Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Côme Lassarat, Victor Sanchez, Marie-Ange Stefanos

February 14, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to build the structure and perform training of an RBF network for either classification or regression purposes
- to recognise and implement different components in the SOM algorithm
- discuss the role of the neighbourhood and analyse its effect on the self-organisation in SOMs

2 Methods

In order to reach the intended goals, the different methods and algorithms were coded in Python, a wide-spread programming language. Several libraries were also imported and used:

- **Numpy** to handle multi-dimensional arrays and to generate linearly-separable and not linearly-separable data
- **Matplotlib** to construct and plot graphs
- **Pandas** to handle (load, concatenate, rename columns) *.dat* file in Dataframes.

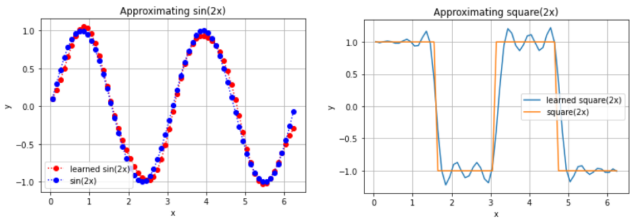
3 Results and discussion - Part I: RBF networks and Competitive Learning

3.0.1 Function approximation - without noise

The goal of this section is to approximate the $sin(2x)$ and $square(2x)$ functions using Gaussian RBFs networks. In order to do this, the segment $[0, 2\pi]$ is sampled in 63 points equally spaced. Thus, the centers of the Gaussians have also been, at first, equally spaced in this segment. The value chosen for σ is 0.5. The residual error of the approximation of both functions is computed.

sin(2x)		square (2x)	
Residual errors	Nb of nodes	Min residual error	Nb of nodes
$0.047365 < 0,1$	8	0.120002	58
$0.009505 < 0,01$	12	X	X
$0.000727 < 0,001$	20	X	X

Tabell 1

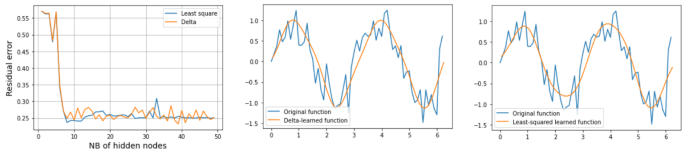


Figur 1: Left: best sin(2x) approximation with RBF network (20 nodes), Right: best square(2x) approximation with RBF network (58 nodes)

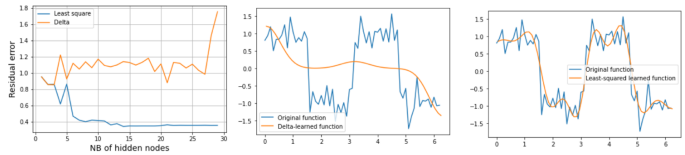
- $sin(2x)$ seems easy to approximate whereas one can notice oscillations to approximate $square(2x)$ (Figure 1)
- the residual error of $sin(2x)$ converges towards 0 when the number of hidden nodes increases, whereas the residual error of $square(2x)$ converges towards a value > 0.1. To reduce its error to 0, one can try to approximate the function piecewised by avoiding the discontinuities that seem at the origin of this error.

3.0.2 Function approximation - with noise

In this section, zero-mean Gaussian noise is added to the training set and the same tests than before are done. The centers of the Gaussian activation functions are, as a first step, equally distributed on $[0, 2\pi]$.



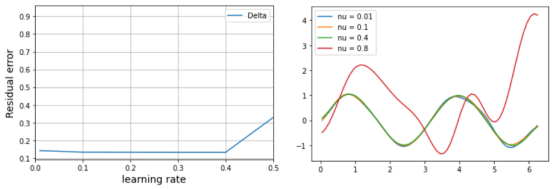
Figur 2: Left: residual errors (using test samples), Middle: best sin(2x) approximation with delta rule (39 nodes, 100 epochs), Right : best sin(2x) approximation using least-square method (8 nodes). Learning rate $\eta = 0.01$, $\sigma = 0.5$



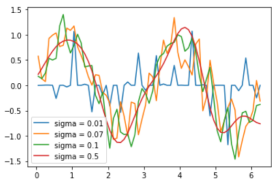
Figur 3: Left: residual errors (using test samples), Middle: best square(2x) approximation with delta rule (3 nodes, 100 epochs), Right : best square(2x) approximation using least-square method (14 nodes). Learning rate $\eta = 0.01$, $\sigma = 0.5$

- Figure 2: correct approximation even with noise, residual error according to the number of hidden nodes first decreases and then increases (overfitting), delta rule and least-square method approximately have the same accuracy
- Figure 3: troubles to approximate the square function with the delta rule, probably because of the discontinuities that are hard to handle for a derivatives-based algorithm. However, the meast-square method seems effective: prefer this method for non-continuous functions.

