Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

HUANG yonghong

feb 10, 2025

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- know how to build the structure and perform training of an RBF network for either classi cation or regression purposes
- to recognise and implement di erent components in the SOM algorithm
- to know how SOM-networks can be used to fold high-dimensional spaces and cluster data
- to be able to comparatively analyse di erent methods for initialising the structure and learning the weights in an RBF network

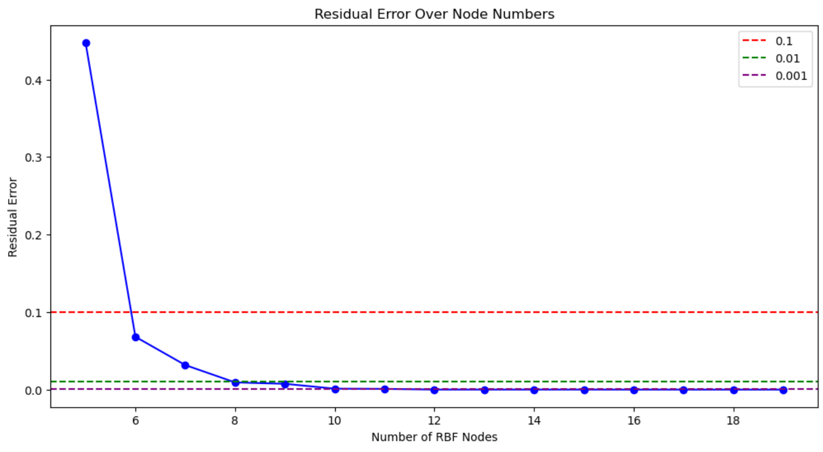
2 Methods

In this study, we implemented the Self-Organizing Map algorithm using Python with NumPy for numerical computations and Matplotlib for visualization. The data processing and analysis were conducted in Jupyter Notebook.

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

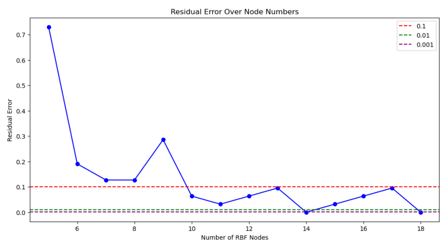
3.1 Function approximation with RBF networks *(ca. 1.5-2 pages)*

In this lab, I set $\sigma = 1$ as the basis function width and vary the number of nodes from 5 to 20 to see the result. The results show that at least 6 nodes are required to achieve an error below 0.1, 8 nodes for an error below 0.01, and 11 nodes for an error below 0.001. The trend suggests that increasing the number of nodes reduces the residual error.

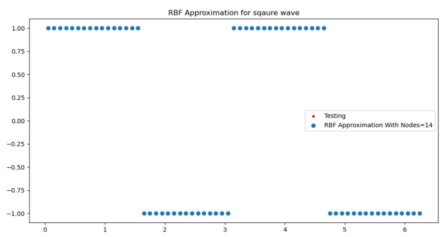


Figur 1: Residual Error Over Node Numbers For Sin

However, for the square(2x) problem, it is hard to find a solution with 0 error, but we can add a threshold fuction (if output¿=0, we set it to 1 otherwise -1) to the output node, thus we can get a 0 error solution. And we can get the best solution with 14 nodes.



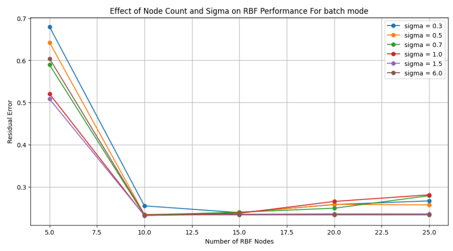
Figur 2: Residual Error Over Node Numbers For Square.



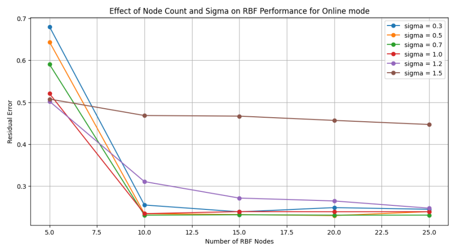
Figur 3: Output with threshold.

3.2 Regression with noise

This time, we add zero-mean noise with variance of 0.1 to the data, and vary the sigma to see the result for both online and batch mode. We can see that for the batch mode the error decreases when we increase the sigma but for online mode, with the augmentation of sigma, we get the best solution when the sigma is about 1.0 and the generalisation get worse and worse.During the lab, for this part, MSE and MAE dose not show much difference, but as the MSE is more prone to outlier and we add noise here, so I use MAE onwards.

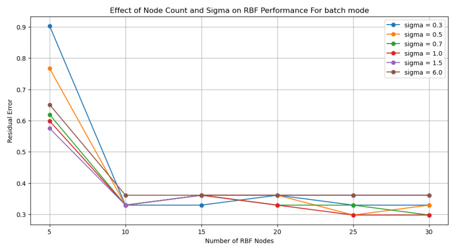


Figur 4: Batch mode for Sin.

