Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

QIU Danny, SAMAILLE Noe

September 20, 2019

1 Main objectives and scope of the assignment

The first main goal in the assignment was to **Build RBF networks** for a regression purpose, study the impact of **RBFs placement** and **initialisation** and **compare RBFs and MLP**.

The second main goal was to **Implement SOM algorithm** for ordering and clustering problems and learn different ways of **visualizing the outputs**.

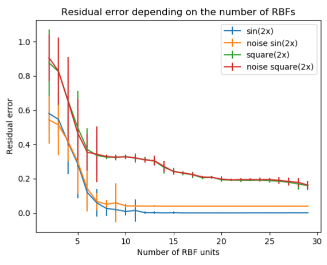
2 Methods

We used **Python** for coding with **numpy** with matrix operations.

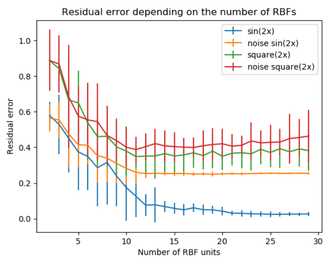
3 Results and discussion - Part I: RBF networks and Competitive Learning *(ca. 2.5-3 pages)*

3.1 Function approximation with RBF networks

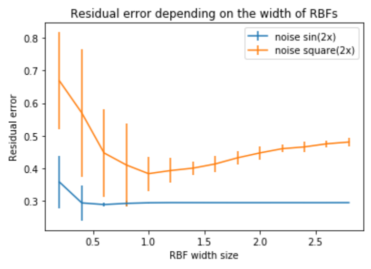
- Figure 1 shows that without noise the minimum residual error that we have got for the $square(2x)$ problem is 0.15 for 33 RBFs. For the $sin(2x)$ problem, it goes below 0.1 for $6 - 7$ RBFs, below 0.01 for $9 - 10$ RBFs and 0.001 for $13 - 14$ RBFs.
- For noisy data, number of RBFs are the same, but errors stagnates at a higher value after that.
- Sequential learning takes more computational time and error is higher for sinus and square.



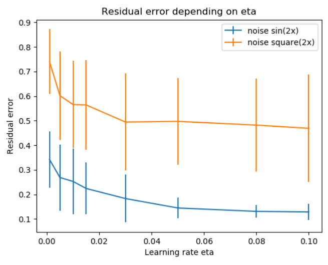
(a) Batch learning



(b) Sequential learning



(c) Batch learning



(d) Sequential learning

Figure 1: Learning depending on the number and width of RBFs and eta.

- For this data, RBFs width at 1 is optimal for batch (and also sequential) learning.
- On a training of 800 epochs, learning rate has an impact on the speed of convergence.

A simple transformation of the RBF Network to reduce the residual error to 0 for the $square(2x)$ problem is to set the output to 1 if the output is $>= 0$, -1 otherwise. The RBF network needs at least 7 RBF units to match all the input locations and direction change. This transformation is more penalizing for the error but the approximation is more faithful to the aspect of the $square(2x)$. In general this transformation can be useful for functions with discrete values.

The positioning of the RBFs and their width σ are also key parameters regarding our network performance. As the inputs are uniformly distributed and we want our RBFs to cover them all, the best thing we can do is to uniformly distribute them with a width high enough for the RBFs to cover all the input space but not too high so they do not overlap.

Strategy	**Average residual error**
$\pi/4$ spaced RBFs	2.2e$-$4
Random datapoints	1.5e$-$2

Table 1: Average residual error depending on the RBF positionning strategy with 9 RBFs, for the $sin(2x)$ problem

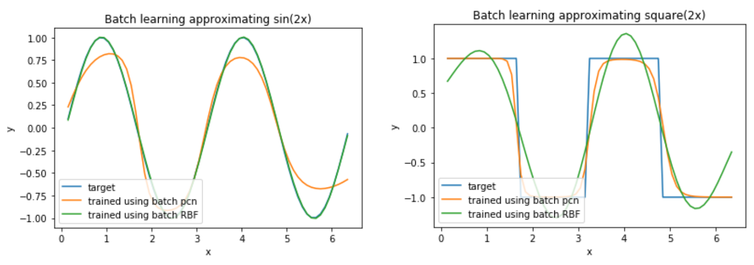
Table. 1 Shows that forcing the RBFs to be equally spaced from each other seems to be the best strategy with uniformly ditributed inputs.

Tests on clean data Results are averaged over 20 trainings using batch learning, with 7 RBFs for square (without transformation) and 13 for sinus.

Network trained...	sin$(2x)$ error	$square(2x)$ error
On clean data	1.6e−3 (sd 4.4e−3)	3.2e−1 (sd 1.5e−2)
On noisy data	6.1e−1 (sd 5.6e−3)	7.0e−1 (sd 6.3e−2)

Table 2: Average residual error on clean and noisy data

Table.2 shows higher error when trained on noise. We may assume RBF networks do not generalize well from noisy training to clean data.



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

Figure 2: Testing Two-layer PCN batch backpropagation vs RBF batch on noisy data

- RBF batch mode performs better for sin$(2x)$ approximation.
- Without any transformation, PCN has better approximation of $square(2x)$. With transformation, the approximation are the same.

On an average of 20 runs, training time for $square(2x)$ is 0.00031 seconds (sd 4.2e-05) for RBF and 0.046 seconds (sd 0.0013) for pcn. Conclusion about time performance cannot be made because the data set is relatively small and fast to train.

