Kungliga Tekniska Högskolan

DD2437 Artificial Neural Networks and Deep Architectures

Lab 2: Radial basis functions, competitive learning and self-organization

Quentin LEMAIRE
Quentin VECCHIO
Kévin YERAMIAN

February 22, 2018

1 Introduction

In this lab the focus will be on unsupervised neural network approaches that involve competitive learning and the related concept of self-organization. In this context, you will also experiment with Radial-Basis Function (RBF) networks, which incorporate both unsupervised and supervised learning to address classification and regression tasks. The second part of the lab is devoted to the most famous representative of self-organizing NNs - Kohonen maps, commonly referred to self-organizing maps (SOMs). These networks map points in the input space to points in the output space with the preservation of the topology, which means that points which are closed in the input space (usually high-dimensional) should also be closed in the output space (usually low-dimensional, i.e. 1D, 2D or 3D). These networks can be used to help visualize high-dimensional data by finding suitable low-dimensional manifolds or perform clustering in high-dimensional spaces. In both cases the objective is to organize and present complex data in an intuitive visual form understandable for humans.

1.1 Aim and scope

The goal of this lab is to learn how to design RBF neural network and implement the core algorithm of SOM. After, we used our knowledge for three different tasks. The first was to order objects (animals) in a sequential order according to their attributes. The second was to find a circular tour which passes ten prescribed points in the plane. The third was to make a two-dimensional map over voting behaviour of members of the Swedish parliament. In all three cases the algorithm is supposed to find a low-dimensional representation of higher- dimensional data.

During this lab we learned to :

- know how to build the structure and perform training of an RBF network for either classification or regression purposes.
- be able to comparatively analyze different methods for initializing the structure and learning the weights in an RBF network.
- know the concept of vector quantisation and learn how to use it in NN context.
- be able to recognize and implement different components in the SOM algorithm.
- be able to discuss the role of the neighbourhood and analyze its effect on the self-organization in SOMs.
- know how SOM-networks can be used to fold high-dimensional spaces and cluster data.

2 Assumptions and tools

We used the library *sklearn* to compute a MLP network and to compare it to the RBG network.

We also used *matplotlib* for all our plots.

3 Results

3.1 Part I

3.1.1 Batch mode training using least squares - supervised learning of network weights

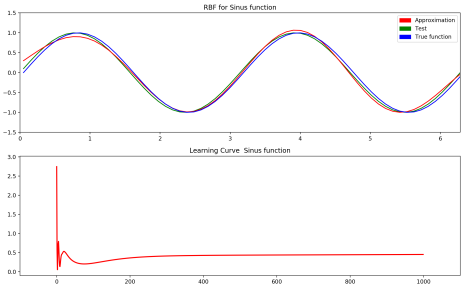
We initialize the weight of the network with the least-square solution, μ_i are distributed all over the range with the same distance to each other and σ_i is a constant equal to $\sqrt{\frac{2\pi}{numberofhiddennodes}}$. The learning rate is 0.2 and the number of epochs is 500 for each case. Our algorithm is deterministic so the number of nodes we found depends of this initialization.

For the sinus approximation on-line we find 8, 12 and 18 For 0.1, 0.01 and 0.001 error respectively.

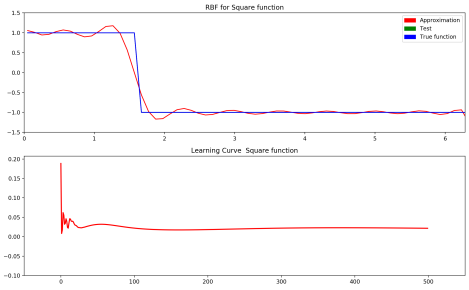
For the square approximation on-line we find 5, 10 and 25 For 0.1, 0.01 and 0.001 error respectively.

For the sinus approximation with batch = 50 we find 6, 12 and 50 For 0.1, 0.01 and 0.001 error respectively.

For the square approximation with batch = 50 we find 6, 8 and 12 For 0.1, 0.01 and 0.001 error respectively.

