Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Shuyuan Zhang, Xinxing Guo, Wenqin Chen

September 17, 2019

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- To understand and implement the RBF network. Be able to use it in regression/classify problems.
- To learn how to use the competitive learning mechanism in a RBF network context.
- To be able to implement the SOM and use it to solve some practical tasks.

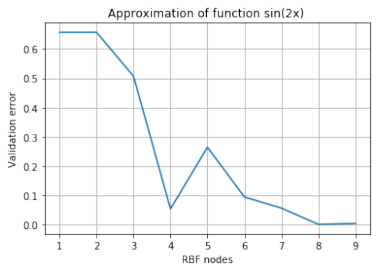
2 Methods

We worked with Python and MATLAB as our programming languages in this project, and we used numpy/matplotlib libraries in Python and the NN toolbox in MATLAB respectively.

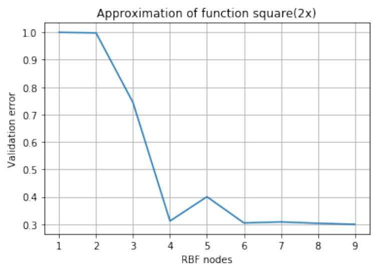
3 Results and discussion - Part I: RBF networks and Competitive Learning

3.1 Function approximation with RBF networks

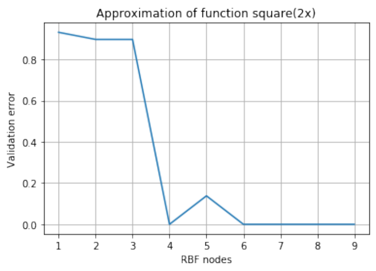
Figure1 shows the relationship between the absolute residual error and the number of RBF nodes. It is easy to see that the decreasing trend of residual error aligns with the increasing of RBF nodes. In figure1a, the validation error drops below 0.1 the first time when the RBF nodes are 4 when approximating $sin(2x)$, and it is also clear that 8 nodes are enough for a residual error less than 0.01 and 0.001. Figure1b shows the error curve of approximating function $square(2x)$ before adding a threshold to reduce the error. When applying a threshold to figure1b, we then get figure1c, which shows the validation error converges nearly to 0 when the number of nodes equals to 4.



(a) Errors and nodes in sin(2x)



(b) Errors and nodes in square(2x)

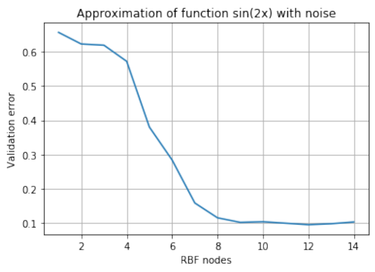


(c) Errors and nodes in square(2x) with threshold

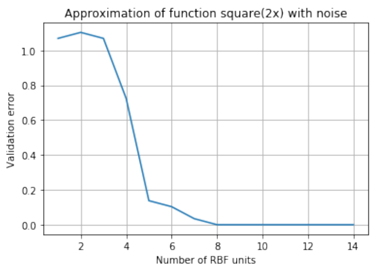
Figur 1: The absolute residual error and nodes when approximating functions without noise

3.2 Regression with noise

When introducing noise in this case i.e, Delta rule in batch, as the figure 2 shows, we obtain that 8 RBF nodes are enough to converge the absolute residual error to close to 0.



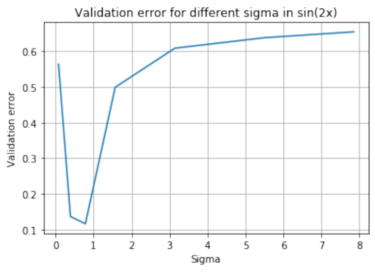
(a) Errors and nodes in sin(2x)



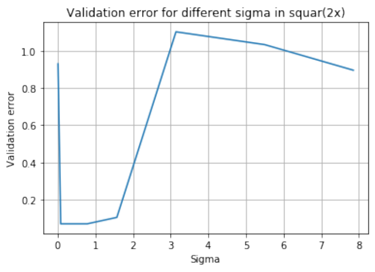
(b) Errors and nodes in square(2x)

Figur 2: The absolute residual error for different nodes with batch Delta rule

Then we fixed the number of RBF nodes to 8 and investigated the influence of the width of RBF nodes, and we obtained in figure 7 that when the value of sigma is close to 1, which is approximately $2\pi/number\_of\_RBF\_nodes$ the validation error reaches its lowest value.



