Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Etienne Jodry, Frano Rajic, Ivan Stresec

September 20, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to analyze the behaviour of RBF networks with respects to different hyperparameters
- to analyze the behaviour of RBF networks in respect to noisy data
- to use and tune Self Organizing Maps unsupervised learning technique to find structure in data

To experiment with RBF networks, we have used two function: the sine wave $sin(2x)$ and the $square(2x)$ function and tested many different parameters and methods as well as the impact of noise in the dataset. We also used competitive learning for clustering and determining the centres of RBFs and finally tested RBF networks on a 2-dimensional ballistical experiment. In the second part of the lab we tested SOMs in three different problems: ordering of animals, cyclic tour and clustering.

2 Methods

We have used Python 3.8 with some of its built-in libraries along with Matplotlib and NumPy libraries for all of our experiments. We also often used Google Sheets for result analysis and visualization.

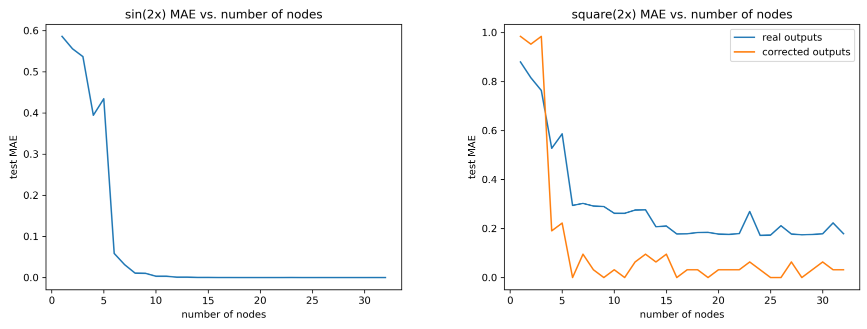
3 Results and discussion - Part I: RBF networks and Competitive Learning

3.1 Function approximation with RBF networks

The centres of RBF units are equally distanced inside of our input space $[0, 2\pi]$, the distance obviously depending on the number of RBF units. The variance of each RBF unit is equal to the square root of the distance as this showed good results in preliminary testing, unless differently specified.

Firstly, we have tested the RBF networks' ability to predict the values of the $sin(2x)$ and the $square(2x)$ functions without any noise and using the least square method (LSM), as described in the assignment 3.1.

The prediction of the $sin(2x)$ function need at least 6, 10, and 12 RBF units for an MAE lower than 0.1, 0.01, and 0.001, respectively. The performance was somewhat worse for $square(2x)$ as expected due to the shape of the function which is hard for an RBF network to describe.



(a) $sin(2x)$ (b) $square(2x)$

Figure 1: Test MAE in relation to the number of nodes

We can greatly improve the performance of the $square(2x)$ RBF network by using a sign (sgn) function. Using a sign function on the output of the network yielded an MAE of 0 with 6 RBF units. With a different distribution of RBF means we could perhaps reach an MAE of 0 with fewer nodes, e.g. placing 4 nodes in the middle of each half-period. Generally, this approach of mapping continuous values to discrete values can be used elsewhere; most obviously classification, but also regression with quantized outputs as we have seen here.

Secondly, we have added Gaussian noise with a standard deviation of 0.1 to our data and tested different RBF networks using the LSM as well as the online delta learning rule (DR). We have tested different learning rates, different RBF widths and different numbers of nodes for both the $sin(2x)$ and $square(2x)$ functions. For the delta rule we have used 25% of the training set for early stopping validation. We have compared the test MAE of different configurations as this error demonstrates the generalisation capabilities best.

For the DR we have observed faster convergence with higher learning rates, but if the learning rate is too high the error would diverge. A higher number of nodes also slows down convergence.

When changing the widths of RBFs we observed that using too small or too large widths causes a decrease in performance. This can be intuitively explained by the RBFs covering too little or too much of the input space which renders them less capable of solving the problem (figure 2).

