Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Nouhaila AGOUZAL, Abdessamad BADAOUI and Nasr Allah AGHELIAS

September 28, 2023

1 Main objectives and scope of the assignment

Our major goals in the assignment were :

- to Develop expertise in building and training RBF networks for classification and regression.
- to Learn the concept of vector quantization and its application within neural networks.
- to Gain proficiency in implementing SOM algorithms, and utilizing SOMs for dimensionality reduction and data clustering.

2 Methods

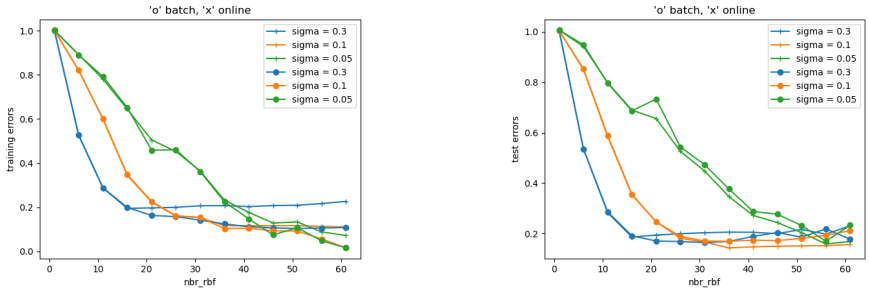
The programming was done in python. We used the numpy library for mathematical calculations and matplotlib.pyplot to generate graphs.

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

3.1 Function approximation with RBF networks

When dealing with the function sin(2x), we find that as we aim to reduce the absolute error, we require an increase in the number of units. Furthermore, this

optimal number of units tends to increase when we reduce the width of RBFs. However, for the fixed optimal number of units, the test error tends to increase as we reduce the width of RBFs. For the square(2x) function, the training error is consistently above 0.1. By setting a threshold of 0, we can reduce the residual error to zero and the optimal number of units is consistently 4 regardless of the RBF's width. This transformation is particularly useful for the classification problems.



(a) Training error

(b) Test error

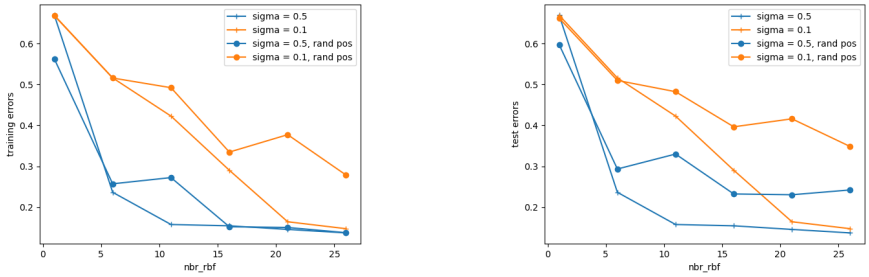
Figure 1: On-line learning VS batch learning

In the Figure 1, $\eta = 0.01$ and the number of epochs $= 10$ for the on-line learning. In this specific configuration, online learning exhibits a slight advantage over batch learning in terms of generalization capability across certain combinations of the number of units and RBF width. However, the performance comparison between on-line learning and batch learning strongly hinges on the values of η and the total number of epochs. Achieving a lower training error necessitates augmenting the number of units, especially when contending with a smaller RBF width. However, this increase in the number of units elevates the risk of overfitting the model.

The impact of the learning rate (η) on the on-line learning scheme is significant because a too large η could lead to miss the optimum, which can give rise to oscillations. Additionally, By selecting an appropriate value for η, we can attain an optimal number of epochs. However, going beyond this value may result in overfitting.

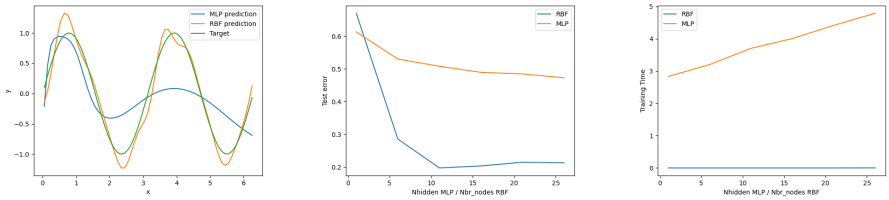
Based on Figure 2, we can notice that the balanced positioning, where the units are equally distributed in the input space, performs better than the random positioning across various combinations of units and RBF widths.

With a balanced positioning, the model exhibits superior performance on clean data compared to noisy data. This is attributed to the model's ability to successfully capture the underlying function, which is more accurately represented by the clean test data, even when trained on noisy data. However, the random positioning of the units introduce some level of randomness in the model's generalization performance. But it consistently demonstrates better performance on clean data when compared to noisy data.