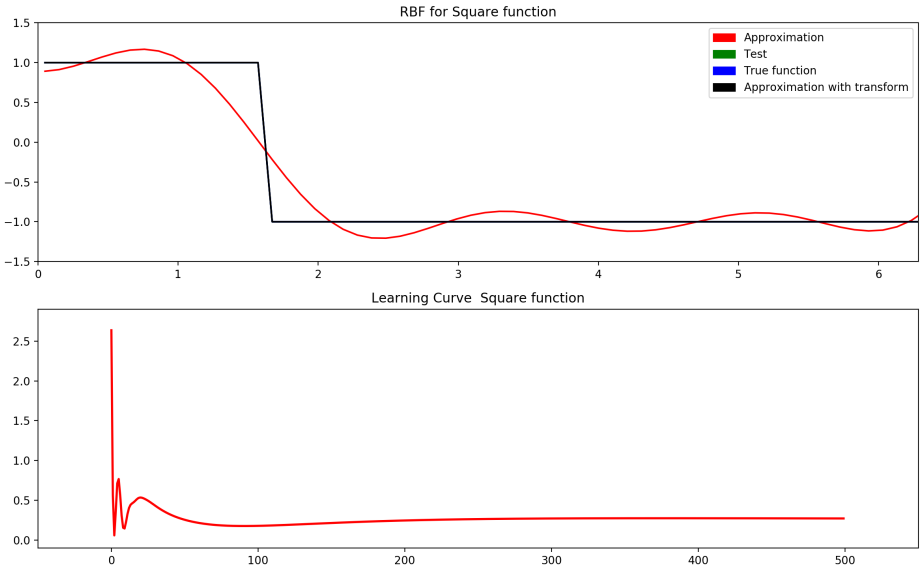


(a) Approximation of sinus with 7 nodes and lr=0.2



(b) Approximation of square with 20 nodes and lr=0.2

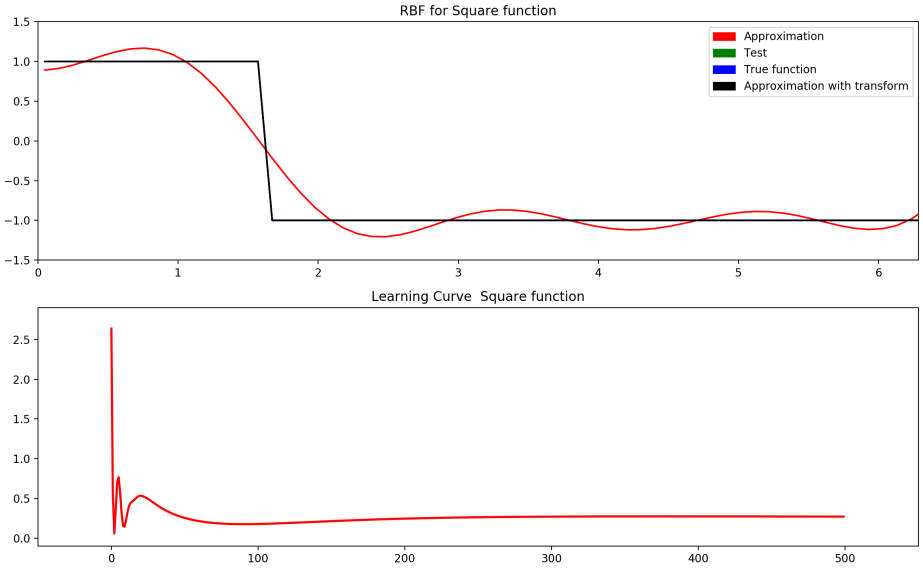
A way to get reduce the error to 0 is to apply a threshold that output -1 if the result is negative and 1 otherwise. We need 7 nodes to do it. We use this transformation in signal theory. Indeed, to communicate we send negative and positive intensity to encode 0 and 1 bit and during the communication there is noise so we use this transformation to decode the bit.



(c) Approximation of square with 7 nodes and lr=0.2

3.1.2 Regression with noise

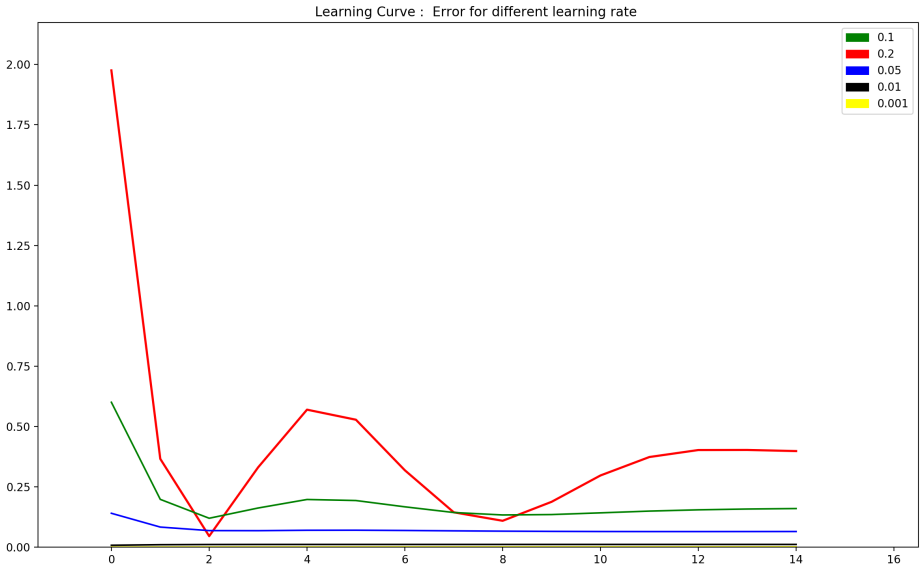
For the sinus approximation with noise we find 6, 11 and 26 For 0.1, 0.01 and 0.001 error respectively. The number of nodes is almost the same except for the last one.