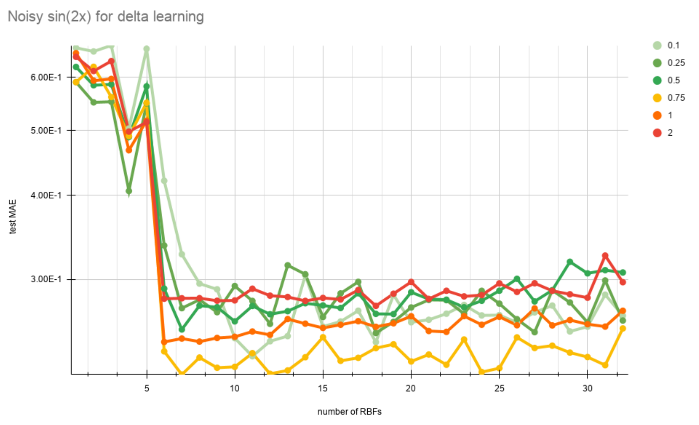


Figure 2: The influence of RBFs' width on performance of the noisy sin(2x)

Picking random RBF centres worked worse than using our strategy of distributing the centers evenly across the input space.

Generally, when using data without noise the LSM outperforms the DR, but when adding noise the DR outperforms the LSM. This is due to the fact that the LSM will find an optimal solution considering the training data, which leaves it exposed to overfitting on bad data, a problem which the DR (with early stopping) handles better.

We have compared our best performing RBF networks using the LSM with a two-layer perceptron with the same amount of nodes for the noisy $sin(2x)$ and $square(2x)$ and the RBF networks outperformed the perceptrons slightly in both problems. More general conclusions cannot be drawn.

3.2 Competitive learning for RBF unit initialisation

3.2.1 Waves

We compared MAE performance with the previously used manual initialization for RBF centres versus automated initialisation using KMeans clustering algorithm. The tests were performed on the noisy and noiseless sine(2x) dataset used before. Manual LSM performed best on noiseless dataset, but automated initialization showed much better results than using manual delta. On the noisy dataset, the automated version was not better and not much worse than the manual initialization approach. The results are summed up in figure 3. We did not monitor and compare convergence behaviour.

We noticed that dead units occurred when KMEans algorithm used random points to initialize centres. Using a random selection of train datapoints to initialize the centres made the KMeans algorithm never have dead units.

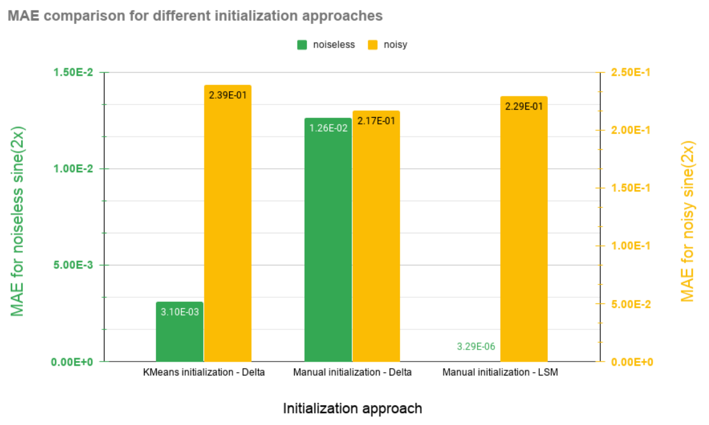


Figure 3: MAE comparison for different initialization approaches

3.2.2 Ballistical experiments dataset

We used the automatically initialized RBFs for two-dimensional regression on the ballistical experiments dataset, which is very noisy itself. The datasets maps (distance, height) pairs to (angle, velocity). KMeans CL clustering was used to initialize RBF centres. Train and test sets were separated. Both had 100 data points. For the needs of early stopping, train set was further divided into two subsets. Different KMeans initialization configurations and RBF learning rates were tested. The network managed to learn the mapping with best MAE value going down to 0.00529 for learning rate of 0.25 and 30 RBF nodes. The resulting and expected functions as well as colored clusters and the positions of RBF centres are shown in figure 4.

