test} = x + 0.05$.

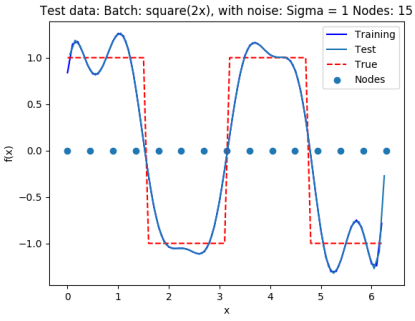
Try to vary the number of units to get the absolute residual error below 0.1, 0.01 and 0.001 in the residual value (absolute residual error is understood as the average absolute difference between the network outputs and the desirable target values). Please discuss the results, how many units are needed for the aforementioned error thresholds?



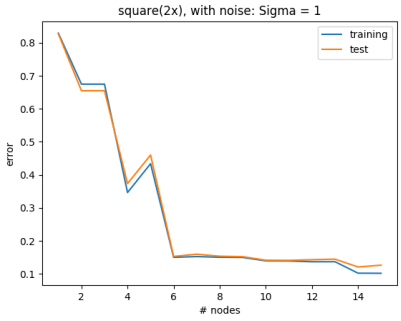
(a)



(b)



(c)



(d)

The error of the sine function went below 0.001 at about 6 nodes with a variance of 1, however for the square function it was much harder to reach the wanted error levels. This seem reasonable since the RBF functions are quite far from the shape of the square function, so the approximation should be quite bad.

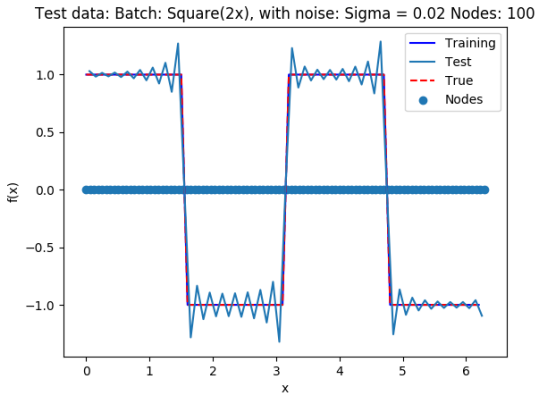
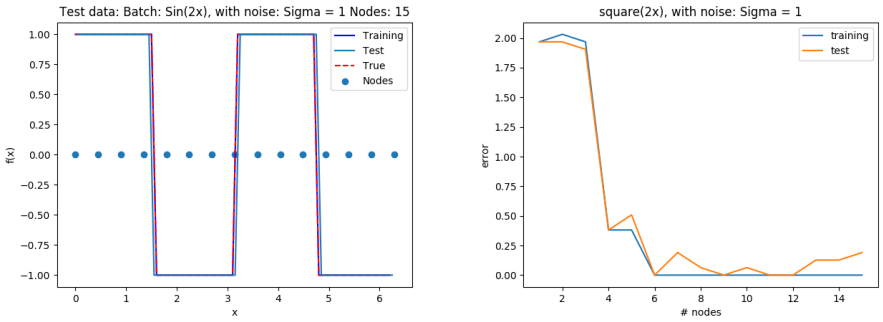


Figure 2: test error 0.06

How can you simply transform the output of your RBF network to re- duce the residual error to 0 for the square(2x) problem? Still, how many units do you need? In what type of applications could this transform be particularly useful?

The residual error can be reduced to 0 by applying a sign function to the output which will act as a activation function. While this greatly improves the result, about ?? nodes were still necessary. This type of activation is useful when the RBF network should be used as a classifier.



(a) (b)

Figure 3: Taking the sign of the estimated square function makes us reach 0 error with only 6 nodes.

2 Regression with noise

When noise was added to the function it became much harder to reach the wanted error levels. The amount of error heavily relied both on the number of nodes and the width of the RBF functions.

