Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Robert Cedergren, Viktor Törnegren and Lingxi Xiong

February 7, 2019

1 Main objectives and scope of the assignment

- To implement an RBF network for the task of regression
- To initialize the RBF mean using comparative learning
- To implement SOM algorithm to find a low-dimensional representation of high-dimensional data

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

2.1 Function approximation with RBF networks

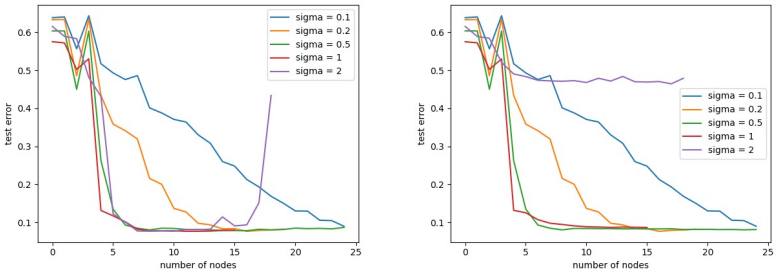
For function approximation of $sin(2x)$, the number of units to obtain the absolute residual error below 0.1, 0.01 and 0.001 is 15, 22, and 57, respectively, by using sigma=0.15 and evenly spaced initialization. And when number of units is greater than 63, the test error starts to increase. This is expected since without noise this is the exact interpolation situation, where the size of the hidden layer is equal to the number of samples (here we have 62 samples). Then there is no doubt that model is overfitting towards with the training data.

For the other function $square(2x)$, with same sigma(0.15) we could only reach the best test error below 0.1 with 57 nodes(training error is 0.071 and test error is 0.095). The network start to overfit also after the number of unit is greater than 63.

In order to reduce the residual error to 0 for the square(2x) problem, we can apply a binary activation function/indicator function/unit step function to the

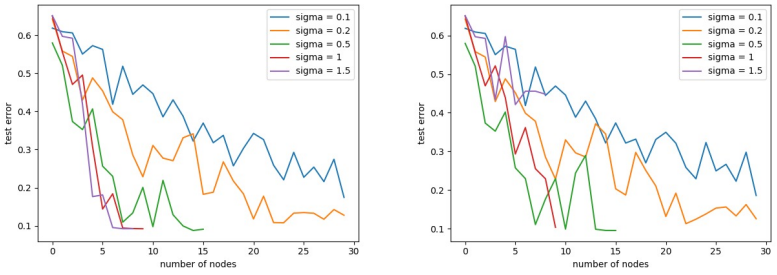
output of the RBF network to transform the result to 0(-1) and 1. And this transformation is particularly useful in classification problems.

2.1.1 Comparative analyses of batch and on-line learning schemes



(a) batch mode using least squares (b) sequential mode using delta rules

Figure 1: generalization performance on noisy data vary from different width and different number of nodes with evenly distributed initialization



(a) batch mode using least squares (b) sequential mode using delta rules

Figure 2: generalization performance on noisy data vary from different width and different number of nodes with randomly distributed initialization

As we can see in Figure 1 or Figure 2, the smaller width of RBF (sigma), i.e. to reduce the radius of the receptive fields, results in having more RBFs in the same region, which will lead to a more finely grained output, i.e., result in lower test errors. Meanwhile the larger width will need less units to get a similar result, and fail with excessive units because of overlapping in receptive fields between each node.

The positioning of the RBF nodes (in the hidden layer), i.e. initialize the positions of the RBF centers, is the first out of two parts in training RBF network, with the purpose of finding a non-linear representation of the inputs. In order to get this, the two mainly used methods are k-means algorithm, and set RBF

centers to be randomly chosen datapoints. We position the center(mean) of RBF nodes evenly in the previous task. The comparison of that with the result of randomly distributing the RBF nodes is shown in Figure 1 and Figure 2. We can see there are significant fluctuation in test error vary from number of nodes. Also, the generalization performance of randomly positioned RBF nodes is worse than the corresponding evenly positioned ones.

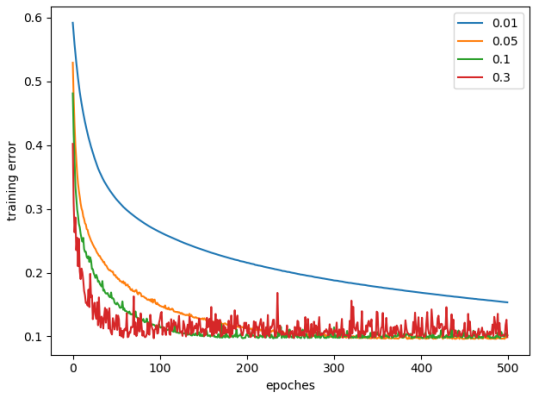


Figure 3: rate of convergence with different learning rate(#RBF=14, sigma=1)

For batch mode using least squares, all training is done within one step of matrix computation, so the learning rate is not applied here. For online learning using delta rule, the effect of the learning rate on the convergence should be similar to the one of perceptron learning, where the rate of convergence will increase with a lager learning rate within limits but will result in overfitting/jumping problem with too large learning rate. So in this experiment we set $\eta = 0.02$ and 800 epochs as a better combination.