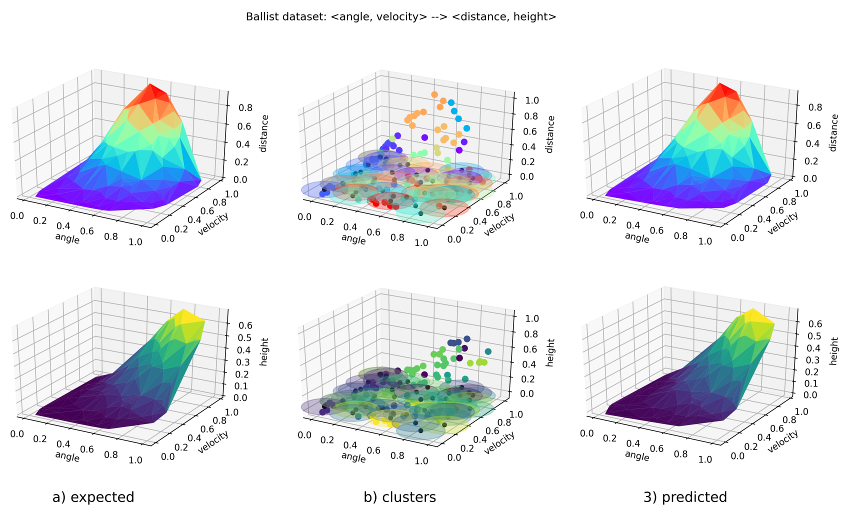


Figure 4: Results on ballistical experiments dataset. Left - expected formula. Middle - visualisation of RBF centres and corresponding clusters. Right - predicted function.

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

4.1 Topological ordering of animal species

Here is the output of the training with 100 nodes, $\eta = 0.2$ and a neighborhood sequence that follows the harmonic series multiplied by 50 in the beginning and ends with $2, 1$ and 0.

```
('bat', 1), ('elephant', 4), ('horse', 7), ('pig', 10), ('giraffe', 12),
('camel', 14), (antelope', 17), ('kangaroo', 20), ('rabbit', 22), ('rat', 25),
('cat', 27), ('lion', 29), ('ape', 32), ('skunk', 35), ('hyena', 38),
('dog', 40), ('bear', 44), ('walrus', 48), ('crocodile', 52),
(sea turtle', 55), ('frog', 59), ('penguin', 62), ('ostrich', 65),
('duck', 68), ('pelican', 71), ('spider', 77), ('housefly', 82), (mosquito', 85),
('butterfly', 88), ('beetle', 93), ('grasshopper', 95), ('dragonfly', 98),
```

Figure 5: SOM for animal species ordering: Input name vs output index

We can observe a natural organization of the data in terms of the neighborhoods:

- From 1 to 48: Mammals; From 62 to 71 Birds;

- From 52 to to 71 Chordates; From 77 to 98 Insects[1]
- Cat & Lion; Hyena & Dog; Kangaroo & Rabbit are neighbors
- ...

4.2 Cyclic tour

On the left of figure 6 we see a SOM trained with 10 nodes (one for each city), $\eta = 0.65$, 20 epochs, and the majority of the neighborhood sequence equal to zero (individual updates), so a nice specialization can happen.

We show the vectors as points in the input space and we can see they constitute a solution to traveling salesman problem if we connect them. The learning rate has to be a bit higher this time as otherwise we observe that the vectors may stay "stuck" between cities and not converge, which can be seen on the right side of figure 6: $\eta = 0.225$.