3.2 Competitive learning for RBF unit initialisation *(ca. 1 page)*

3.2.1 Competitive learning vs previous approaches for function approximation

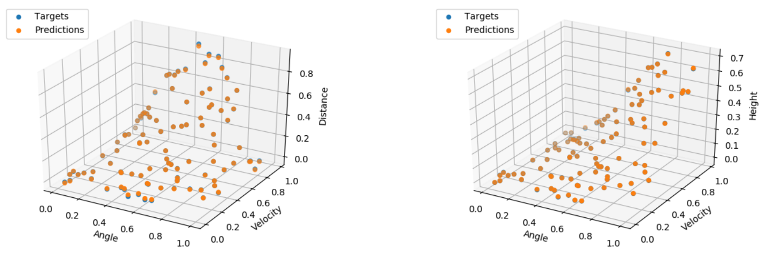
Table. 3 Shows that competitive learning does not seem to make a big difference for the $sin(2x)$ problem. We assume it is perfectly normal since our data is

Strategy	Error (Clean data)	Error (Noisy data)
With CL	1.1e−4 (std: 3.4e−4)	2.1e−1 (std: 2.0e−4)
With CL, 3 winners	4.2e−4 (std: 1.9e−3)	2.1e−1 (std: 2.1e−1)
Without CL	3.4e−4 (std: 1.6e−3)	2.1e−1 (std: 4.1e−4)

Table 3: Average residual error with/without competitive learning with 16 RBFs, for the $sin(2x)$ problem

uniformly distributed and our previous strategy positioned our RBF units uniformly, it is the best the network can do, so the competitive learning approach is not expected to do better.

3.2.2 Two-dimensional regression with RBF networks



(a) Distance (b) Height

Figure 3: Distance (left) and height (right) depending on Angle and velocity.

Figure.3 compares our RBF Network predictions and the targets for the ballistic test dataset. The data points are almost always overlapping, showing the good generalisation capabilities of our network.

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

4.1 Topological ordering of animal species

Training was made over 20 epochs. The learning rate η of the winner, of the neighbourhood η_n and the size of this neighbourhood are decreasing after each epoch. For exemple, given an epoch $i \in \{1, ..., n\}$, $\alpha = 0.4$, learning rate is updated as follows: $\eta_{i+1} = \alpha \eta_i^{\frac{i}{n}}$

Result of an unsupervised ordering using SOM

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

The order varies from one training to another and we can give different interpretations. We see birds and insects grouped together. Seaturtle, frog and crocodile are often together and they are of water and oviparous animals. Lion, cat, ... and bear are four legged mammals, with ungulate together. Rodents are next to each other.

4.2 Cyclic tour

The idea here was to find a cyclic tour minimizing the total travel distance. The only difference of the training phase with the previous problem (animals) is the neighbourhoods handling. Here neighbourhood is circular and it's size is decreased at each epoch i using $new_size = round(2\frac{n-i}{n})$. In most cases, it

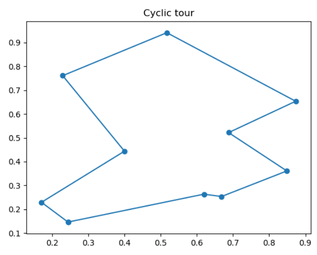
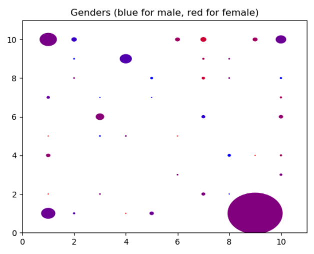


Figure 4: SOM cycling tour

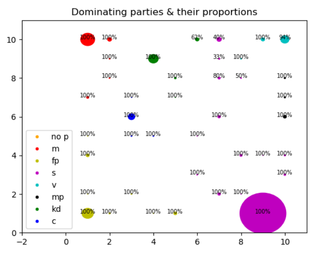
finds a good cyclic tour like in Figure.4, but sometimes we see a local detour around two cities. Our supposition is the training was stopped early and/or neighbourhood is to small.

4.3 Clustering with SOM

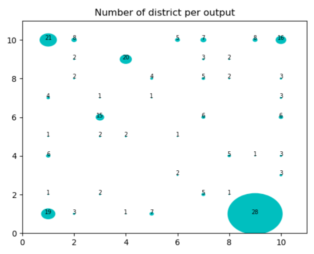
- A cluster is red if there are mostly women in it, blue otherwise, purple if even. Gender does not seem to affect the way people are voting.
- Parties affect how people vote: center-left and left-wing parties are on the right, center-right and left-right parties on the left. In each cluster, most people are of the same parties.
- The number of district in each cluster grows with the cluster size (the larger one has 28/29 districts), thus districts does not affect one's vote.



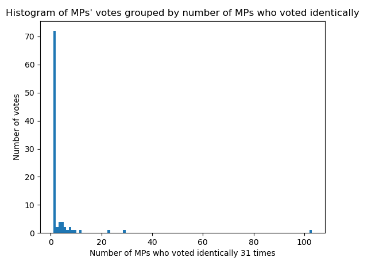
(a) Genders



(b) Parties



(c) Districts



(d) Vote distribution

Figure 5: Topological mapping displayed with different attributes

- There are 103 identical and 72 unique votes (over the 31), it explains why the grid shows a large cluster and is desert on some other places.

5 Final remarks *(max 0.5 page)*

This lab helped us understand how RBF networks and SOM networks can be used for regression, ordering and clustering. Our results showed us how RBF batch learning was more efficient in time and accuracy compared to sequential. However, regression performance depends on the type of function; sometimes MLP is better. Furthermore, MLP is more tolerant to noisy data. Regarding SOM networks, it was interesting to have a practictal approach to understand how problems can be transposed to other dimensions and interpret the outputs.