Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2
Radial basis functions, competitive learning and self-organisation

Benjelloun Hamza, Benchekroun Hamza and Ait lahmouch Nadir

February 9, 2021

1. Main objectives and scope of the assignment

Our goals from this assignment are:

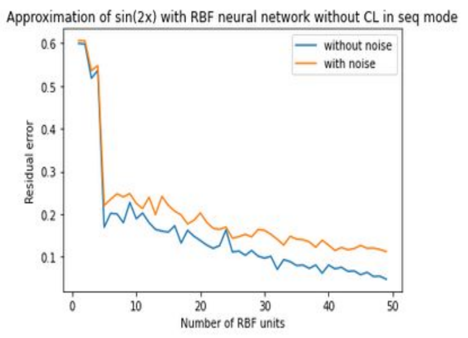
- Build RBF network from scratch and use them in regression
- Compare RBFs and MLPs.
- Study the impact of RBF centers initialisations.
- Implement the SOM algorithm and learn how to visualize the outputs.

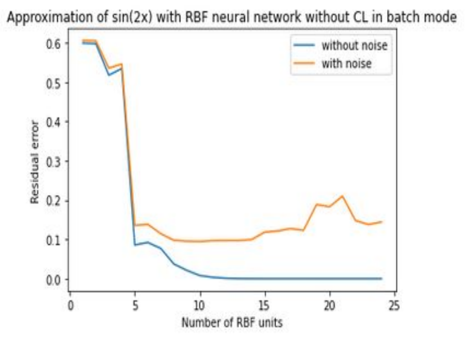
2. Methods

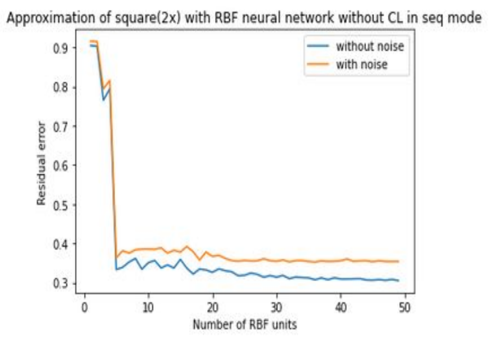
In this Lab, we used **Python** and the **numpy** library for the matrix manipulations and operations. Some **Matlab** was used for the second part.

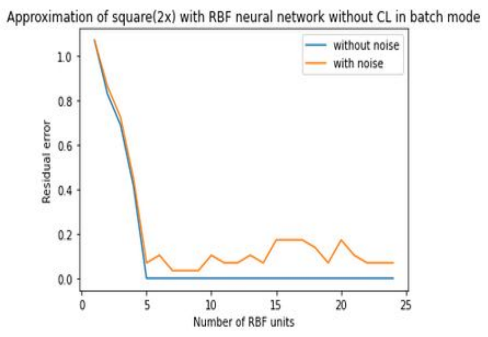
3. Results and discussion - Part I: RBF networks and Competitive Learning

3.1 Function approximation with RBF networks



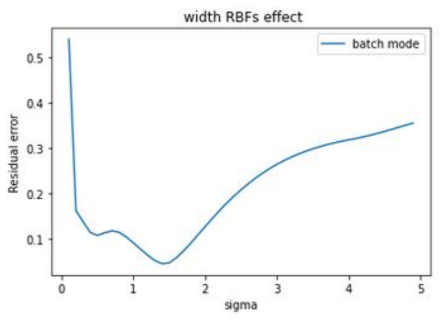






From the plots above we conclude:

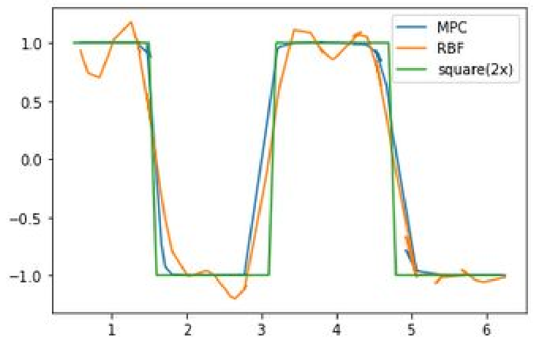
- For noisy data the error stagnates in a higher value for both sin and square functions in both modes.
- By using the transformation (output = 1 if output >= 0 and output = 0 otherwise) RBF neural networks need only 5 RBF units to reduce the residual error to 0. In general this transformation is used for functions with discrete values.
- Sequential learning mode has higher error than batch mode. (it is also computationally more expensive than batch mode).