(d) Approximation of square with 7 nodes and lr=0.2

The learning rate is more important in this case. Indeed, we need to decrease

the learning rate due to the noise. The noise adds a difficulty and if we keep the learning high we will oscillate.



(e) Learning Curve

The RBF nodes positioning is very important. Closer a point is to a node, more important will be the impact of the error of this point. So if we initialize randomly all nodes in many cases we will give "importance" to an outlier point or not give importance to a dense area of points. Our strategy is to distribute uniformly the node on the range. Below a graphic that show the important of positioning nodes.

(f) Difference of initialization of Mu

Width is a parameter that controls the importance you give to the area around the μ for each Gaussian. If the width is low so you will divide by a lowest number so the Gaussian will increase. You give more importance to a bigger area. On the contrary, if you fix the width very high you will give low importance to areas far from the center of the Gaussian.

The learning has an important influence in this case if we have a learning too high, we will oscillate and if we take a learning rate too low the rate is too slow. The number of nodes is also important. Indeed, if we set the network with too much node we will overfit the noise.

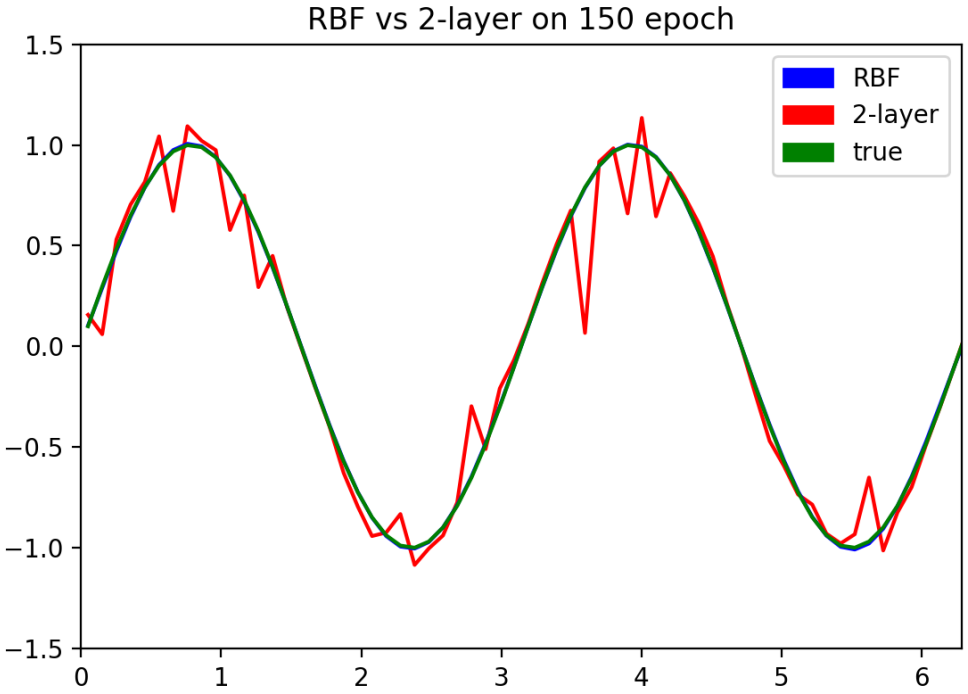


Figure 1: Comparaison between the RBF and MLP method

We have also compared the result with a MLP for the same number of nodes.

The results are clearly better with the RBF than for the MLP but the computation time is worse:

