Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and

self-organisation

Samuel Leonardo Gracio and Martin Verstraete

#### **September 26, 2018**

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

List here a concise list of your major intended goals, what you planned to do and what you wanted to learn/what problems you were set to address or investigate, e.g.

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

*•* to perceive the interest of Self Organizing Maps

*•* to understand how the RBF units can improve the learning

*•* to compare different Competitive Learning strategies

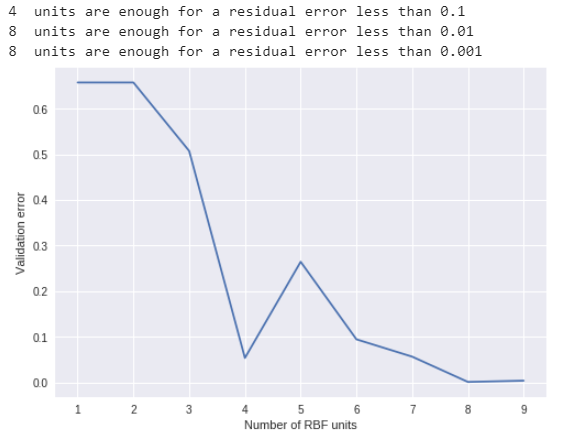
# **2 Methods**

We worked with Python and with a Google Colab environment. We only used numpy functions and matplotlib.

# **3 Results and discussion - Part I: RBF networks and Competitive Learning *(ca. 2.5-3 pages)***

### **3.1 Function approximation with RBF networks**

In the absence of noise, the absolute residual error decreases as following (using the LMS in batch mode):

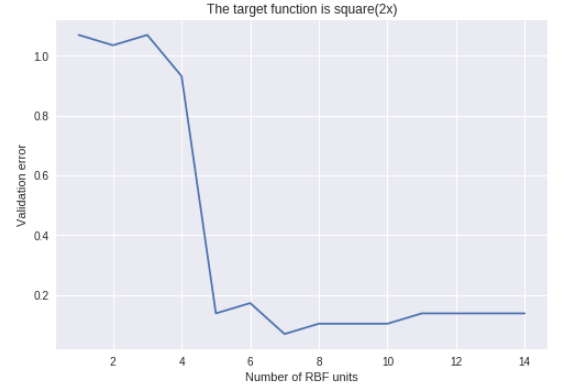
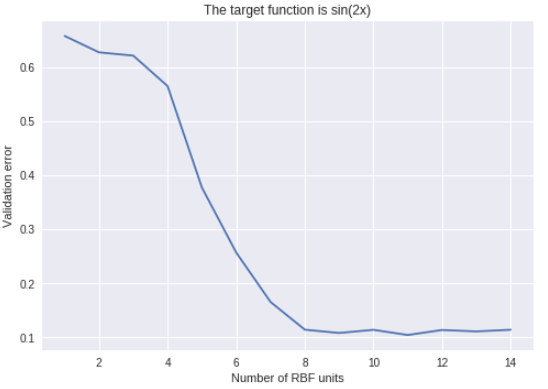


*Figure 1: Sin(2x) Figure 2: Square(2x)*

Thus, the more units, the less the absolute residual error.

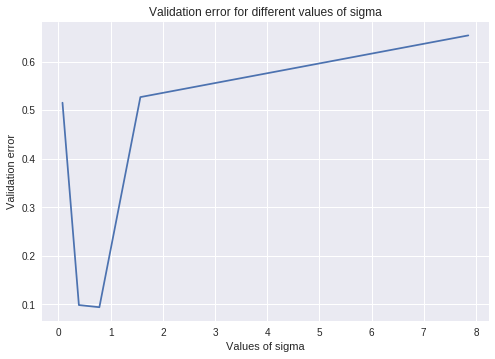
With noise, we studied the difference between delta rule in batch and sequential mode for both functions and the impact of several parameters.

* Delta rule in batch mode



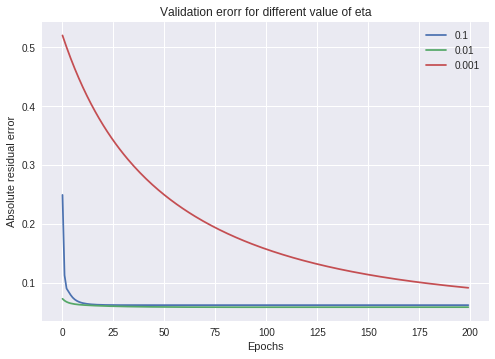
*Figure 3 Figure 4*

1. In both cases, 8 RBF units are enough to make the error small enough, let’s fix the number of RBF to 8 and study the impact of other parameters.
2. We computed the absolute residual error for different values of sigma and the following plot showed that a value of 2π/number\_of\_RBF was good. Bigger or smaller values can make the error diverge.