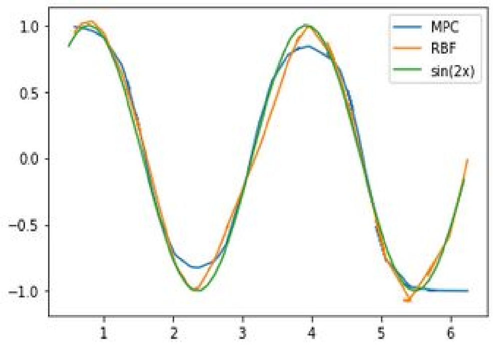


Sigma affects the response space of hidden neurons. Too high or too low values of sigma inhibit good function generalization, which can be seen from the plot above. Sigma chosen should be larger than the distance between adjacent input vectors, but smaller than the distance across the whole input space.

MPC trained on	**sin(2x) error**	**square(2x) error**
clean data	0.1038 (sd = 0.0069)	0.0991 (sd = 0.1338)
noisy data	0.1565 (sd = 0.0027)	0.2064 (sd = 0.1291)

RBF	**sin(2x) error**	**square(2x) error**
clean data	1.155×10^{-5}	0.1934
noisy data	0.1226	0.2353



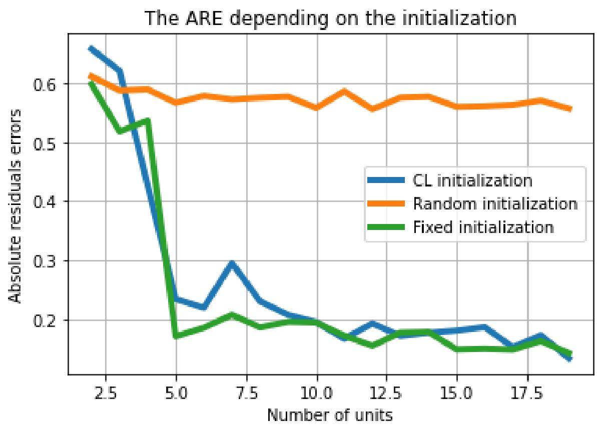


- RBF performs better for sin(2x) approximation
- without transformation MPC performs better for square(2x) approximation.
- with transformation MPC and RBF give the same results.

3.2 Competitive learning for RBF unit initialisation

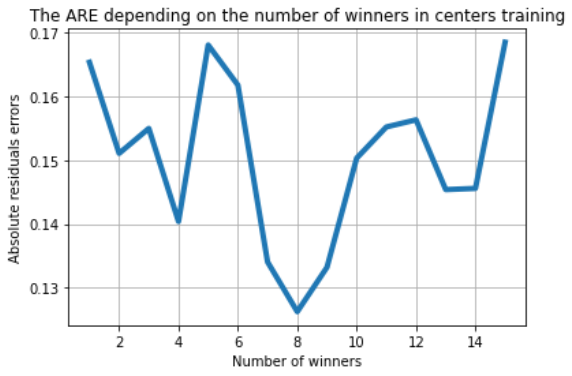
a) Competitive learning vs previous approaches for function approximation

Here, we will compare between the CL learning with the Random initialization and our initialization which consists of uniformly distributed centers.



The graph above shows the Absolute residual error depending on the number of units for all three initializations. As expected, the CL yields better results than the Random one. However, it's not performing better than our fixed initialization and that's normal because the learning can't perform better than the uniformly distributed RBF units.

Let's check now, whether the number of winners at each iteration improves the results.