(a) Training error (b) Test error

Figure 2: Random positioning VS balanced positioning of the units



(a) Predictions for test data (b) Test error (c) Training time

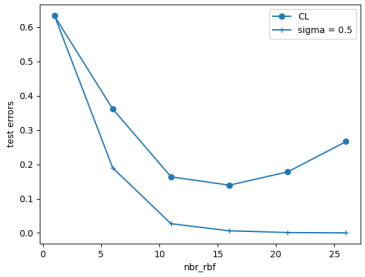
Figure 3: RBF network VS single-hidden-layer perceptron

Based on Figure 3, we can conclude that the RBF network performs better than the single-hidden-layer perceptron trained with backpropagation in terms of both generalisation performance and training time.

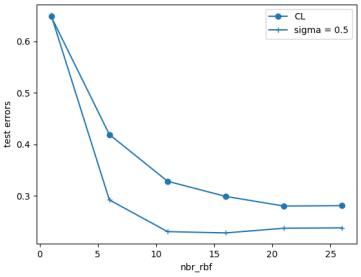
3.2 Competitive learning for RBF unit initialisation

In this assignment, we have to program a simple Competitive Learning (CL) algorithm to position the RBF units throughout our training interval instead of doing it manually like in the two previous sections. We used a straight forward winner-takes-all algorithm to determine the positions of the units and the k-nearest neighbours to estimate the width of them. For the noisy and clean data of $sin(2x)$, we observe the results in the figure 4

When training data points are uniformly spread across the interval $[0, 2\pi]$, placing RBF units manually at regular intervals aligns well with the data distribution. This positioning ensures each unit adequately covers a specific portion of the input space, enabling effective representation and approximation of the function. In contrast, utilizing a competitive learning algorithm for unit initialization may not distribute the units optimally in alignment with the evenly spaced training data. The algorithm might converge to a suboptimal configuration that does not match the inherent structure of the data distribution.



(a) Clean data of sin(2x)

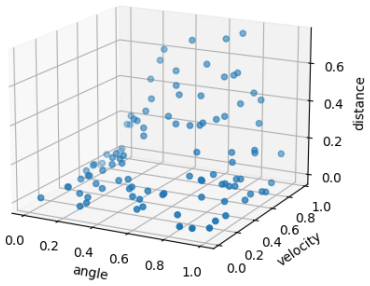


(b) Noisy data of sin(2x)

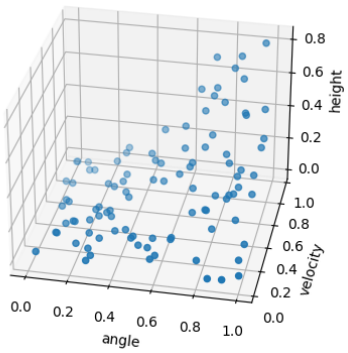
Figure 4: The test error for our RBF neural network between CL and manual positioning across different numbers of units

The CL algorithm is sensitive to initial unit randomization. We employed two heuristics to mitigate this. One initializes units based on manual positions from prior sections. The other applies a soft winning rule, where the winner shares the glory with best losing units (neighbors). Neighbourhood size decreases exponentially across iterations. We deliberately initialize units in a small interval to obtain dead units. While training performance is similar, a high number of RBF units significantly impacts hard CL test performance due to a higher chance of dead units, unlike the more robust soft CL.

In the next task, we have to approximate a 2 dimensional function : <distance, height> with respect to <angle, velocity>. The following two figures represents the distribution of height and distance with respect to angle and velocity for the original training data



(a) Distance with respect to Angle Velocity



(b) Height with respect to Angle Velocity

Figure 5: Distance and Height with respect to Angle and Velocity

RBFNs struggle to capture meaningful patterns or relationships within the input space if there is no regularity or structure. This is because RBFNs rely on learning localized patterns using radial basis functions, and the lack of regularity

makes it difficult to identify such patterns. Our CL learning algorithm might converge to distribution of units that's not optimal for our training data. To avoid the dead units issue, we initialised the RBF units by random patterns from the input training data.