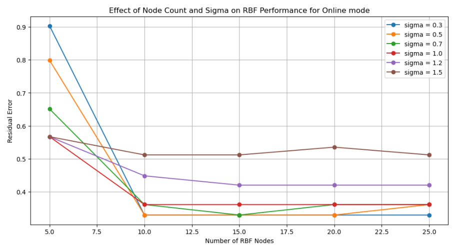


Figur 5: Online mode For sin.

The only difference between the two function is that for the sqaure(2x), we can see that the solution also get worse after the best solutionåhen we continue increase the sigma.

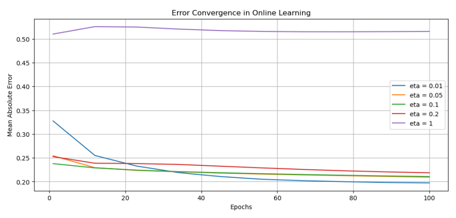


Figur 6: Batch mode For square.

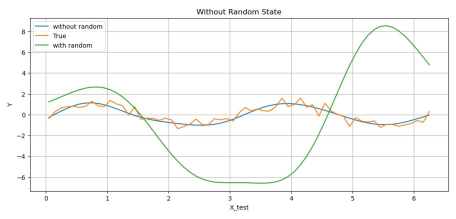


Figur 7: Online mode for square.

Basically, higher learning rate means converging faster while too higher learning rate causing fluactuating and high error.Here the 0.01 seems to be the best one. We can also see that if we choose the node positions randomly, the result is not good as we choose the innitial node position evenly.

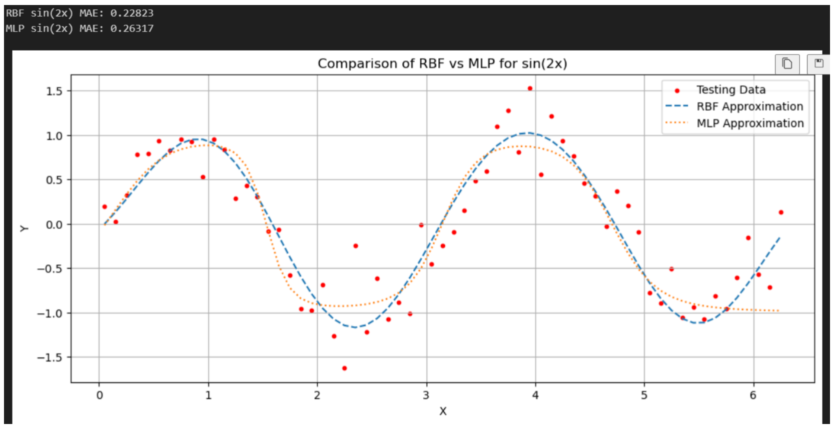


Figur 8: learning rate.



Figur 9: result.

When considering MAE, we can see that the RBF provides a slightly better solution than the MLP. However, the difference is minimal,so both models have sufficient capacity to approximate the function.

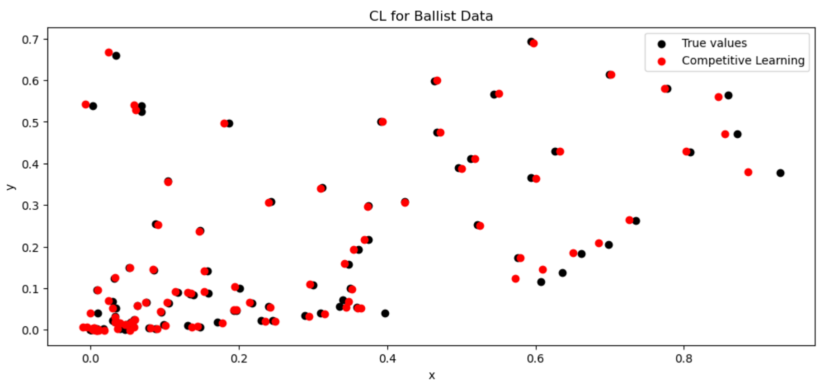


```
RBF sin(2x) MAE: 0.22823
MLP sin(2x) MAE: 0.26317
```

Figur 10: MLP VS RBF

3.3 Competitive learning (CL) to initialise RBF units

In the lab, the CLbeta(also update weight for neighbor) get a mean MAE of 0.0035621925532978212 and for simple CL the value is 0.0065396464356223885, so the clbeta does perform better than simple one.