MLP time = 0.04856500000000008

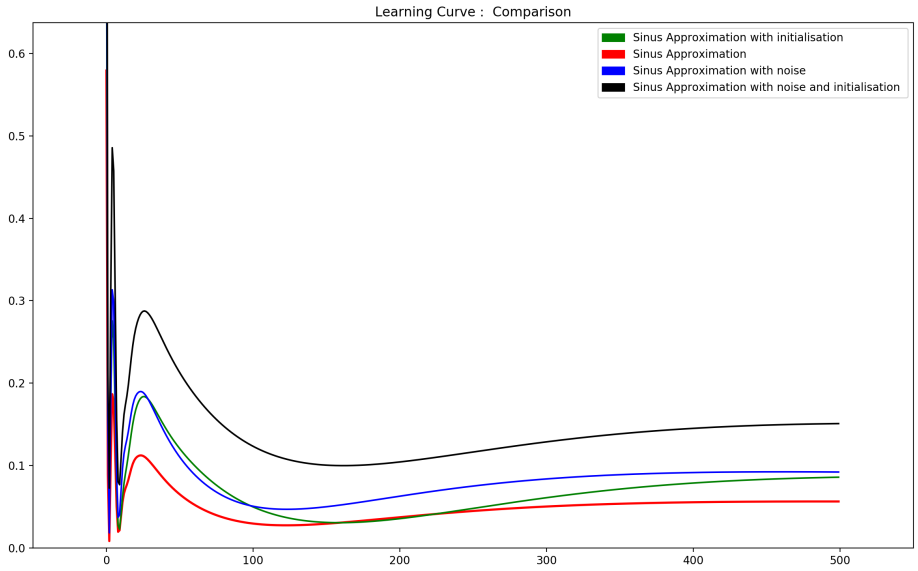
RBF time = 0.23814199999999996

To get a better result with a MLP, we would need much more nodes.

3.1.3 Competitive learning for the initialisation of RBF units

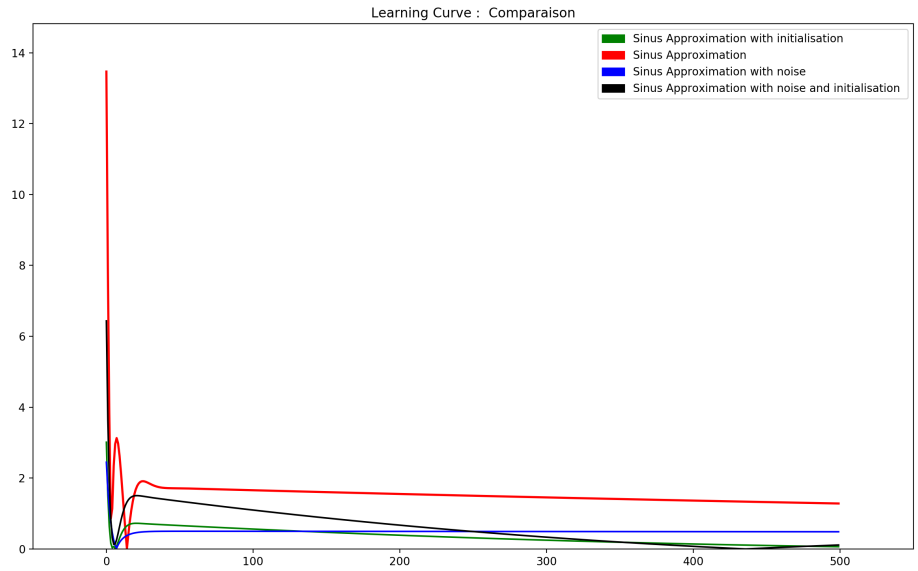
Compare the CL-based approach with your earlier RBF network where you manually positioned RBF nodes in the input space. Use the same number of units (it could be the number of units that allowed you to lower the absolute residual error below 0.01). Make this comparison for both noise- free and noisy approximation of sin(2x). Pay attention to convergence, generalisation performance and the resulting position of nodes.

In this case, the competitive learning converge to a uniform distribution and it makes sense because our points are uniformly distributed over a range. The performance of the CL is lower than our manual initialization (exactly distribute).



(a) Comparison

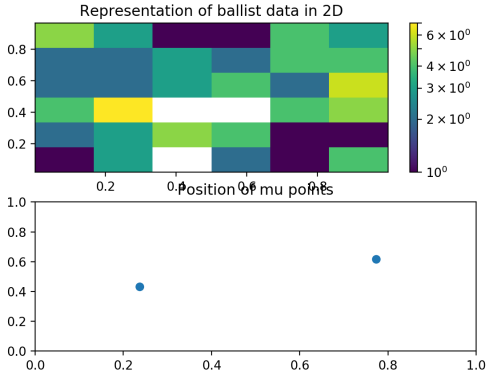
In another case, the inputs data are not distributed uniformly so the CL should work better.



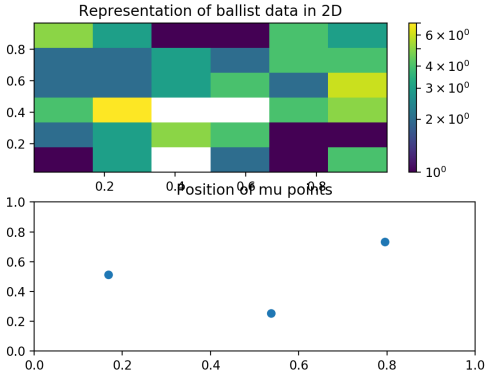
(b) Comparison

To avoid dead units we use the Frequency Sensitive Competitive Learning (LSCL). We multiply by the square of the number of times we select the units divisdd by the number of steps. The more you are selected, greater is the coefficient so greater is the distance.

