Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Boyu Li,Omar E Contreras Z, Yu Hu

February 12, 2020

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to build and perform training of RBF neural networks and implement competitive learning for unsupervised learning.
- to implement SOM algorithm, disscuss the role of neighbourhood and use SOM to fold high-dimensional spaces and cluster data.

2 Methods

We use Python to implement the RBF Neural Networks and solve the first problem of part 2. Matlab is used for solving the last two problems in part 2.

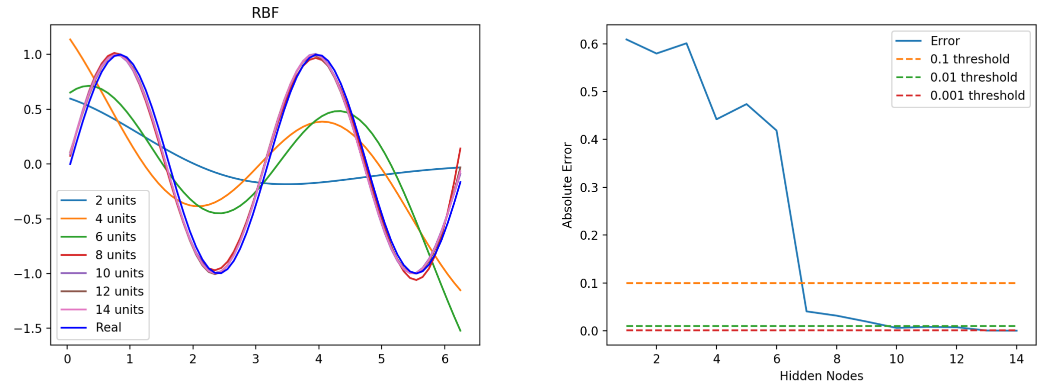
3 Results and discussion - Part I: RBF networks and Competitive Learning

In this report, we divide the results into two parts: Noise-free data and Noisy data. Both of this two kinds of data set will be used when doing bath mode learning and online learning.

3.1 Noise-free data

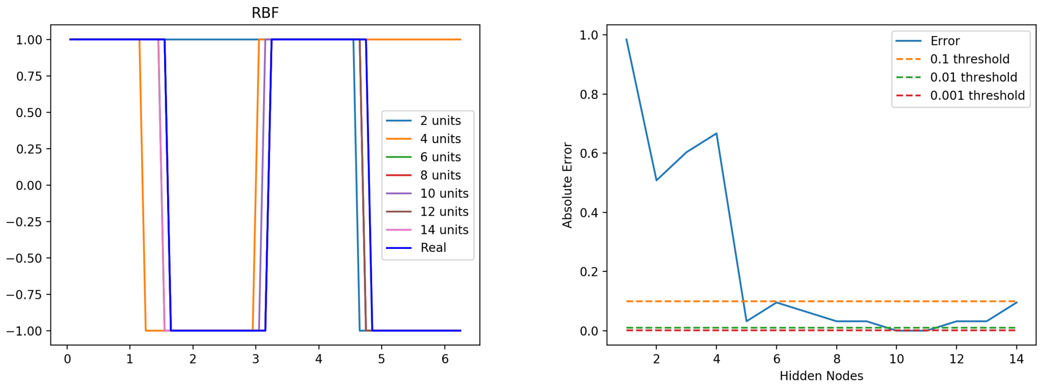
3.1.1 Batch mode training using least squares

We vary the number of units to obtain the absolute residual error below 0.1,0.01 and 0.001. For $sin(2x)$ function, we need 7, 10, 12 units to get the converge. For $square(2x)$ function, we need 5, 10, 11 units to get the converge. We transformed the output of RBF network to reduce the residual error: when output is larger than 0, we set the output to 1, when it's less than 0, we set the output to -1. This transform is particularly useful for classification.