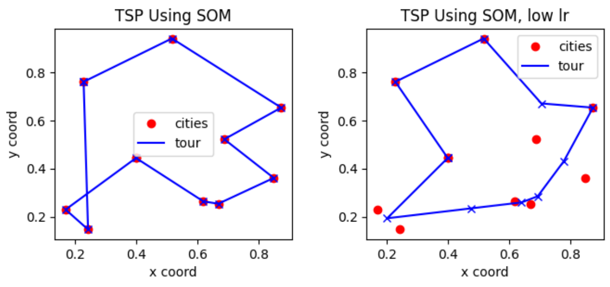


Figure 6: Cities: SOM vectors shown in the input space

4.3 Clustering with SOM

This part was a little trickier as we had to train the model separately with the result of the vote and one of the 3 metrics we hoped to do clustering on ($\in$\{sex, party, district\}), and finally build a map as shown in figure 7, associating each point to said metric.

Even when with severe varying of the neighborhood sequence and the learning rate, we got a lot of results as the one shown in figure 7: numerous mixed squares and no structure on gendered ones. Given these results, one could claim that in a country such as Sweden there is actually no gender difference in respect to votes in the parliament and do so with evidence.

On the other hand, we managed to get nice party clusters that even reflect the sizes of each party (Social Democrat party which was in majority during

[1]We include spider as an insect even though it is an arachnid, no offense intended to any biologists reading this :)

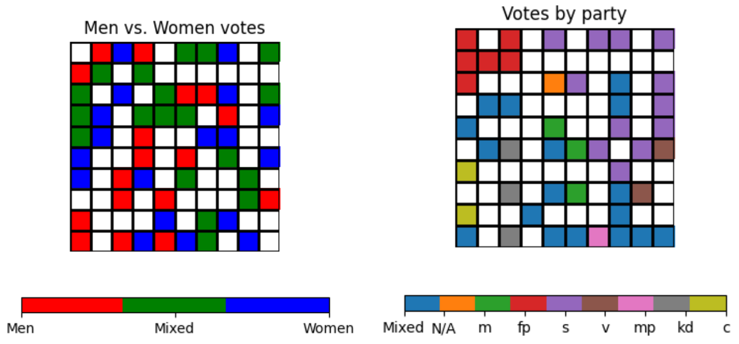


Figure 7: MP Votes: SOM 10 × 10 Colormap in respect to sex or party

2004-2005). There are also geometric intuitions that are related to the actual structure of parliament[2] that can be derived from the graphic as clusters of the social democrat (S) party and the left party (V) are close - they had an agreement during this mandate. The moderate (M) cluster is in the middle as well.

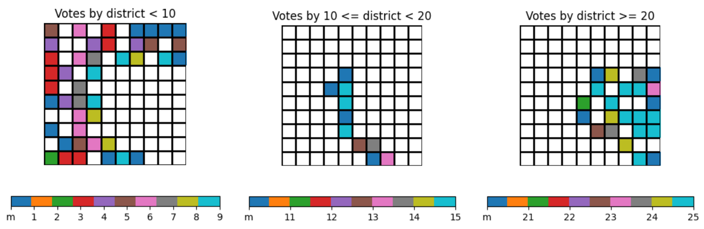


Figure 8: MP Votes: SOM in respect to Districts split in 3 plots

For figure 8 we split district plots in 3 for readability reasons. Since there is so many districts is is hard to derive conclusions but we can see general clustering tendencies, particularly when comparing districts from 0 to 9 vs. districts from 20 to 29.

5 Final remarks

The part about unsupervised learning was a bit confusing as there is no real metric to compare results. We produced a lot of graphs and kept the models that resulted in the most visually satisfying plots, but it was definitely not convenient for hyper-parameter searching. The neighborhood sequence requires a lot of tuning as well, which could be seen as tuning several hyper-parameters. For the last question of the assignment concerning the Swedish parliament, even if the clustering did work well, it was not easy to derive conclusions without knowing much about Swedish political system. This also made it interesting.

[2]Aylot, Nicholas (2015) The Party System