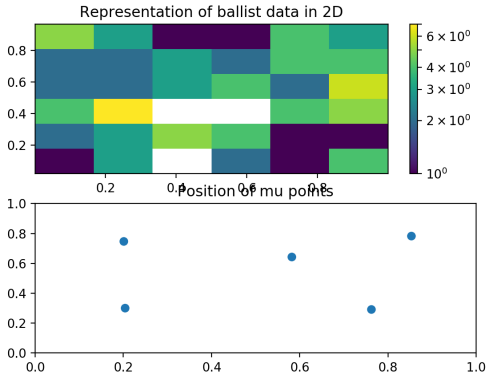
Let apply this technique on the ballistic dataset.



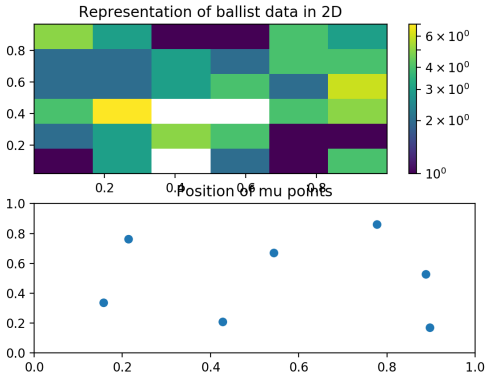
(c) LSCL on ballistic with 2 units



(d) LSCL on ballistic with 3 units



(e) LSCL on ballistic with 5 units



(f) LSCL on ballistic with 7 units

3.2 Part II

3.2.1 Topological Ordering of Animal Species

For this part, we use a SOM to make a topological ordering of animal species given, for each species, 84 features. After several executions, we saw that SOM ordered our different species well. In the array below, we can see the 3 distinct groups, insects, oviparous and mammals deducted by the ordering.

Species	Animals
Insects	beetle butterfly dragonfly grasshopper housefly moskito spider
Oviparous	duck pelican ostrich penguin crocodile frog seaturtle
Mammals	ape walrus bat rabbit rat hyena skunk dog kangaroo cat lion bear horse antelop elephant giraffe camel pig

3.2.2 Cyclic Tour

In this part, we used SOM to resolve a TSP problem given 10 cities. The SOM did a matching between the 10 inputs and the 10 outputs. We used the same SOM as in part 4.2, but with a circular selection of neighbors. First, we tried to resolve the problem without competition learning and with 100 epochs and a learning rate of 0.1. The weights of the output were initialized randomly. As you can see below, results are really bad. The outputs didn't converge.

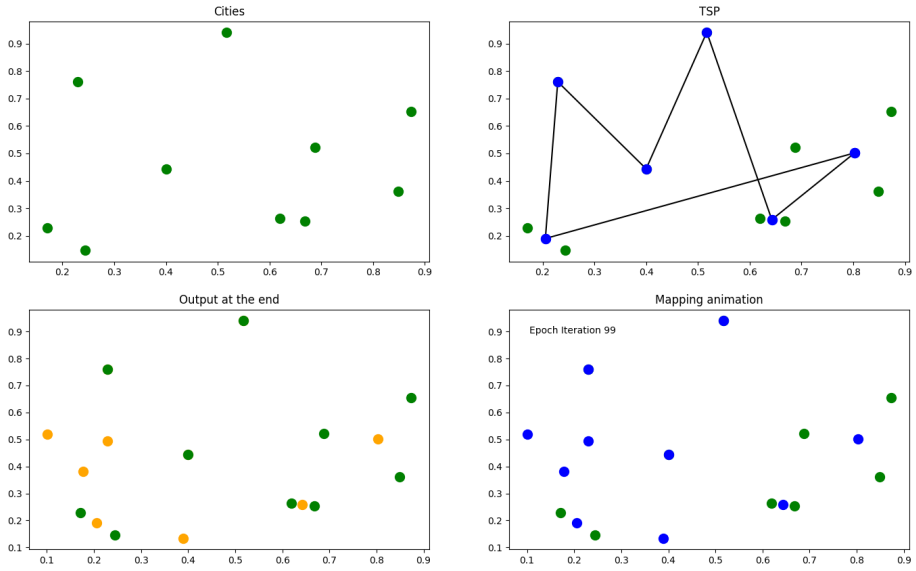


Figure 2: SOM Result, without CL, epochs=100 and lr = 0.01

Next, to improve our results, we use the competition learning for the matching. With this improvement we had better results as you can see below.