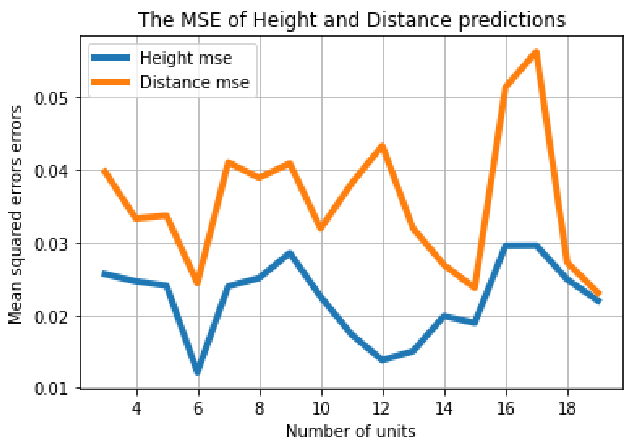


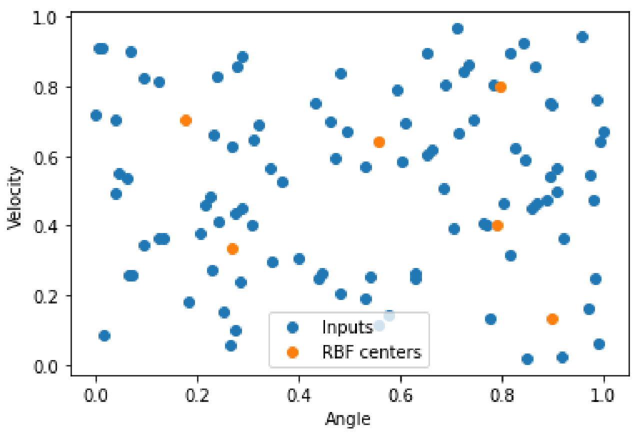
As we can see in the graph above, there's no clear correlation between the number of units and the error given. But it seems that for a certain configuration, a number of winners can be more effective than others.

Learning	**Error (data without noise)**	**Error (noisy data)**
CL, 1 winner	0.16 (sd = 0.015)	0.18 (sd = 0.02)
CL, 4 winners	0.22 (sd = 0.045)	0.23 (sd = 0.03)
Random initialization	0.56 (sd = 0.0065)	0.56 (sd = 0.0056)

In the table above, we gather the results of applying the delta rule learning with a learning rate of 0.01, 17 RBF units for the *sin(2x)* problem. We can conclude that the adding the number of winners doesn't necessarily improve the error.

b) Approximation of two dimensional function





The image on the right shows the centers of RBF units on the input space using Competitive Learning with one winner. We can say from the figure above that there is no apparent correlation between the number of RBF units and the performance of the network. One explanation would be the nature of the data (image on the right) which seems very difficult to cluster.

4. Results and discussion - Part II: Self-organising maps

1- Topological ordering of animal species

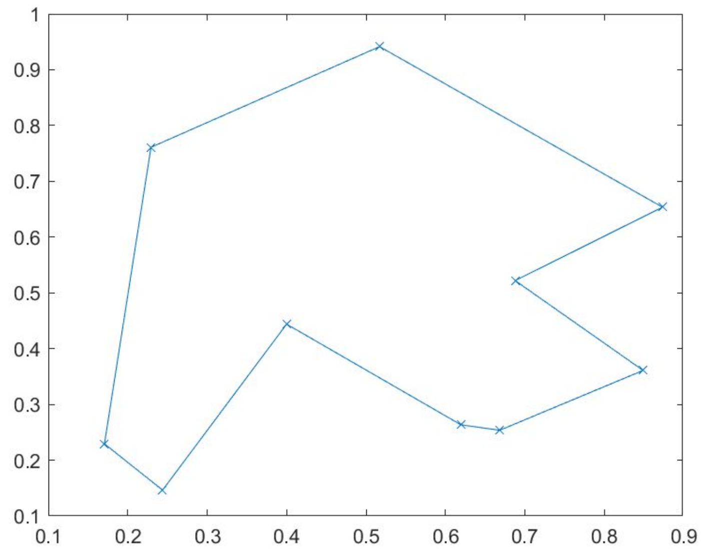
Below is the table that shows the embedding of a 84 dimensional space containing the animals' characteristics into a one dimensional space, using SOMs.

We can see that this one dimensional output separates insects (output<=20) from cats (cat, lion). The results however are generally poor, a 2 dimensional embedding might have yielded better results because it's not oversimplified.