Effect of altering sigma:

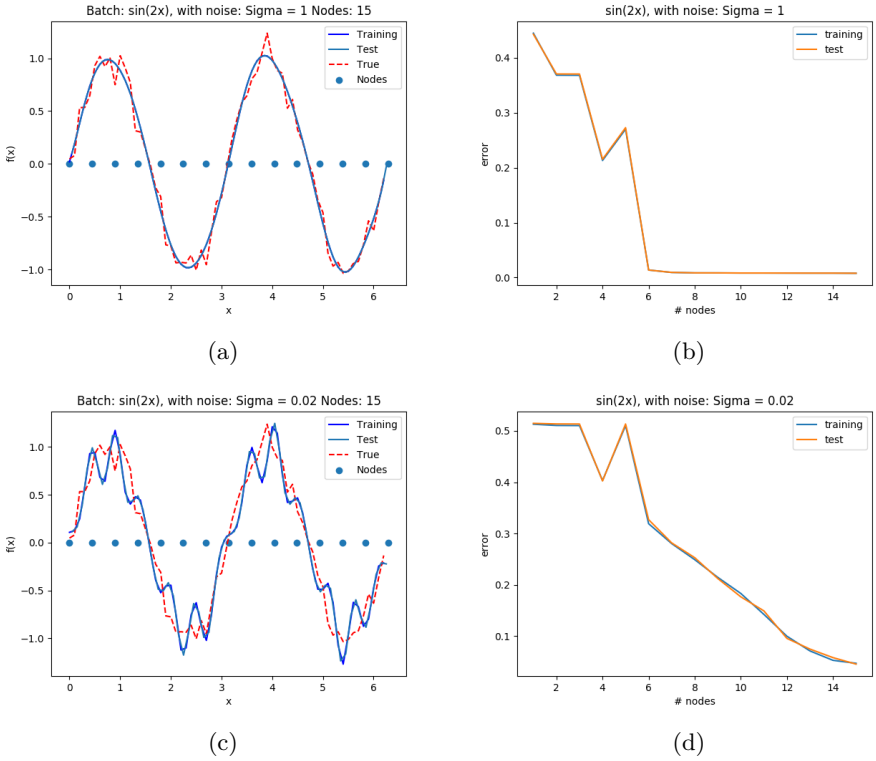


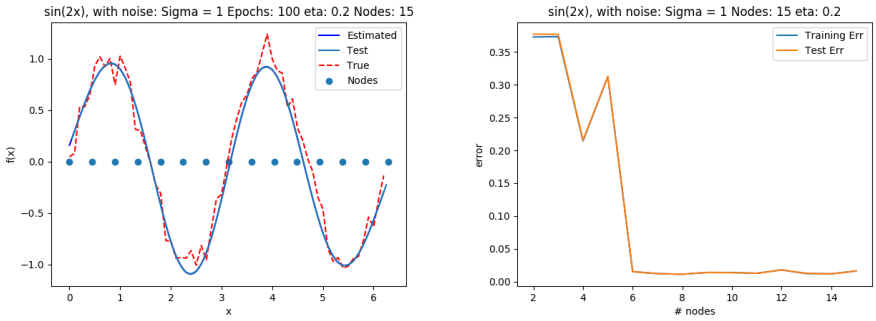
Figure 4: Plots of the effect of altering sigma on networks trained with batch method

Changing the width of the RBF functions changed the amount of smoothing applied to the input data. A larger width resulted in a smoother prediction, more similar to the underlying function. A smaller width resulted in a prediction containing more of the noise that was added to the labels. Effect of number of epochs:

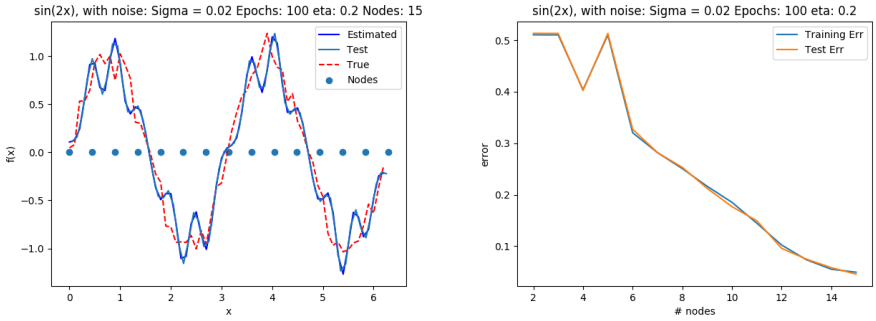
The rate of convergence follows changes in eta quite well. A increase in learning rate increases the rate of convergence up to a certain degree, if the learning rate is to high the error start to oscillate. A lower learning rate leads to a more stable error but requires longer learning time.