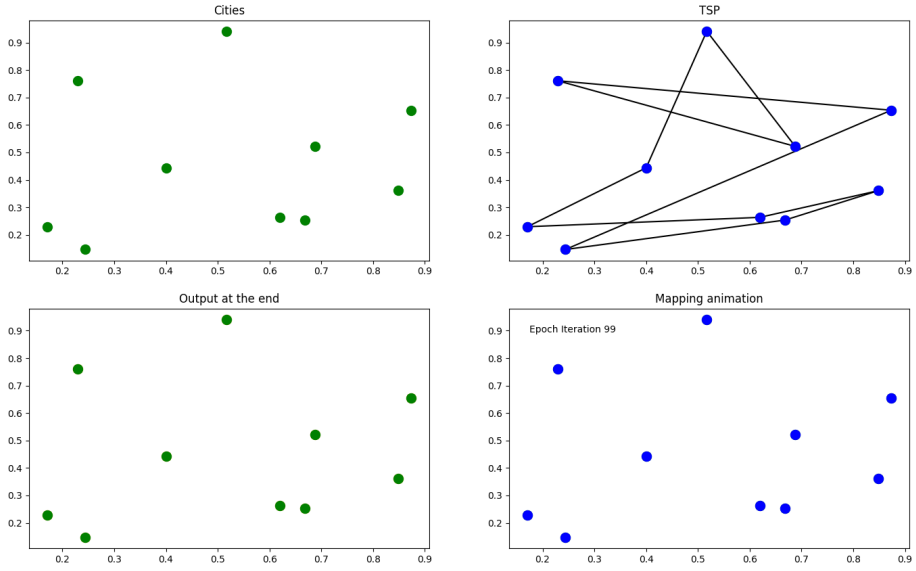


Figure 3: SOM Result, with CL, epochs=100 and lr = 0.01

Finally, we tried a last improvement with the initialization of the weights. We placed initially the outputs as a ring around the inputs. With this improvement,

the convergence is much better. We have almost the optimal path.

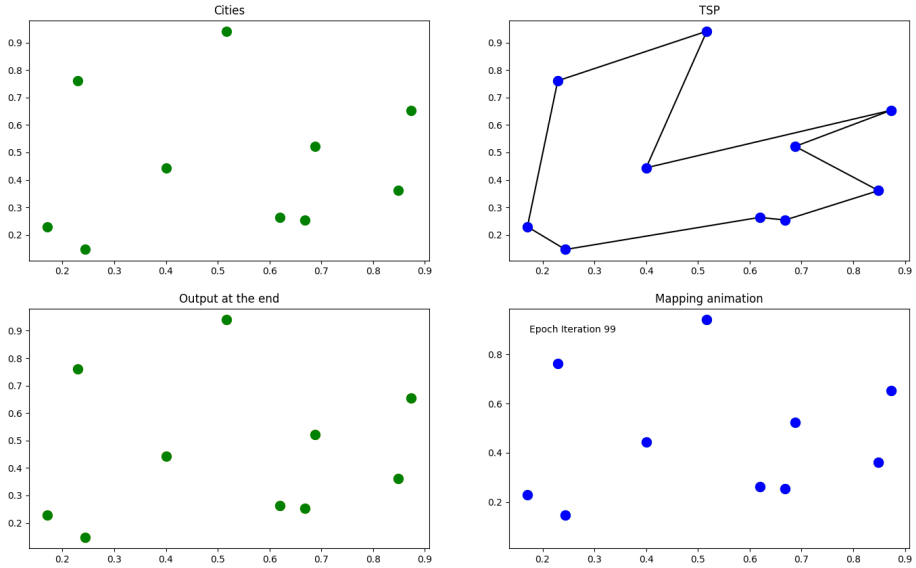


Figure 4: SOM Result, with CL, epochs=100, lr = 0.01 and weights optimization

We also tried with a different number of outputs and as you can see, if the number of outputs is larger than 10, we have the optimal path.

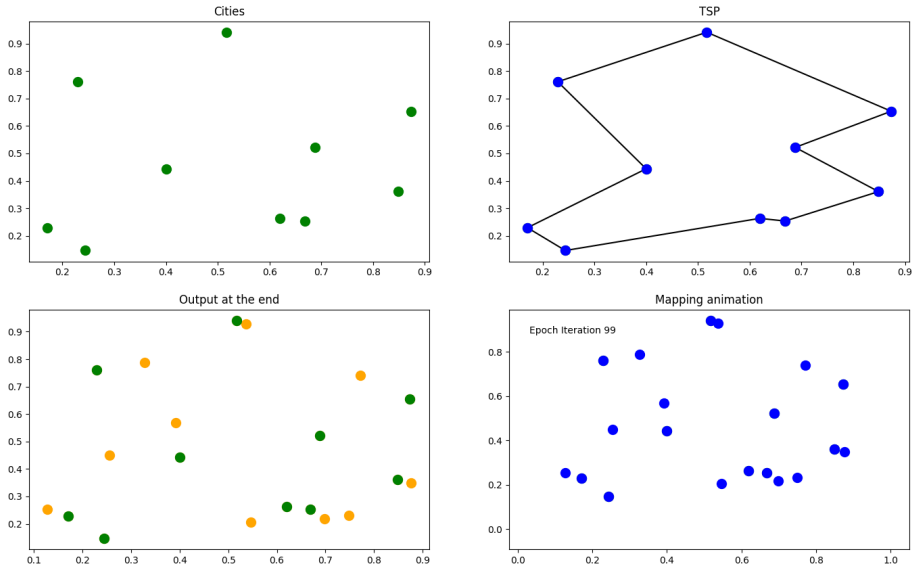


Figure 5: SOM Result, with CL, epochs=100, lr = 0.01, weights optimization and 20 outputs