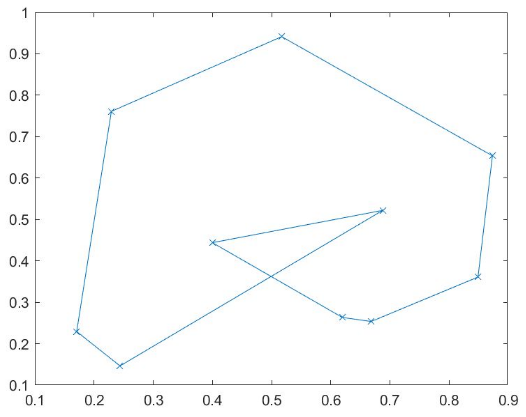
'dog'	1	'ape'	20
'grasshopper'	1	'pig'	20
'horse'	2	'skunk'	22
'giraffe'	3	'bat'	24
'housefly'	4	'hyena'	26
'duck'	5	'ostrich'	28
'dragonfly'	6	'penguin'	31
'moskito'	6	'kangaroo'	34
'seaturtle'	8	'crocodile'	36
'camel'	10	'frog'	39
'rabbit'	10	'elephant'	42
'bear'	12	'cat'	45
'beetle'	14	'spider'	49
'rat'	15	'pelican'	53
'butterfly'	16	'antelop'	58
'walrus'	18	'lion'	65

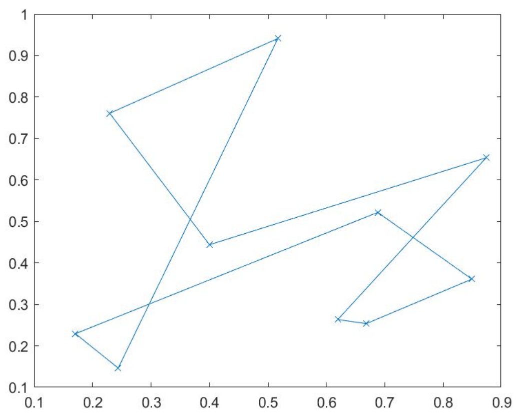
Cyclic Tour

Using SOMs, we produce a two dimensional output to find the optimal way for a cyclist to visit all the cities while minimizing the total distance. We distinguish however between two cases : random weight initialization and deterministic initialization w_0 = (0.5, 0.5, ..., 0.5)



weight initialization using w_0; distance=2.778215



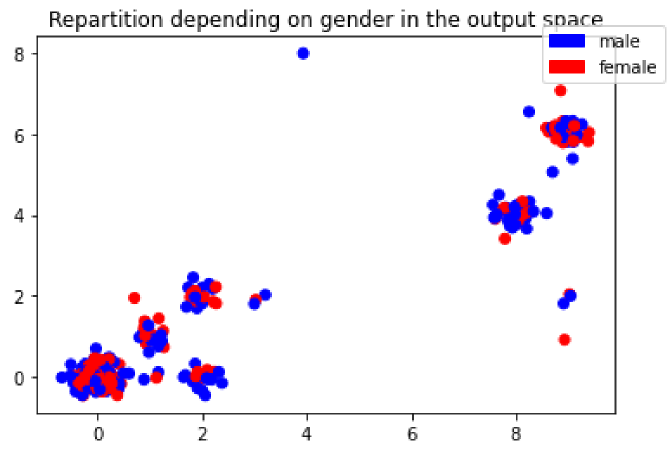


random weight initialization; left (distance=2.778215)/right(distance= 3.715785)

