Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and

self-organisation

Gabriel Ulander Voltaire, Leonard Halling and Philip Claesson

#### October 10, 2018

1. **Main objectives and scope of the assignment**

##### Our major goals in the assignment were

*•* to get hands on experience of how to implement a RBF single layer neural network from scratch

*•* to understand and analyze how different hyperparameters alters the results and behavior of a RBF NN

*•* to understand the concept of SOM and where/how they may be applicable

# Methods

We have used python 3.0 in Jupyter Notebooks in order to perform the tasks of this lab. The Numpy library has been used to expose functions for maths and linear algebra. In the final part of SOM the library Pandas was used in order to easily handle the visualization of different parameters through containing and matching different datatypes in single dataframes.

# RBF networks and Competitive Learning

We have used python 3.0 in Jupyter Notebooks in order to perform the tasks of this lab. The Numpy library has been used to expose functions for maths and linear algebra. In the final part of SOM the library Pandas was used in order to easily handle the visualization of different parameters through containing and matching different datatypes in single dataframes.

### 3.1 Function approximation with RBF networks

### 3.1.1 Analysis of n > N

First of all, it was evident that the number of Nodes, n, could not exceed the number of samples N in batch mode. If n > N the equation system shown in part 2.1.1 in the lab became unsolvable because it had more unknown variables than equations (overdetermined). It could have been solvable if some of the equations were linear combinations of each other. However in batch training, it could look like in fig. 1, the error grows very large due to the insolvable system given n > N.

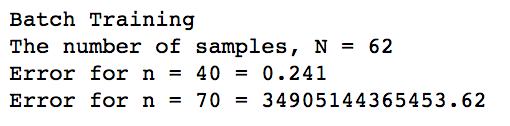


Figure 1. The residual error for batch training. The system lacks solution in the case of n > N, hence the error becomes exponentially large.

### /Users/philipclaesson/Dropbox/Screenshots/Skärmklipp 2018-10-10 17.19.16.png3.1.2 Optimal n, width, learning rate and delta.

Figur 2 The residual error for delta learning with different unit width and number of units. Units distributed evenly in input space. No clear correlation of width and performance, could be due to noise.

### In order to analyze the importance of the number of units (n) and their width we ran the batch and delta learning iteratively and stored the residual error on the test set in a matrix. The units were placed evenly in the interval. The conclusion was that the variance does *not* have a significant impact on the error, the small differences may very well just be due to some setups fitting better to the noise. However, we are quite skeptical to this, we believe the variance should in fact alter the performance of the RBF. In similar manners we could conclude that a learning rate of about 0.7 with about 500-700 epochs was performing best for delta-learning (they are not used in this batch learning). These results were quite clear, in order to keep it brief we are not adding these tables but they can be found in the code.

### 3.1.3 Positioning of units

|  |  |  |
| --- | --- | --- |
|  | Batch | Delta |
| Smart | 0.235 | 0.27 |
| Even | 0.23 | 0.26 |
| Random | 0.529 | 0.59 |

### Our hypothesis was that placing nodes in the extreme points of the sinus curve would yield the highest performance, in particular for sin2x. We hence placed 17 nodes with an interval of pi/8 in the input space (including 0 and 2pi), meaning one node in each max, min point and at x = 0. We compared this to placing at random and to even distribution. The results did according to Table 1, not show any significant difference between the “smart” and the “even” distribution. There was no significant improvement for step(2x).

### Table 1. Residual error for batch and delta training with different node placements of 9 nodes according to random, even or “smart” distribution. Data from sin(2x) function with added noise.

|  |  |  |
| --- | --- | --- |
|  | Batch | Delta |
| Noisy sin(2x) | 0.198 | 0.22 |
| Noisy sin(2x) | 0.035 | 0.06 |
| Clean step(2x) | 0.126 | 0.16 |
| Clean step(2x) | 0.126 | 0.16 |

### 3.1.4 Clean vs. Noisy data

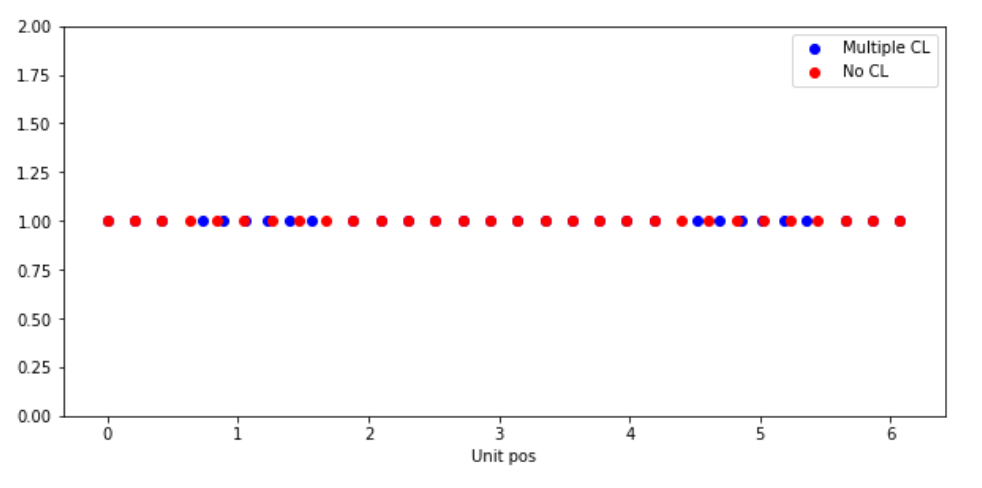
### We trained the RBF for batch and delta learning with n = 30, width = 0.1 and (for delta training) lr = 0.7, 500 epochs. We used both clean and noisy data of step(2x) and sin(2x). The results were higher accuracy for clean data and for the sin function, which was expected.