The positioning of nodes in the input space is quite important. The strategy used was to place the nodes at even intervals of quarter pi, aiming to place the nodes at the tops and bottoms of the sinus wave. This method was better than totally random placement since the shape of the RBF functions approximate the shape of the tops and bottoms of the sinus wave quite well.

In comparison to a MLP network the RBF approach is way more efficient and accurate at predicting the sinus function. This was especially noticeable when it came to prediction of validation data where it was impossible to predict any of the function noise. The training time was also magnitudes higher with at least 15000 epochs was necessary to approach convergence.

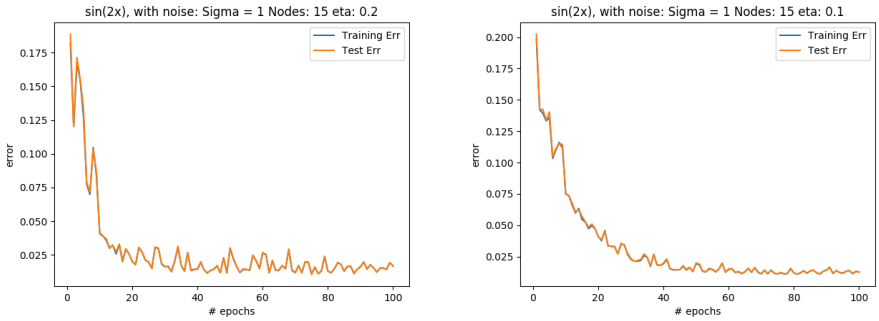


(a) (b)



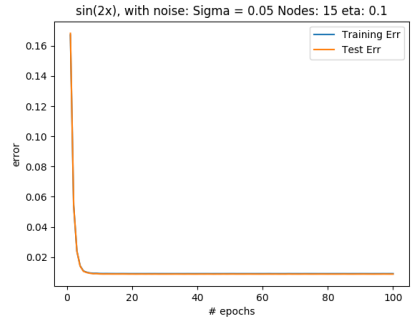
(c) (d)

Figure 5: Plots of the effect of altering sigma on networks trained with sequential method

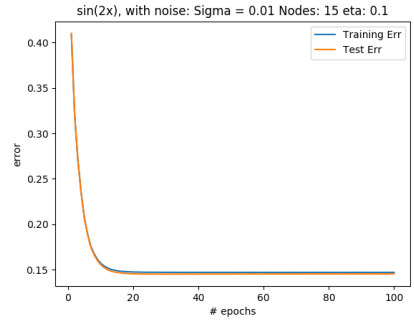


(a)

(b)



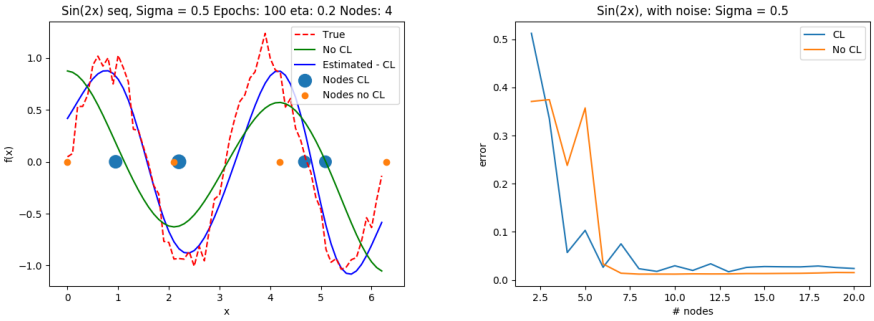
(c)



(d)

Figure 6: Plots of the error as a function of eta on networks trained with sequential method

3 Competitive Learning

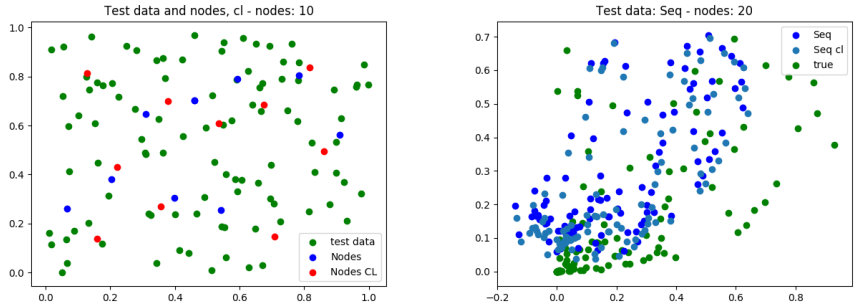


(a) Generated functions plotted with the node RBF node positions, with and without CL.