(a) Errors and sigma in sin(2x)

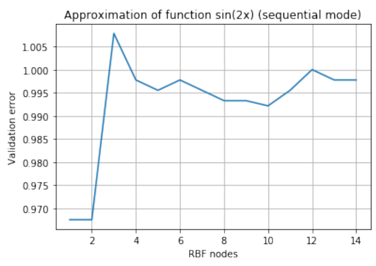


(b) Errors and sigma in square(2x)

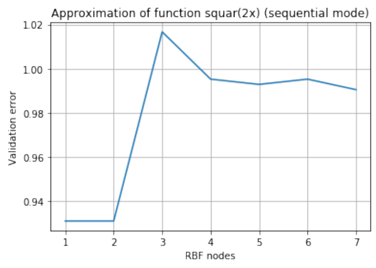
Figur 3: The absolute residual error for different sigma

Figure 4 shows the validation error trend for sequential delta rule with different RBF nodes. It shows that the Delta rule is worse than the Least Mean Squares(LMS) rule when compared to figure 1.

Moreover, we investigated the influence of *eta*, i.e, the learning rate, on the rate of convergence.

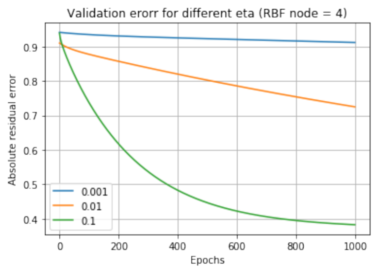


(a) Errors and nodes in sin(2x)

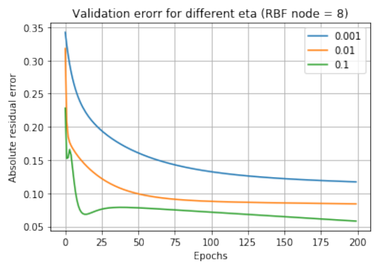


(b) Errors and nodes in square(2x)

Figur 4: The absolute residual error for different nodes with sequential Delta rule



(a) Validation error for 4 RBF nodes

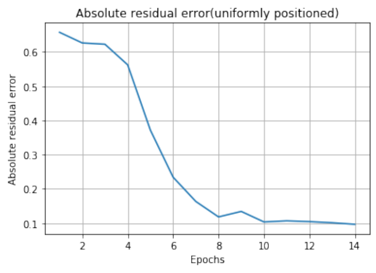


(b) Validation error for 8 RBF nodes

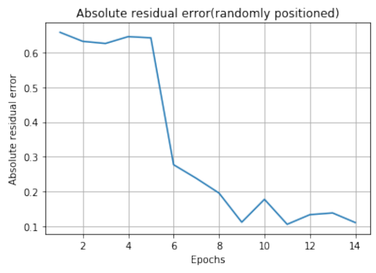
Figur 5: The absolute residual error for different value of eta

Figure 5 shows clearly that the absolute residual error converges faster with a higher learning rate with the same RBF nodes, i.e, the error curve with the eta 0.1 decreases faster than that with eta 0.01 and 0.001 respectively. In addition, with more RBF nodes in network, it is much more probable to converge to zero.

In figure 6, we investigated the position of RBF nodes on the residual errors. Those RBF nodes are either uniformly or randomly distributed between 0 and 2π, and the result corresponds to figure 6a and figure 6b respectively. We found that the performance of randomly position or uniformly position does not improve the validation error significantly.



(a) Centers uniformly position

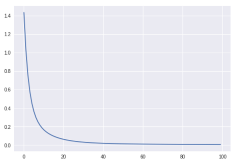


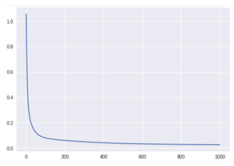
(b) Centers randomly position

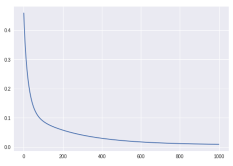
Figur 6: The absolute residual error with different positions of RBF nodes

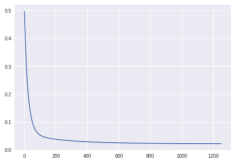
3.3 Competitive learning for RBF unit initialisation

Adding the CL algorithm to the RBF learning gives better performance for both noisy and clean data.Testing on clean data improves the results and gives better generalisation performance.