3.2.3 Data Clustering: Votes of MPs

We then used the SOM to cluster the MPs of Sweden according to their votes between 2004-2005.

There are 349 MPs and we have the records of 31 votes.

The goal is to place them in a 10*10 map and, therefore, to observe clusters of, hopefully, the same party. Parties with the same ideology should also be close to each other.

Moreover, we want to see if there is any link between the votes of the MPs and their sex or district.

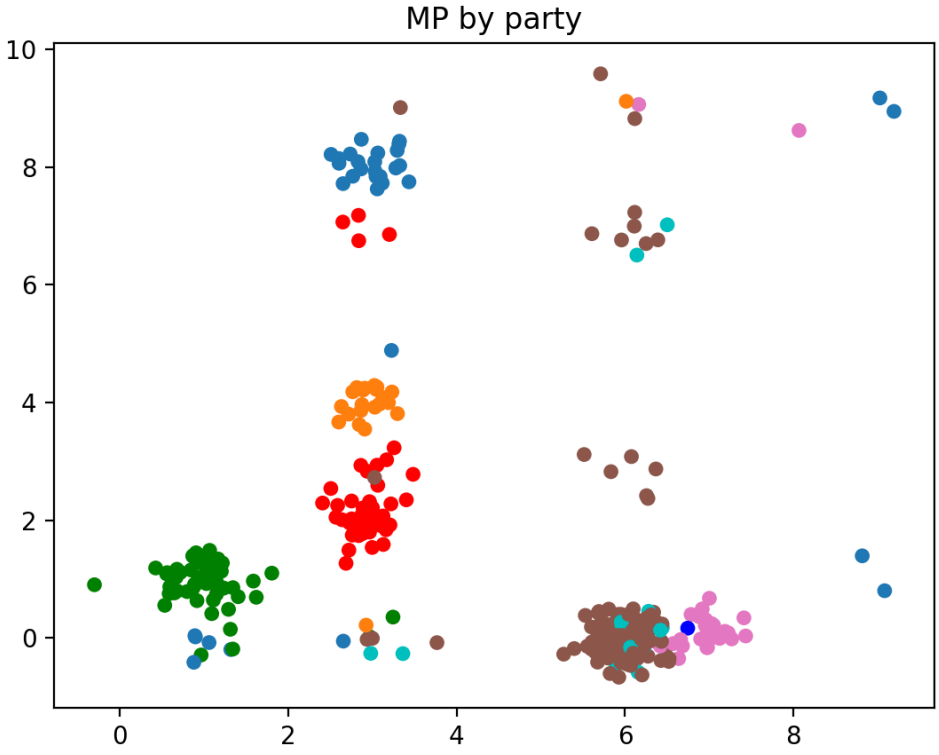


Figure 6: MPs in the 10*10 map colored by their party.

In the figure 6, we have placed the MP in the calculated map and we have colored the points according to their party. We used the following colors:

- pure blue = no party
- green = Moderata samlingspartiet
- red = Liberalerna
- brown = Socialdemokraterna

- pink = Vänsterpartiet
- cyan = Miljöpartiet de gröna
- pale blue = Kristdemokraterna
- orange = Centerpartiet.

We have also added the value of a Gaussian of mean 0 and of variance 0.25 to each point to make them sparse and to have a better visualization.

We can see clear clusters of the same party in the figure so the MPs are very well separated by the SOM algorithm.

There are two coalitions in Sweden: Alliansen and De rödgröna.

The first is composed of the green, orange, red and blue parties. And the second of the brown, cyan and pink parties.

We are able to identify those coalitions in the figure, the Alliansen coalition is at the left of the figure and the De rödgröna at the left.

Therefore, the voting patterns which are similar end up close to each other.