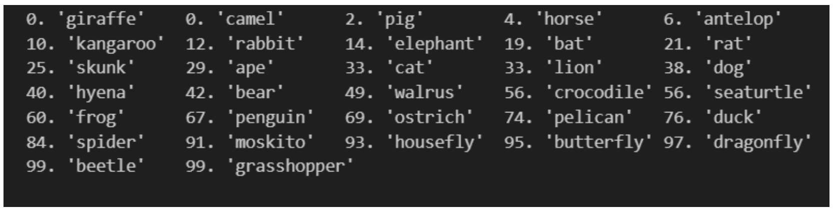


Figur 11: Ballist with MAE: 0.006236883601093216

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

4.1 Topological ordering of animal species

As a result, we can see that The clustering appears reasonable, with insects (e.g., mosquito, spider, beetle, butterfly, dragonfly) grouped together, amphibians and reptiles (e.g., frog, seaturtle, crocodile) forming another cluster, and large mammals (e.g., elephant, giraffe, bear) positioned closely. Remember here the number is the idx in the output layer, so the animals with closer number tend to be closer.

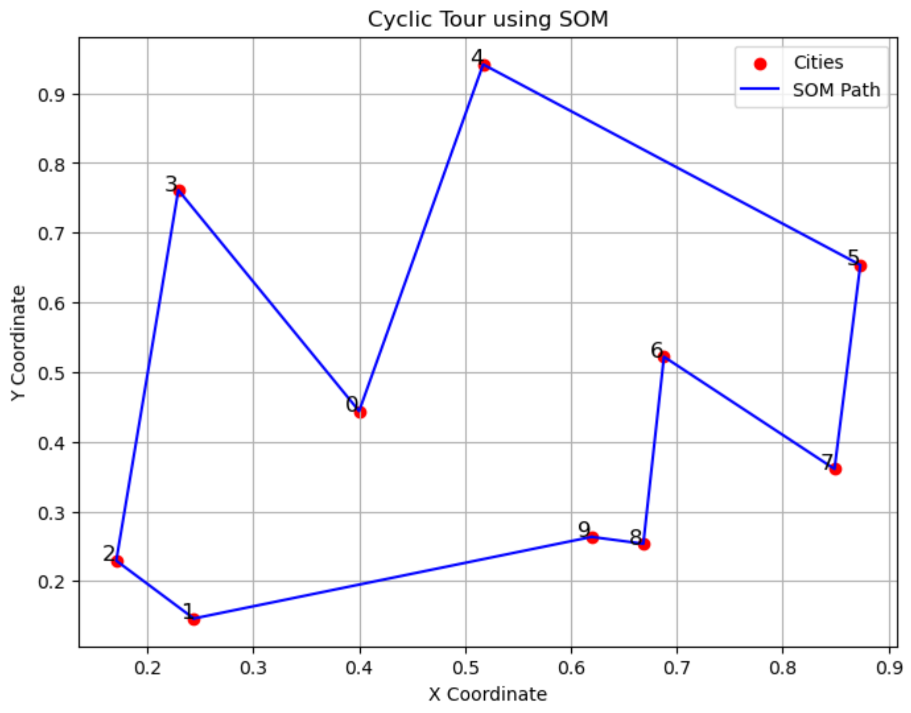


```
0. 'giraffe'    0. 'camel'      2. 'pig'        4. 'horse'      6. 'antelop'
10. 'kangaroo'  12. 'rabbit'    14. 'elephant'  19. 'bat'       21. 'rat'
25. 'skunk'     29. 'ape'       33. 'cat'       33. 'lion'      38. 'dog'
40. 'hyena'     42. 'bear'      49. 'walrus'    56. 'crocodile' 56. 'seaturtle'
60. 'frog'      67. 'penguin'   69. 'ostrich'   74. 'pelican'   76. 'duck'
84. 'spider'    91. 'moskito'   93. 'housefly'  95. 'butterfly' 97. 'dragonfly'
99. 'beetle'    99. 'grasshopper'
```

Figur 12: animal name and their output idx

4.2 Cyclic tour

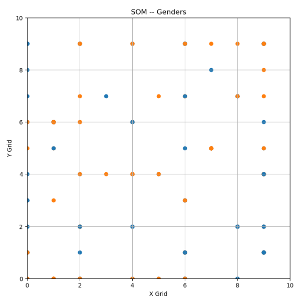
We can see the SOM successfully mapped the cities into a cyclic tour. The red dots represent the original city locations, while the blue lines show the SOM's learned path connecting them.



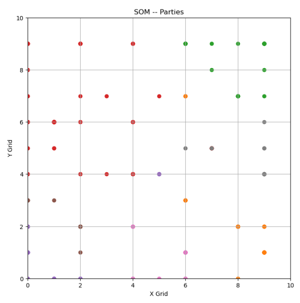
Figur 13: cyclic tour

4.3 Clustering with SOM

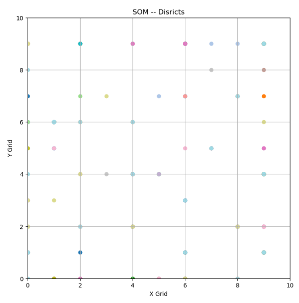
The results show that MPs from the same party tend to be mapped closely together in the output grid, suggesting that voting behavior is highly party-dependent. However, there is no significant clustering based on gender or district, which means these attributes do not effect the voting behavior much.



Figur 14: Gender.



Figur 15: party.



Figur 16: district