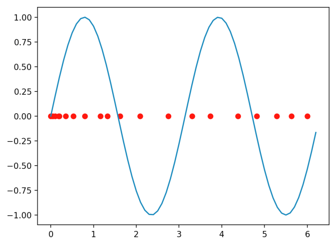

(a) CL-based approach no noise

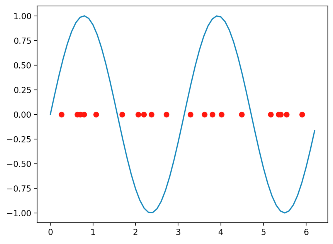

(b) CL-based approach with noise


(c) Manually positioning nodes no noise


(d) Manually positioning nodes with noise

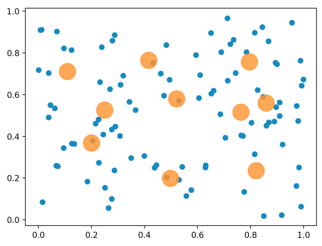
Figur 7: CL-based approach vs manually positioning nodes

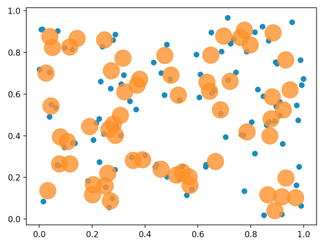

(a) Update only the closest node


(b) Update 5 nodes

Figur 8: Dead unit problem

As we can see from figure 10, since we always update the closest RBF unit (the winning unit), the other units may not be moved throughout training process (Units were initialized between 0 and 1). In order to avoid dead units, we try to update the nearby 5 units, then the units become sparse after training.


(a) 10 nodes


(b) 50 nodes

Figur 9: Ballist dataset

Now we apply the strategy to our RBF network. According to figure 11, most of units are close to the data points, also nodes seldom goes to area with less data.

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

4.1 Topological ordering of animal species

The original data is defined in a 84-dimensional space. Our task is to train a SOM in one-dimensional space to organize these animal species.

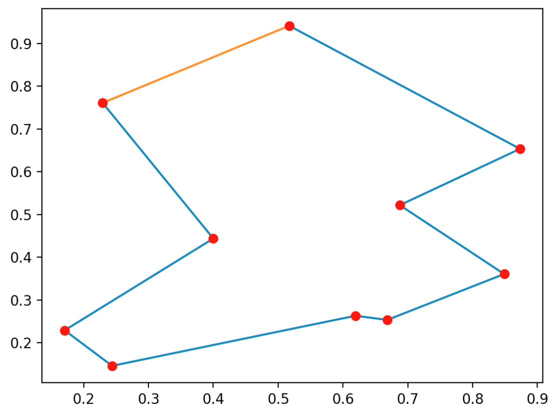
The topology of our SOM network is rather simple. The nodes are arranged in a linear way, every node is connected to two other nodes(except for head and tail). So the neighbourhood is also in a one dimensional world.
When updating the neighbours, the size of the neighbourhood decreases as iteration goes. The neighbourhood was large at first to speed up convergence and became small later to prevent allocating multiple nodes to a single animal.
The neighbourhood size was [30, 28, 27, 25, 24, 22, 21, 19, 17, 16, 14, 13, 11, 9, 8, 6, 5, 3, 2, 0].

And the final result of our animal species map is: *[spider, housefly, moskito, butterfly, grasshopper, beetle, dragonfly, pelican, duck, penguin, ostrich, frog, seaturtle, crocodile, walrus, bear, hyena, dog, lion, cat, bat, skunk, ape, rabbit, rat, elephant, kangaroo, antelop, pig, horse, giraffe, camel]*

As we can see from the list, similar animals tend to be placed together. Overall, the order is insect->bird->amphibian->mammal.