(a) (b)

Figure 1: RBF networks and absolute error with $sin(2x)$ function

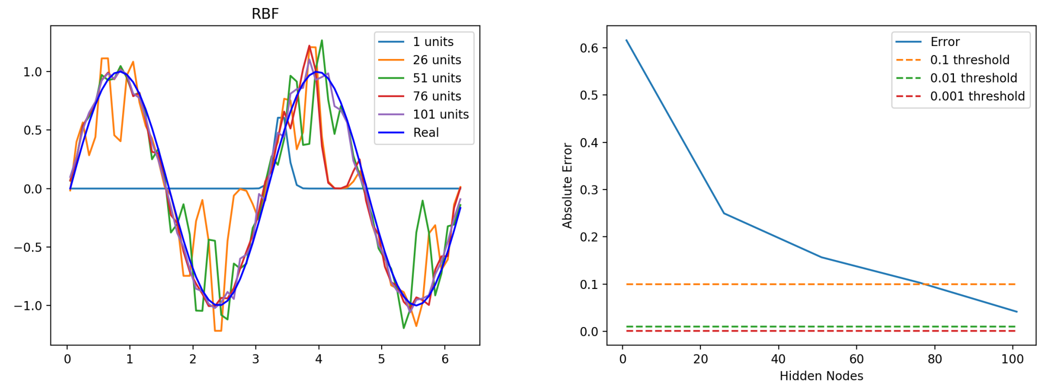


(a) (b)

Figure 2: RBF networks and absolute error with $square(2x)$ function

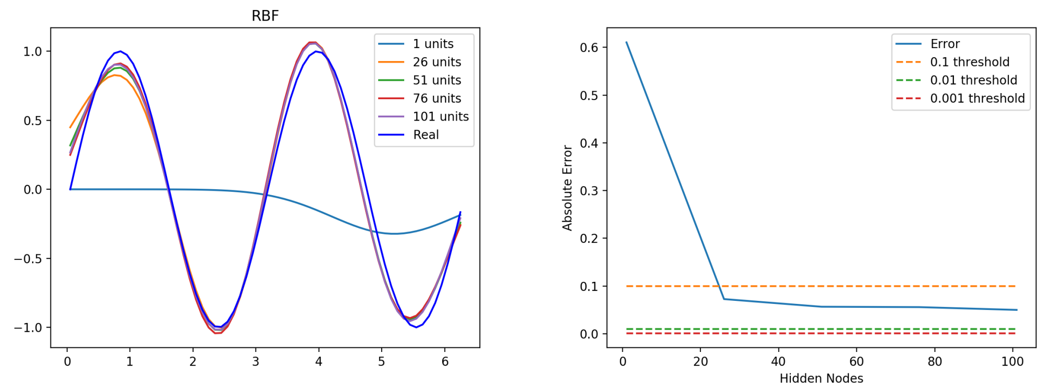
3.1.2 Online learning with delta rule

We adjusted parameter σ of RBF network, the value of σ depends on the data.If the data are widely separated, it's reasonable to choose a larger width to cover more data points as much as possible. If the data are very close, then we should choose a smaller width. σ decides the width.



(a) (b)

Figure 3: $\sigma = 0.1$



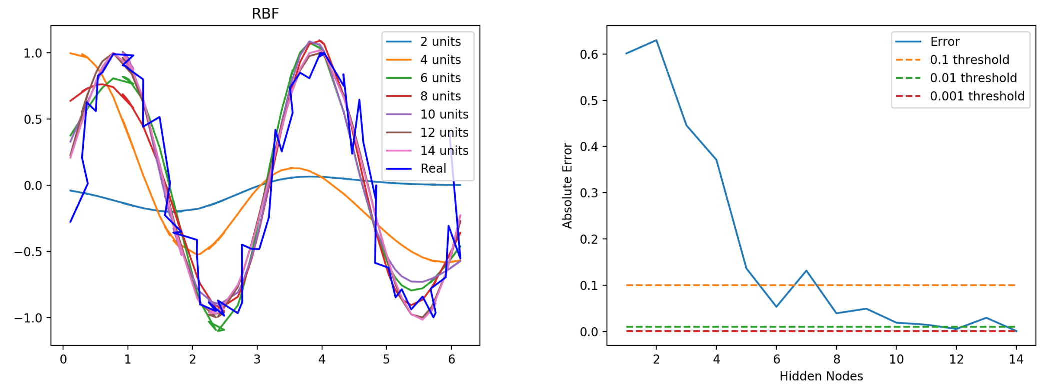
(a) (b)