*Figure 5*

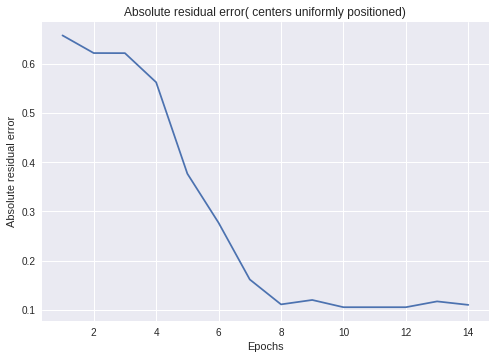
3. The following graph shows the influence of the value of eta.



*Figure 6*

Here the convergence is quicker if eta is smaller. In the three cases, the value of convergence is the same.

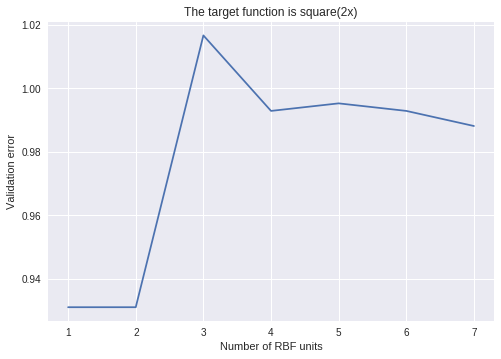
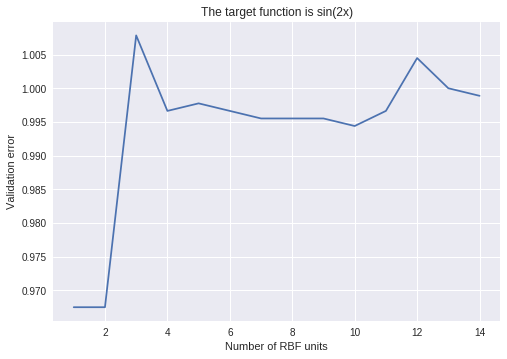
4. We then studied the position of the RBF nodes: either uniformly positioned between 0 and 2𝛑, or randomly positioned.



*Figure 7 Figure 8*

Uniformly positioning the RBF nodes does not seem to significantly reduce the validation error.

* Delta rule in sequential mode

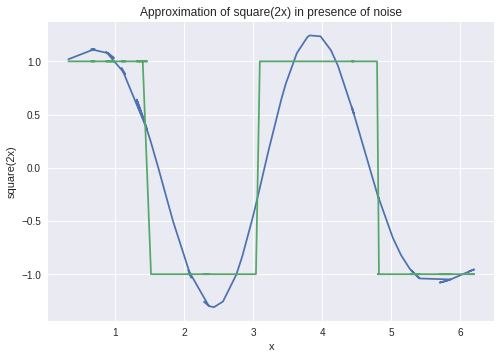
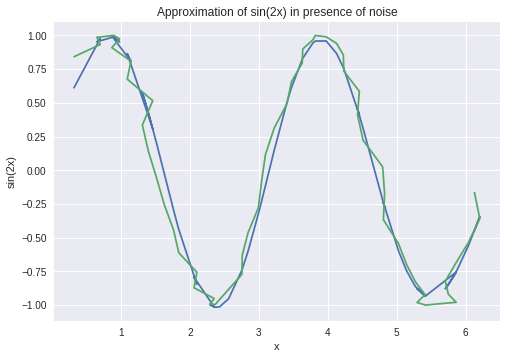


*Figure 9 Figure 10*

As shown by these figures, the delta rule in sequential is really bad to approximate the two functions. Thus, we will not discuss the impact of the parameters for the sequential case.

Now we compare the results of the approximation for our best RBF network trained with LMS algorithm in batch mode with a one-hidden-layer perceptron in batch mode.

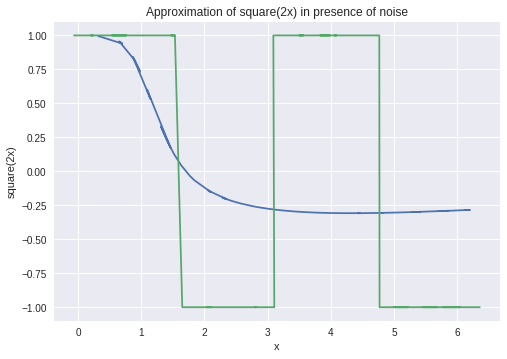
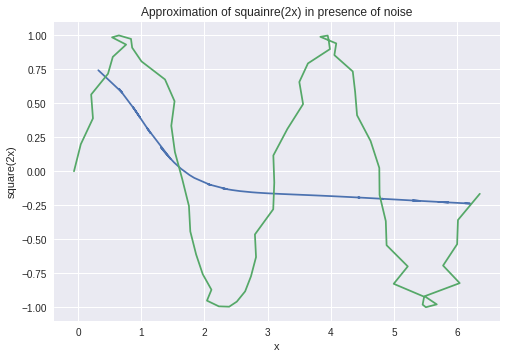
* RBF network with 8 RBF units and centers uniformly positioned