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

4.1 Topological ordering of animal species

In this first task, where we construct a self-organizing map to group different animal species, we employed 20 epochs and a learning rate of 0.2. The number of neighbors decreases exponentially over time (epochs), starting with 50 neighbors and gradually reducing to 0 neighbors. In our simple approach, we apply the same update for the neighbors as we do for the best matching unit. After training, and iterating over all the animals, we sort them based on their indices in the output space, yielding the following results :

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

4.2 Cyclic tour

In this assignment, the objective is to associate 10 cyclic nodes with 10 cities denoted by coordinates in a two-dimensional plane. The cyclic nature of the nodes is important as we aim to establish a tour that visits each city, necessitating the first and last nodes to be adjacent. We conduct training on a Self-Organizing Map (SOM) for 100 epochs, utilizing a learning rate of 0.2 after initializing the weights in a random manner. The initial neighborhood size is set at 2 and diminishes to 0 over a specific number of epochs. The result of this process is the obtained tour, representing the path through the cities in the figure 6 Due to the proximity of certain cities, a single node is responsible for representing both of them, causing the node to be positioned in between them, ultimately leading to inactive units.

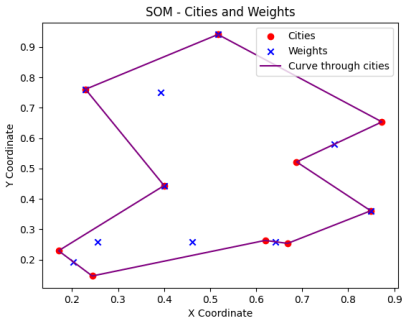
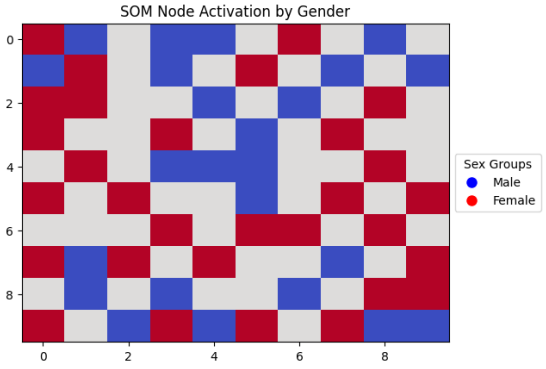


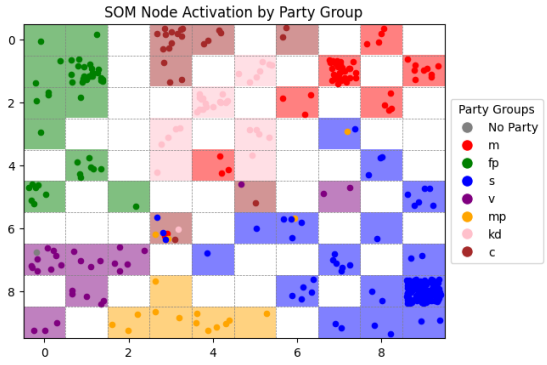
Figure 6: The tour going through the cities

4.3 Clustering with SOM : MPs' votes

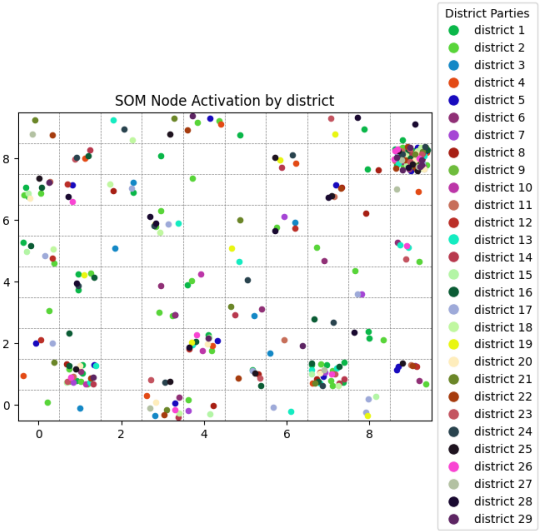
For this problem, we utilized a 10x10 2D grid as the output space. We ran a total of 100 epochs with a learning rate of 0.2. The number of neighbors also decreases exponentially in this case, and we employ the Manhattan distance in the output space to determine which nodes will be updated along with the best matching units. As we can observe in these figures 7, there is no clear pattern indicating that gender or district significantly influences MPs' votes. This suggests that their votes do not depend on gender or district. However, when it comes to party affiliation, we can clearly discern the formation of clusters, indicating that the party has a strong influence on the MPs' votes.



(a) Self Organizing map with respect to gender



(b) Self Organizing map with respect to party



(c) Self Organizing map with respect to party

Figure 7: SOMs with respect to different attributes (gender, party, district)

5 Final remarks

Coding the RBF neural networks and Self Organizing Maps from scratch helped us internalize the theoretical aspects of both these architectures by going through a practical approach.

Furthermore, the lab highlights the importance of the generalisation capability and the training time as important aspects to be wary of.

From the similarities between animals to the cyclic tour and finally the votes of the MPs, we mostly mapped high dimensional data into a small output space which was an interesting task aiming to condense the inherent similarities in the input space into a more comprehensible output space.