Figure 4: $\sigma = 1$

3.2 noisy data

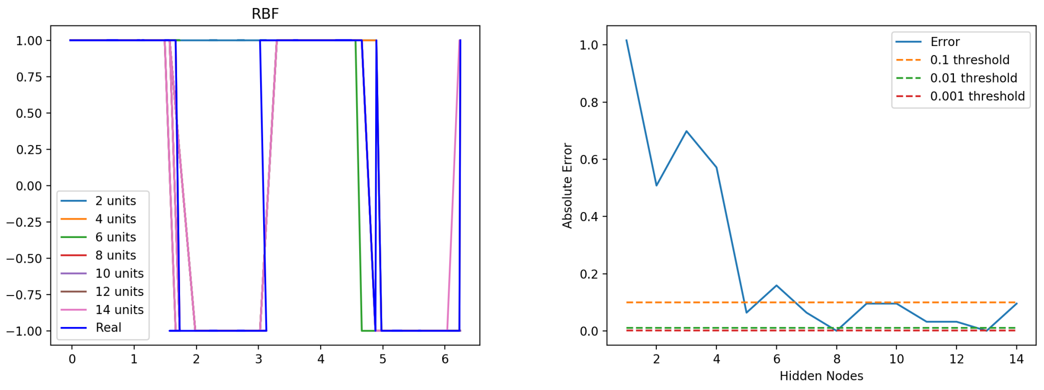
3.2.1 Batch mode training using least squares

When we added noise to data set, more units needed to reduce the absolute error to 0.



(a) (b)

Figure 5: RBF networks and absolute error with $sin(2x)$ function

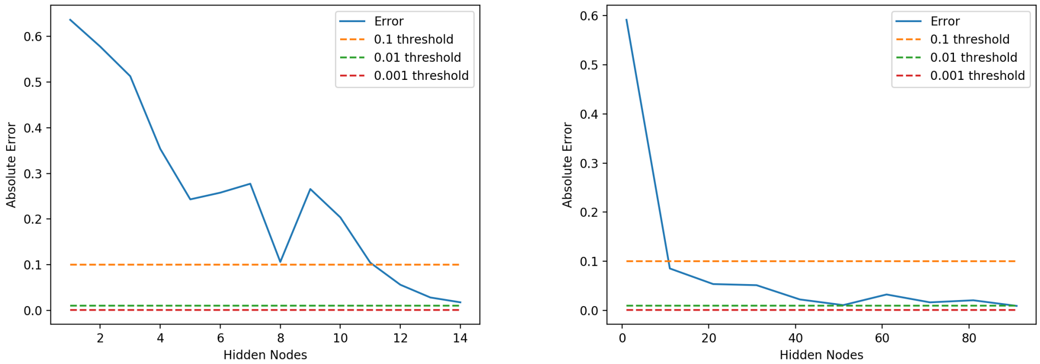


(a) (b)

Figure 6: RBF networks and absolute error with $square(2x)$ function

3.2.2 Online learning with delta rule

When we applied online learning to noisy data set, it needs more units and more epoch to reduce error.



(a) $sin(2x)$

(b) $square(2x)$

Figure 7: Error with different function

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

Competitive learning with more than one winner is a way of avoiding dead units.

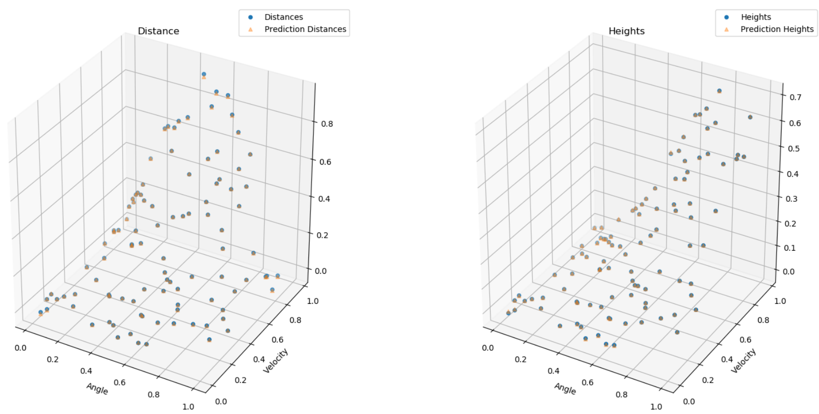


Figure 8: RBF network with the use of CL for positioning the RBF units to approximate a two-dimensional function.