The influence of parameters over the approximation is also measured (Figure 4)



(a) Influence of the learning rate η on the approximation of sin(2x) (Nodes = 19, $\eta = 0.01$). Left: residual error according to η. Right: shape of the function approximated according to η



(b) Influence of σ (of the RBF units) on the approximation of sin(2x) (Nodes = 19, $\sigma = 0.5$)

Figur 4: Influence of different parameters on the approximation of sin(2x) by the delta rule (100 epochs)

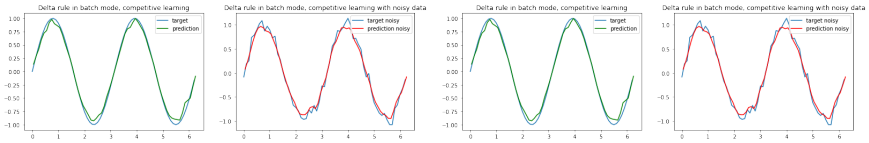
- Figure 4a: the residual error passes through a minimal according to the learning rate: need to choose this value.

- Figure 4b: as σ increases, the approximated function is more faithful to the original (similar curve also obtained for the least-square method). If σ is too small, the activation function is too restrictive.

Other tests have been conducted: the test performance has been estimated on the original clean data previously used in section 3.1.1: one can see the residual error decrease (decreasing bias) and then increase (overfitting).

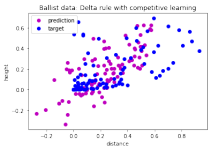
3.1 Competitive learning for RBF unit initialisation

Now the goal is to initialize the RBF node with an other method called Competitive learning (CL-based learning). This method consist of selecting a åinneråhich is the closest from the weight of each node.



Figur 5: CL Learning with clean and noisy data with an averaging over 50 iterations with (left) and without (right) dead unit avoiding

Dead node are nodes that are losing every time. A strategy to avoid them is to use them as a criterion of avoidance instead of using the winner. The goal is then to get away from loser instead of getting closer from winner. We obtain the results. We finally applied the competitive learning to Ballistic data that were provided. The goal here is to approximate a two-dimensional function : input (angle, velocity) ; output : (distance, height). Here are the prediction we have obtained.



Figur 6: Ballistic data prediction result

We used a noise with a zero mean and 0.1 variance. The residual error was 0.089 which is quite efficient.

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

For each task, the SOM algorithm is supposed to find a low-dimensional representation of higher-dimensional data. Then, sorting the array of the winner nodes and saving their indices (*argsort()* function from Numpy Python module) give the ordered cities.

4.1 Topological ordering of animal species

The data consists of the 32*84 matrix where lean 84 attributes for each row, that is each of the 32 animals. The names of the animals cna be found in an additional file names *animalnames.dat*. Here are more information about the parameters we used : the weight matrix has size 100x84, that is to say there are 100 nodes, the learning rate $\eta = 0.2$ and the number of epochs is 20 as suggested in the instructions. The neighbourhood size evolves through the epochs and has value $50 - 2.5 * epoch$ when *epoch* is the value of the current epoch index.

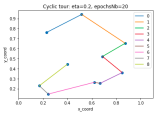
Figur 7: Topological ordering of animal species

Figure 7 shows that the animals are quite organized according to their attributes. For instance we can see that animals with wings are grouped together (penguin, pelican, ostrich, and duck), as are insects (spider, beetle, butterfly, housefly, moskito, dragonfly and grasshopper). However, we can also observe that some animal could have been better sorted such as the bat between the bear and the giraffe. The walrus between the horse and the elephant would have been better sorted with the other aquatic animals. The reasons of such errors can be due to several factors and can be improved using different method. The first one is to use better representative and relevant criteria than those attributes. The second one is the change the parameters, such as using a decreasing learning rate through the epochs, increasing the number of epochs or also changing the number of nodes. We can also note weights are randomly initialized so

the results are different from a running to an other. Some initialization weights will give better results than the others.

4.2 Cyclic tour

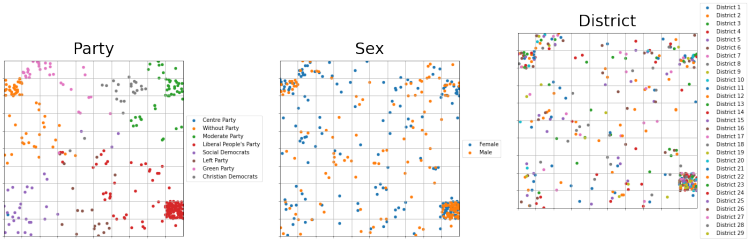
A good result obtained with the SOM algorithm can be seen on the figure 8. We can indeed observe that the tour is very close to what one would have drawn with a pencil. However, such good result is not always whats we get. Since the result depends on the initialization of the weights, that is performed randomly, the results differ a lot through the runs.



Figur 8: Cyclic tour

4.3 Clustering with SOM

The aim of that last subsection is to use SOM algorithm so as to position Member of Swedish Parliament based on their votes during 2004-2005. For each MP we know their party, gender and district. The final aim is to position them on a 10 x 10 grid according to their votes. We can observe that the mapping is effective when the criterion is the Political Party because we can really see clusters. Nevertheless with Sex and District criterion we can not distinguish any special clusters



Figur 9: Clustering of MPs

5 Final remarks

This assignment enables us to study and build on our own an RBF (on classification and regression) and an SOM networks, and perform them into different situations with different sorts of data and understand the choices of the various parameters values.