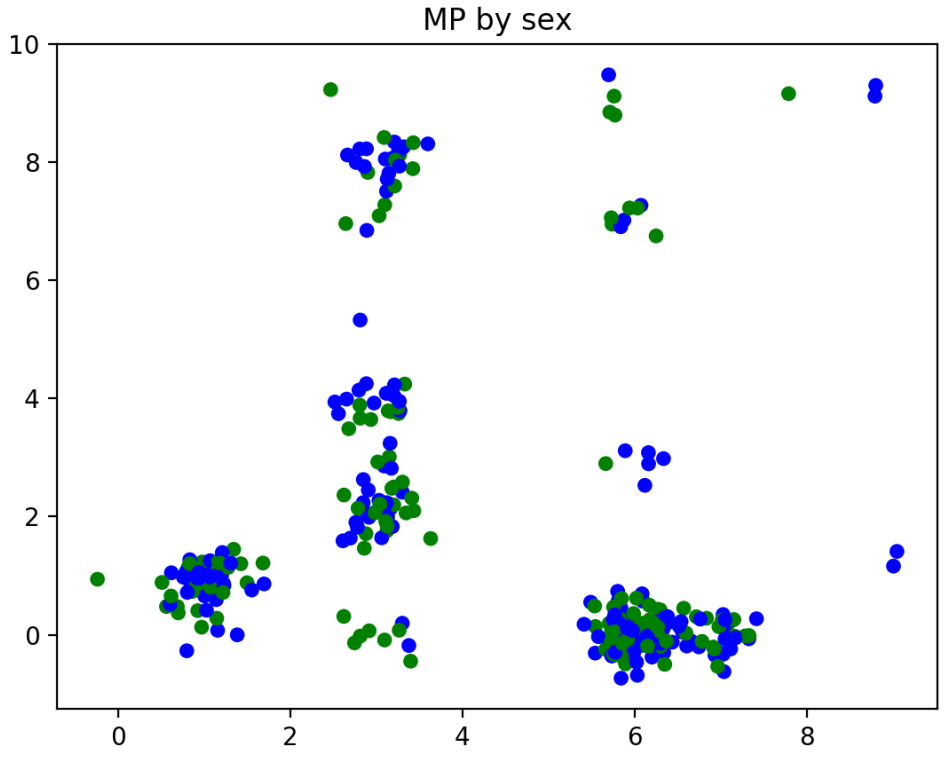


Figure 7: MPs in the 10*10 map colored by their sex.

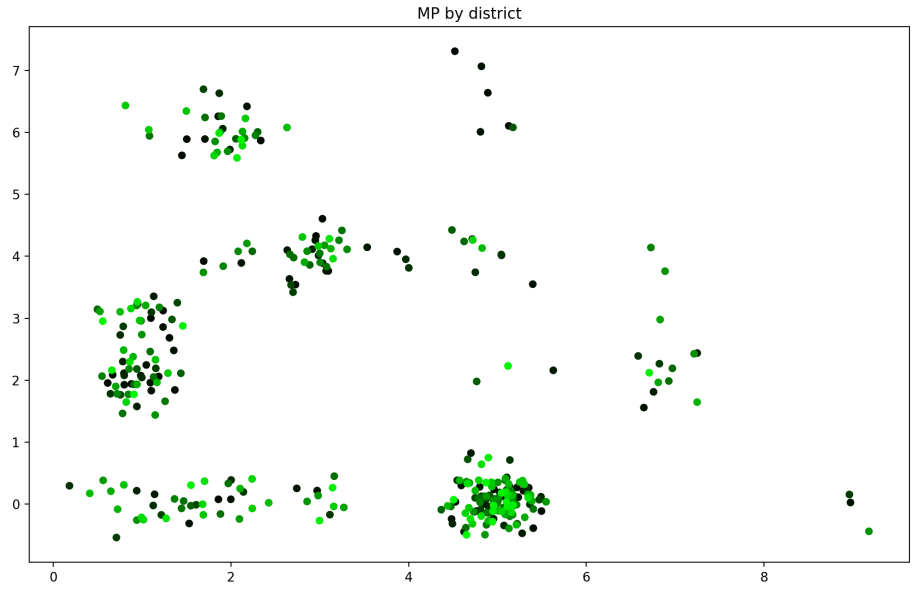


Figure 8: MPs in the 10*10 map colored by their district.

In the figures 7 and 8, we have plotted the map by colored according to their sex and then their district.

We are not able to see any correlation between the position on the map and the sex or district of the MP. Therefore, we can say that the division seems equal.

4 Reflections, open questions and conclusions

Over this assignment, we have seen that the RBG network is a powerful method that can be used in a lot of cases. For some of them, it can be really resistant to the noise and it can even perform better than a traditional MLP method.

It can also be used with the competitive learning in order to find good positions for the nodes.

Then we have studied SOM algorithm in different applications.

We achieved to resolve a problem similar to the salesman problem.

We have also seen that the SOM algorithm is useful to extract patterns and correlation from a very high dimensional input. We achieved to get good clusters from the votes of the Swedish MPs which was a 349*31 matrix. And we used it to animal classification.

At the end, all these data were represented in a lower dimension.