Figure 8 demonstrates the capability of a network to represent a two dimensional function.

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

4.1 Topological ordering of animal species

As the figure below shows the SOM can be used to assign a natural order to objects.

```
(base) n183-p245:lab2 yuhu$ /Library/Frameworks/Python.framework/Versions/3.7/bin/python3 /Users/yuhu/Desktop/p3-ann/lab/Ar
ks-and-Deep-Architectures/lab2/som.py
["'antelop'\t\n", "'ape'\t\n", "'bat'\t\n", "'bear'\t\n", "'beetle'\t\n", "'butterfly'\t\n", "'camel'\t\n", "'cat'\t\n", "'
'\t\n", "'dragonfly'\t\n", "'duck'\t\n", "'elephant'\t\n", "'frog'\t\n", "'giraffe'\t\n", "'grasshopper'\t\n", "'horse'\n",
na'\t\n", "'kangaroo'\t\n", "'lion'\t\n", "'moskito'\t\n", "'ostrich'\t\n", "'pelican'\t\n", "'penguin'\t\n", "'pig'\t\n",
\t\n", "'seaturtle'\t\n", "'skunk'\t\n", "'spider'\t\n", "'walrus'"]
[[8.32257177e-08 1.12751609e-26 4.25672390e-02 ... 7.70663196e-26
  4.28212024e-26 1.63881165e-02]
 [1.38663614e-08 4.91225646e-28 2.56170506e-02 ... 1.42812395e-27
  2.49433843e-24 4.00280966e-02]
 [8.87447130e-09 9.47201109e-29 1.53605898e-01 ... 2.55934840e-28
  1.10516156e-24 4.00266038e-02]
 ...
 [2.22256812e-13 1.00000000e+00 4.55511389e-15 ... 4.55896368e-03
  9.95441036e-01 6.62462899e-02]
 [3.02166555e-13 1.00000000e+00 2.86244170e-13 ... 7.12124865e-03
  9.92878751e-01 8.50341556e-02]
 [5.37618880e-13 1.00000000e+00 1.63315152e-14 ... 7.12124865e-03
  9.92878751e-01 8.50341556e-02]]
[28  0 17  0 94 91 33  7 51  4 99 69 18 56 33 96 43 87  0 22  7 85 61 66
 63 39 25 13 48  4 78 10]
["'ape'\t\n" "'bear'\t\n" "'hyena'\t\n" "'dog'\t\n" "'skunk'\t\n"
 "'cat'\t\n" "'lion'\t\n" "'walrus'" "'rat'\t\n" "'bat'\t\n"
 "'elephant'\t\n" "'kangaroo'\t\n" "'rabbit'\t\n" "'antelop'\t\n"
 "'camel'\t\n" "'giraffe'\t\n" "'pig'\t\n" "'horse'\n" "'seaturtle'\t\n"
 "'crocodile'\t\n" "'frog'\t\n" "'ostrich'\t\n" "'penguin'\t\n"
 "'pelican'\t\n" "'duck'\t\n" "'spider'\t\n" "'moskito'\t\n"
 "'housefly'\n" "'butterfly'\t\n" "'beetle'\t\n" "'grasshopper'\t\n"
 "'dragonfly'\t\n"]
```