4.2 Cyclic tour

The original problem lies in a 2 dimensional space. This time, our topology map is still one dimensional, but cyclic. That is, the head and tail also links and they count as neighbours. The neighbourhood is one dimensional.
We have to reduce the maximum size of neighbourhood drastically because the problem is sparse, there are only 10 nodes and 10 cities. A large neighbourhood may result in multiple nodes be put on a same city.
After 200 iterations, our algorithm successfully converges and put a node on each city. See figure.

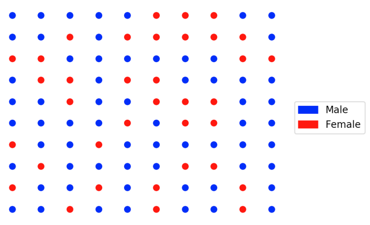


Figur 10: Cyclic tour map

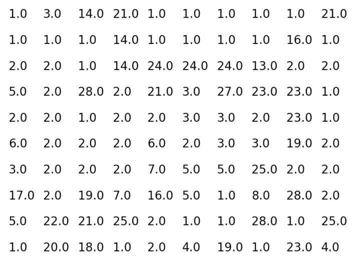
4.3 Clustering with SOM

SOM can also be used to deal with clustering problems. In this part, we are going to analyze voting patterns of different Parliament members.
The topology of our SOM in this part is a 2-dimensional grid. There are 100 nodes and they are arranged in a 10*10 grid.
This 2D topology SOM was put in a 31 dimensional space. After sufficient iterations, the map will be completely unfolded in the 31 dimensional space. If a group of Parliament members have similar voting pattern, vectors which represent their voting scheme will be close in the original 31D space, thus, they will also be close in the 2D SOM.

Results are shown in figure below.

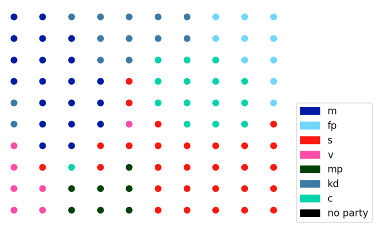


(a) Sex



1.0	3.0	14.0	21.0	1.0	1.0	1.0	1.0	1.0	21.0
1.0	1.0	1.0	14.0	1.0	1.0	1.0	1.0	16.0	1.0
2.0	2.0	1.0	14.0	24.0	24.0	24.0	13.0	2.0	2.0
5.0	2.0	28.0	2.0	21.0	3.0	27.0	23.0	23.0	1.0
2.0	2.0	1.0	2.0	2.0	3.0	3.0	2.0	23.0	1.0
6.0	2.0	2.0	2.0	6.0	2.0	3.0	3.0	19.0	2.0
3.0	2.0	2.0	2.0	7.0	5.0	5.0	25.0	2.0	2.0
17.0	2.0	19.0	7.0	16.0	5.0	1.0	8.0	28.0	2.0
5.0	22.0	21.0	25.0	2.0	1.0	1.0	28.0	1.0	25.0
1.0	20.0	18.0	1.0	2.0	4.0	19.0	1.0	23.0	4.0

(b) District



(c) Party

Figur 11: Clustering results of different attributes

Different colors are used when showing members with different sex or party, and members from different districts are marked with numbers from 1-29.

From the results we can notice that in sex and district maps, members from the same district or with the same gender do not necessarily have the same voting pattern. Male/Female members are distributed evenly in the sex map and members from the same district can be far away from each other in the district map. But members from the same party are clustered together in the party map. So we may draw the conclusion that party membership have a very strong impact on the voting patterns of Parliament members.

5 Final remarks

In this project, we looked deeply into the structure and training process of an RBF network, either for classification problems or for regression purpose. In addition, we knew the concept of vector quantisation and managed applying it in NN context. Moreover, we conducted experiment with RBF networks, which incorporate both unsupervised and supervised learning to address classification and regression tasks. Then we tried to use SOMs to recognise the role of different parameters and analyse their effect on the self-organisation in SOMs according to three tasks:Order objects (animals) in a sequential order according to their attributes; Organize circular tour which passes ten prescribed points in the plane; Make a two-dimensional map over voting behaviour of members of the swedish parliament. For example, as for the neighbourhood size, it has a great impact on the result. A main discovery was that the neighbourhood size must decrease very slowly for the last epochs to make the detailed positioning correct. If the size was not monitored carefully towards the end, task 4.1 for example, responded by positioning an animal among others with no similarity.