*Figure 11 Figure 12*

In both cases, the absolute residual error is around 0.1

* One-hidden-layer perceptron with 8 hidden nodes



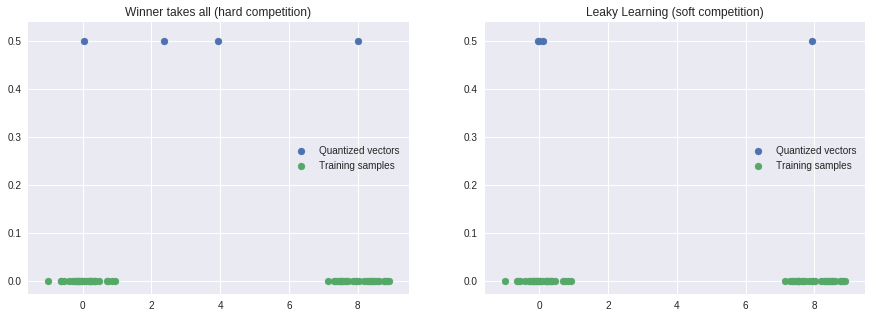
*Figure 13 Figure 14*

For sin(2x), the absolute residual error equals 0.41 and for square(2x) it equals 0.83

Obviously, the perceptron is bad at approximating these two functions.

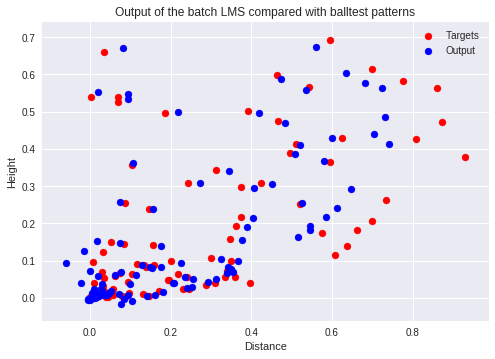
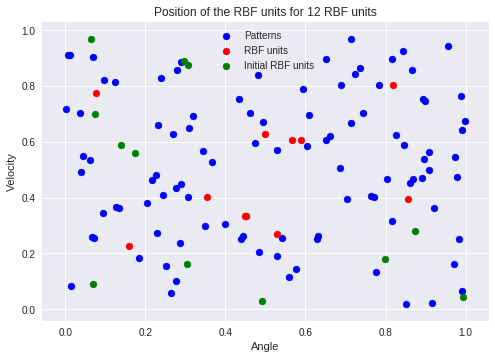
### **3.2 Competitive learning for RBF unit initialisation**

When we position the RBF units, we need to avoid dead units. So, we implemented a soft competition of competitive learning, these two figures illustrates the results in position of the RBF centers.

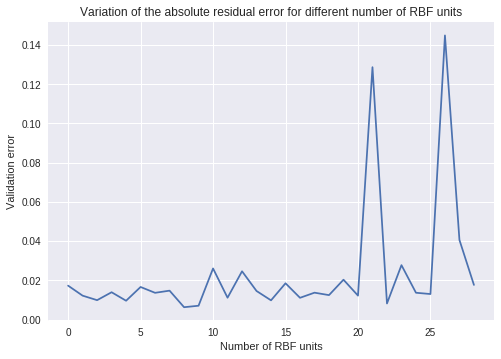


*Figure 15*

We can now apply our RBF network trained with the LMS algorithm and with RBF units according to a soft competition to the ballist and balltest data.



We can also see the influence of the number of RBF units on the value of the absolute residual error



We can see that just a few RBF units are enough to correctly compute the target points.

**4 Results and discussion - Part II: Self-organising maps *(ca. 2 pages)***

### **4.1 Topological ordering of animal species**

With the animals.dat dataset, we’re using the SOM algorithm on a one-dimensional curve in the 84-dimensional input space. First of all, we had to code the core algorithm. We did in Python, by implementing two functions : a core function to train the network and then a test function to calculate the index of the winning node and be able to visualise a list of animals. We used many parameters in this function, in order to be able to re-use it after, for the other parts. There are the following results of the SOM algorithm on the animals.dat file :

["'beetle'", "'dragonfly'","'grasshopper'","'moskito'","'butterfly'", "'housefly'", "'spider'","'duck'", "'pelican'", "'penguin'", "'ostrich'","'frog'", "'seaturtle'", "'crocodile'", "'walrus'","'bear'", "'hyena'", "'dog'", "'kangaroo'","'skunk'", "'bat'", "'elephant'", "'rabbit'","'rat'", "'ape'", "'cat'", "'lion'", "'horse'",”'camel'", "'giraffe'", "'pig'", "'antelop'"]

We can see that the list is coherent. Insects are grouped together, animals that produce eggs are together and mammals are also grouped together, which is what we wanted.