Figure 9: result

1.1 Cyclic tour

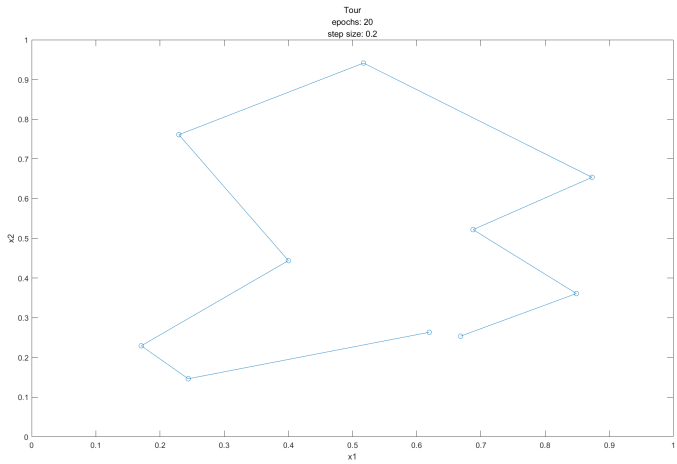
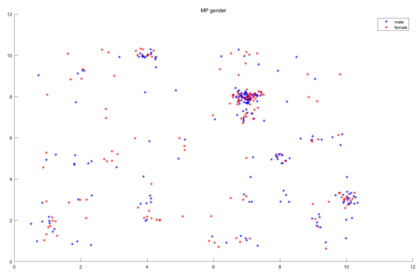


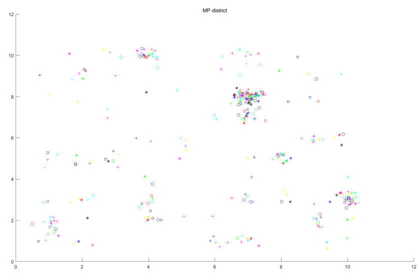
Figure X: A fairly short route found by the SOM algorithm.

The initial size of the neighborhood is 2, decayed by 0.8. The step size is 0.2. We use an outer loop to train the network for 20 epochs and used a weight matrix initialized with random numbers between zero and one. In Figure X, we can see that it is a fairly short route that passes all cities. But the performance might depend on the initialization of the weight matrix.

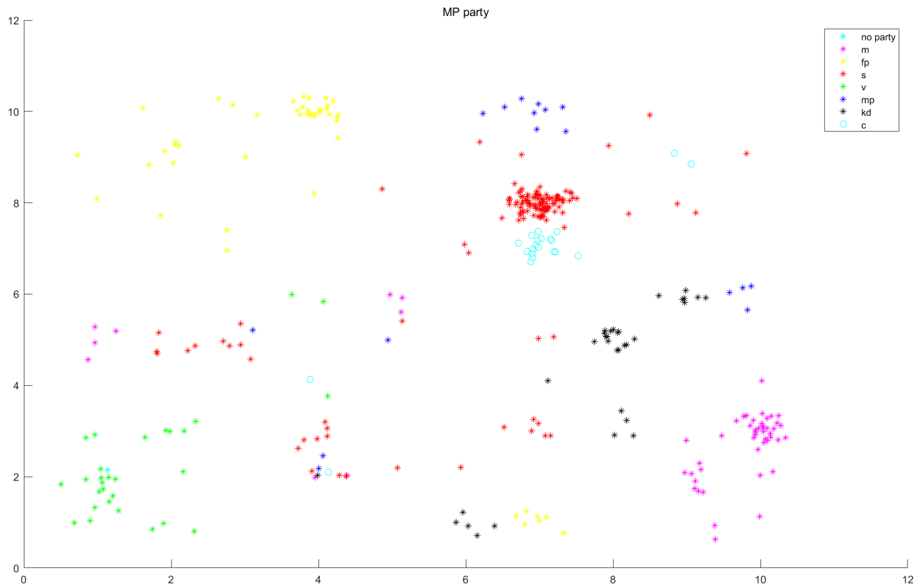
1.2 Clustering with SOM



(a)



(b)



(c)

Figure Y: Position of all MPs with respect to (a) gender, (b) district, (c) party.

The initial size of the neighborhood is 5, decayed by 0.8. The step size is 0.2. We use an outer loop to train the network for 20 epochs and used a weight matrix initialized with random numbers between zero and one. We added some noise to the output coordinates to avoid overlap. In Figure Y, we can see that gender and district might not have a significant effect on the votes of the MPs. The votes of the MPs should depend on the party since for the same party the points are close to each other.

2 Final remarks

For Part II, we found that it is important to handle the neighborhood sizes.