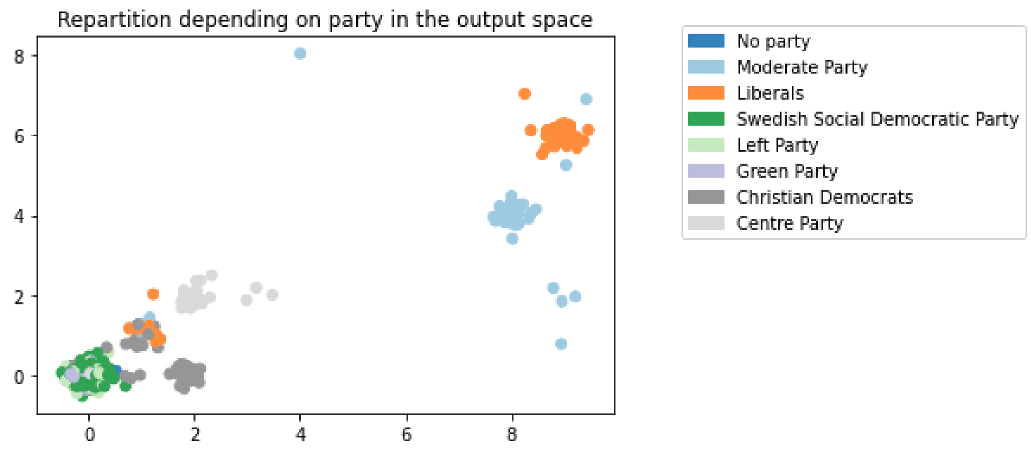
We notice that the distance is minimal using w_0, and varies a lot depending on the random initialization. Varying the number of epochs using random initialization doesn't help, which simply means that SOM seems to converge to a local minima.

Vote of MPs

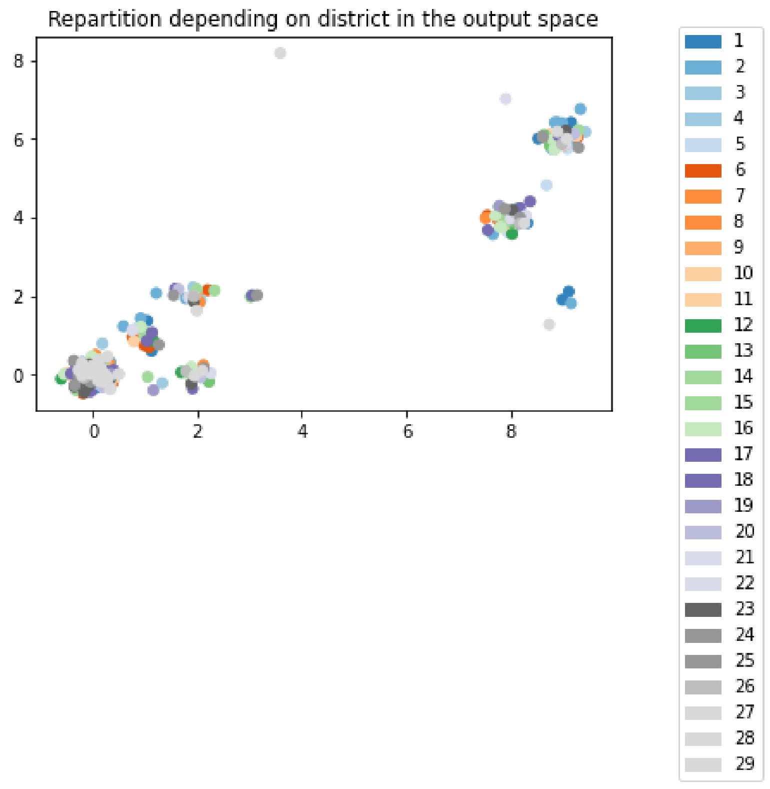
Once again we apply SOMs to produce a two dimensional output, by putting MPs in a 10x10 grid using their votes.



Repartition depending on gender in the output space
male
female



Repartition depending on party in the output space
No party
Moderate Party
Liberals
Swedish Social Democratic Party
Left Party
Green Party
Christian Democrats
Centre Party



Repartition depending on district in the output space

The plotting shows a clustering of votes, we conclude that:

- wrt to gender and districts, there is no apparent correlation.
- wrt to the party, there is clustering, with the liberals and the moderate party being closer to each other, and far away from the christian democrats for example.

5. Final remarks

This lab was very helpful because it allowed us to understand new notions which are the RBF and SOM networks, and how they can be used for regression and clustering. The comparison between MLP and RBF allowed us to understand what algorithm works better in what situation (e.g. MLP performs better for noisy data). Also, dealing and building SOM networks allowed us to understand how we can transpose problems to other dimensions to facilitate the understanding and the visualization of the outputs.