### **4.2 Cyclic tour**

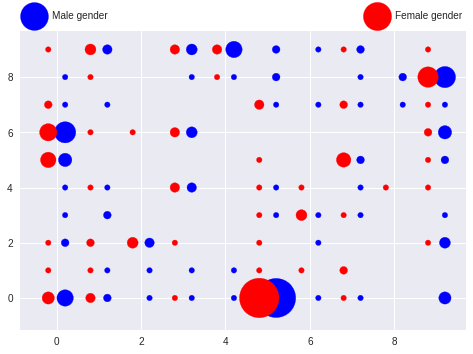
This time, we want to layout a curve in two dimensions : it has to pass by all or a maximum number of training points. The main difference between this problem and the last one is the shape of the input space, from 84 to 2. We also have to change our way of computing the neighbours : now, it needs to be circular, which means that when we compute the neighbours, we have to count the last and the first output nodes as neighbours.

We had a problem with the ‘cities.dat’ file, it wasn’t possible to open it with numpy.genfromtxt so we changed the format of the data without changing the data itself. After that, we used almost the same algorithm as the last part but changing the way of computing the boundary

|  |  |
| --- | --- |
|  |  |
| *FIG 18 : Beginning of the Algorithm. Epoch 1* | *FIG 19 : End of the algorithm, 100 epochs, 0.3 of learning rate.* |

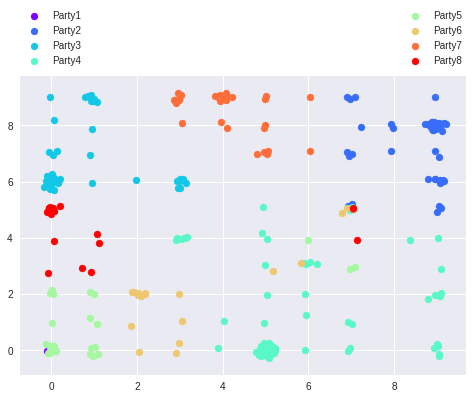
In facts, the route seems to be the same every time we run the code. As a consequence, it may be an optimal route. We can see in *Figure X2* that the tour try to pass by every points or in the middle of two points with a certain success rate. It gives a cyclic tour, we don’t pass two times on the same points, which is what we wanted with this algorithm. Moreover, it gives a good result in a little amount of time.

**4.3 Clustering with SOM**



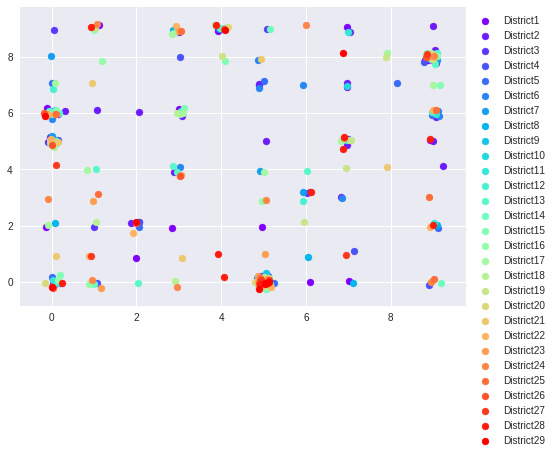
**Figure 20 : Male and Female repartition**

We can see in this figure that gender is not a criteria to vote differently. They are almost the same number of men and women voting for a left or right law. The gender is definitely not a good way to classify the data.



**Figure 21 : Different parties**

Here, we can see all the different parties votes. The main information here is that belonging to a party seems to be the main criteria to see what a member of the parliament will vote. In facts, we can see that are almost no outliers, the blue points form a group, the red one too, etc. The party is the main criteria, we can see the differents groups in term of left-right or even liberals agains anti-liberals.



**Figure 22 : Different districts**

Here, we can see the difference of votes in terms of districts.The points seems almost random : this means that the district is not a criteria to classify the data. By belonging to a certain district, we can’t now if you will be more in a right or left position.

**5 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.*

The SOM algorithm part was very interesting because it was really concrete. We could directly see the possibilities of the algorithm for several domains, from biology to politics. Since I would like to work in a company that applies Machine Learning to more casual domains like politics (or even Football !), this was really interesting.