(b) Error as a function of the number of nodes, with and without competitive learning.

Figure 7: Plots of the effects of Cmpetitive Learning

The main improvemnt from using competitive learning is the increased learning rate or quicker convergence of the error. Along with this quicker learning speed a improved prediction result could also be observed which is not surprising when looking at the placement of the RBF units.



(a) Test data and the node positions before and after competitive learning

(b) Output of networks trained with and without competitive learning

Figure 8

The effects of applying competitive learning to the 2D data were quite weak. The resulting predictions could be improved a bit which can be seen in the result plots. Perhaps more interesting is what hap pend to the location of the nodes in the input space. The nodes were noticeably more spread out, allowing for better representation of the input data. There were still some areas near the middle of the data where multiple nodes ended up, however at the edges the nodes placed with the use of CL reached much more of the input data. The same amount of "representation" could probably have been reached with random placement, however with CL this could be done using a

lot fewer nodes.

4 Topological Ordering of Animals

As can be seen in the plot, similar animals are grouped together. This is expected due to how the weight nodes are aggregated together using neighbourhoods. The neighbourhood is defined with a cutoff at the edges, so index 0 and index N, where N is the maximum index are considered far away from each other. One interesting observation is that bat and elephant end up very close together. While this might seem wrong, if the distance between their respective attribute vectors are computed they end up at a distance of about 3 from each other. This is quite close considering that the minimum distance is about 1.6 and the maximum distance is about 5.6.

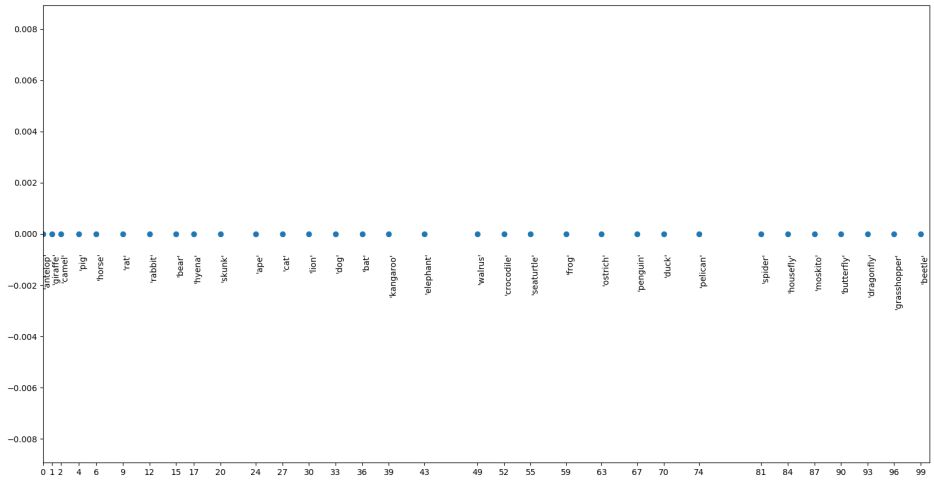
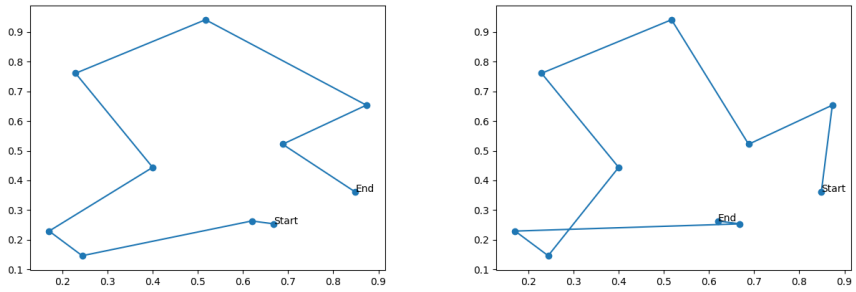


Figure 9: Topological ordering by self organising map

5 Cyclic Tour

In this part a similar network as in the last part was created, with the main difference being that the neighbourhood is now cyclic so that the first and last index are considered neighbours. Most of the time the network is able to find a short route however it sometimes get stuck in a local minimum and has to cross over another path. This is most likely due to the stochastic nature of how the weights are initiated.