### Table 2. Residual error for batch and delta training with noisy and clean data for the two functions.

### 3.1.5 RBF vs MLP

### Competitive learning for RBF unit initialisation

In order to not initialize the RBF units arbitrary, we implemented competitive learning. We implemented both learning that affected just the nearest node (“vanilla method”) but also competitive learning which affects k nearest neighbors. Although examining several settings in terms of number of units, data from both functions both with and without noise we were not able to see a significant improvement in terms of performance.



Figur 3 The unit positions, with and without competetive learning. The units pi/2 and 3\*pi/2 were significantly affected by the competetive learning.

### Two-dimensional regression with RBF network*(ca. 1 page)*

Please refer first to the results in the previous section, i.e. those obtained without any automated initialisation. Then in the next subsection focus on two-dimensional regression with RBF networks.

* Gör klart 2-d

1. **Results and discussion - Part II: Self-organising maps**

### Topological ordering of animal species

The task, to order a set of animals, each represented as a set of binary features, served as a good introduction to SOM learning. With an initial neighbour size of 50, decreasing by 2 per epoch for 20 epochs until nsize = 1. See output in fig. 1. As the results can only be assessed qualitatively, it is hard to assess the exactness of the algorithm. However Fig. 1 shows that some of the similarities between the animals are caught, for example we find insects gathered in the bottom, followed by birds and then reptiles. In the top of the table are mostly four legged mammals.

### Cyclic tour

### *../../../../../Dropbox/Screenshots/Skärmklipp%202018-10-10%2012.34.45.png*In the Cyclic tour task, we were asked to implement the SOM algorithm in order to find some solution to the travelling salesman problem.

### The solution is largely the same as the one in 4.1, but with some differences. The neighbourhood was made circular to capture the fact that any point can be connected to any other, i.e. point 0 and 9 are neighbors. This was done by using the mod(10) operator in get\_neighbours() function.

### As can be seen in Fig. 3, a short path between the nodes is found. It is arguable that there may be shorter paths. The result is stable with increased number of iterations 20+.

* 1. **Clustering with SOM**

### In the third part, the same algorithm is used in order to model the similarity between MPs based on their voting in 31 voting rounds. The similarity was modeled in two dimensions to a 10x10 grid. The main differences from 4.2 was the need to once again change dimensions. This time we wanted 100 different weight vectors of 31 columns each, but more importantly we needed to regard similarity not only in 1 dimension but in both X and Y axis. This was done by altering the similarity function into computing the spatial distance (assuming a 10x10 grid) between two datapoints. This was done in a brute force manner, which cause somewhat bigger time complexity in the training phase. The neighborhood size (which was now interpreted as a boundary of spatial distance rather than number of neighbors) was originally set to 25 and ../../../../../Dropbox/Screenshots/Skärmklipp%202018-10-10%2014.29.12.pngdecreased for each epoch.

Figure 3 The results of the neighborhood mapping by sex. As can be seen, there are some significantly overlapping segments, but also some clear differences between how MPs of different genders voted.

### The result was then paired with and grouped by features known for each MP: Gender, Political Party and District. Plot size is weighted by number of MPs. Please note that the plot size of the *different* figures however were given different weights in order to increase visibility. In fig. 3, it can be seen that MPs have similarities based on gender that there are some small areas with clusters where either men or female MPs were overrepresented.

### ../../../../../Dropbox/Screenshots/Skärmklipp%202018-10-10%2014.35.13.pngIn fig 4. It can be seen that parties that are on different spectrums of the real political spectrum, such as S and M are also fairly separate in the 10x10 mapping. In the bottom left corner, the fairly similar parties C and FP are clustered. It can also be seen that for example S has low spread, with most MPs in the same point with just minor clusters ranging in close proximity. The MP party has a relatively big spread with 5 points of similar size with fairly big range.

Figure 4 The results of the neighborhood mapping by Party, shows voting tendencies for parties, for example S and M in different sides of the Y-axis spectrum. It also captures the variance within each parties.

### We can conclude from section 4.1 – 4.3, that the SOM algorithm according to this empirical analysis seem to do a fairly good job at catching underlying patterns and modelling similarity from binary features. The mapping per district was left out as it did not show interesting correlations due to lack of background info about the districts.

1. **Final remarks** *(max 0.5 page)*

*Please share your final reflections on the lab, its content and your own learning. Which parts of the lab assignment did you find confusing or not necessarily helping in understanding important concepts and which parts you have found interesting and relevant to your learning experience?*

*Here you can also formulate your opinion, interpretation or speculation about some of the simulation outcomes. Please add any follow-up questions that you might have regarding the lab tasks and the results you have produced.*

Summering:

* Lyckades inte implementera RBF för 2-dimensionlla inputs.
* Hann inte jämföra RBF med MLP från labb 1. Det borde egentligen bara vara att köra MLP med datan som genereras i labb 2 och jämföra accuracy, men min hjärna är lite för kokt just nu för att det ska gå. Sorry.