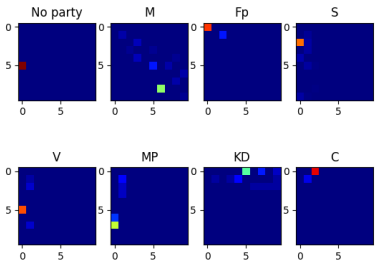
(a) Example of a good path created by the SOM.

(b) Example of a bad path created by the SOM.

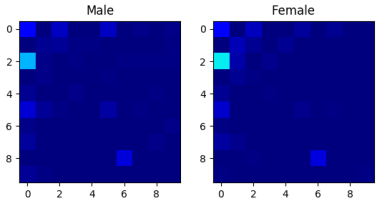
Figure 10

6 Data Clustering: Votes of MPs

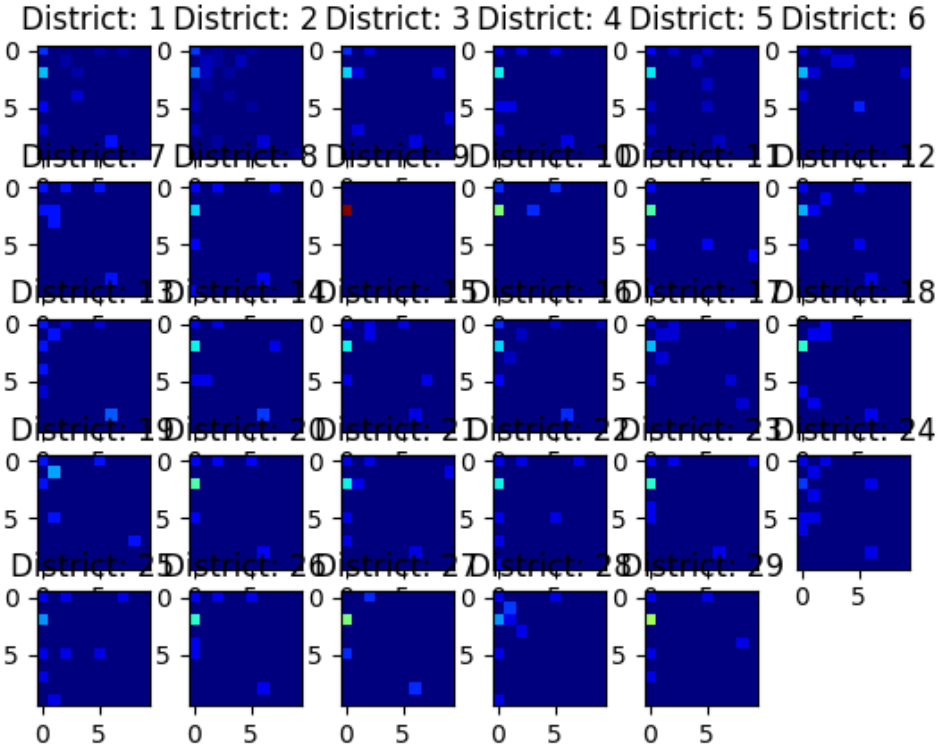
In this part the 31-dimensional input data consisting of how different MPs of parliament cast their votes in 31 different questions was mapped to a 10x10 grid, visualising voting patterns. These patterns were then evaluated with respect to gender, district and party alignment. In both the gender and party alignment obvious trends could be observed. Most of the time MPs belonging to parties with similar political alignments would place their votes in the same quadrant, while MPs of opposite parties would be spread apart. When evaluated with respect to gender it was easy to see that MPs voted very similarly independent of being male or female. Finally for the district evaluation it was hard to draw any conclusions other than that vote patterns were heavily dependant on the size of the districts. Bigger districts had a larger spread in their votes than smaller districts.



(a) MP votes evaluated by party allegiance.



(b) MP votes evaluated by gender.



(c) MP votes evaluated by district.