If we train the network with noisy data and test them with clean data, less number of nodes will be needed to reach a same certain amount of test error, which also imply the potential to get a better performance.

2.1.2 Comparison between RBF- and MLP-based approaches

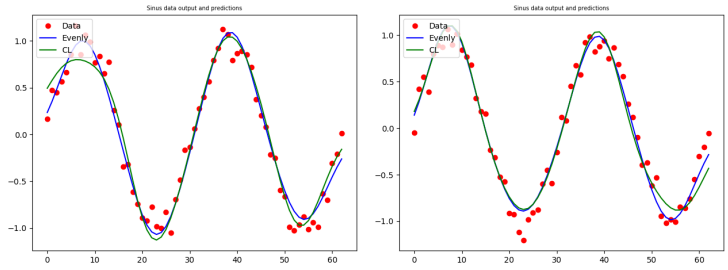
For function $sin(2x)$ with sigma = 0.5 and eta = 0.02, the RBF network with sequential learning and evenly positioning kernels performs better than the MLP network as can be seen in the table below. The RBF network does not need any epochs in batch mode, while the MLP network needs 5000 epochs in batch mdoe to get a quite low test error.

	train error	test error
RBF	0.0830	0.0901
MLP	0.0861	0.1252

Table 1: RBF compared to MLP, sigma = 0.5, eta = 0.02, data = noisy $sin(2x)$

2.2 Competitive learning for RBF unit initialisation

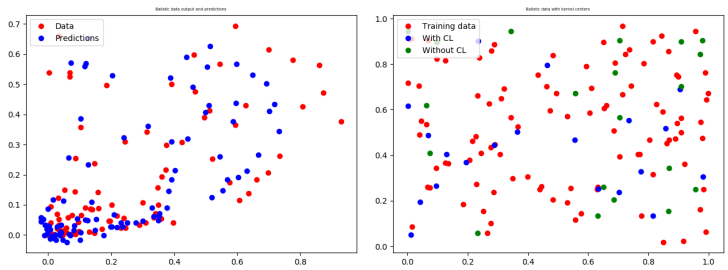
From Figure (a) and (b) below, we can clearly see that evenly spaced kernels and CL-based kernels produces similar results. Thus, we can conclude that evenly spaced kernels is a good choice. Further, if we compare the CL-method without avoiding dead units in Figure (a) with the CL-method that tries to avoid dead units in Figure (b), we can clearly see that for the noisy sinus data, they perform equally good.



(a) (b)

Figure 4: The noisy sinus data set with epochs = 800, sigma = 0.5, eta = 0.02. Figure (a) is CL without avoiding dead units and Figure (b) is CL with avoiding dead units

When applying the RBF network to the ballistical data with sigma = 0.5, eta = 0.02, epochs = 8000, nodes = 20 we get *Train error 0.021 Test error: 0.026*. As can be seen from the test error as well as the plot below (Figure (a)) the predictions are really good. Moreover, from the kernel plot we can see that it really does not matter if we are using randomly chosen center of the kernel or the competitive learning method.



(a) (b)

Figure 5: The ballistic data set with epochs = 8000, nodes = 20, sigma = 0.5, eta = 0.02

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

The SOM algorithm is used to assign natural order to objects characteristics by a large number of features. This is done by letting the SOM algorithm create a topological mapping from the high-dimensional feature space to a low-dimensional output space.

3.1 Topological ordering of animal species

The SOM network is trained by showing an observation (the feature vector) of one animal at a time and maps onto 100 output nodes such that similar animals tend to be close whereas different animals are further away along the sequence of the nodes. The training is done for 20 epochs with a constant learning rate of 0.2. For each epoch and each animal the closest weight vector is found and updated together with its neighbours. The size of the neighbourhood $h(t)$ is 50 from the start and gradually decays for each epoch t according to $h(t) = 50/50^{t/20}$. A one-dimensional neighbourhood is used and thus results in a one-dimensional topology. Note that the distance in the high-dimensional feature space is not preserved in the one-dimensional topology, hence, the distance in the one-dimensional topology does is not a distance in a normal sense. We can see in the table below that insects are grouped together, birds are grouped together, water animals are grouped together, big animals seems to be grouped together etc.

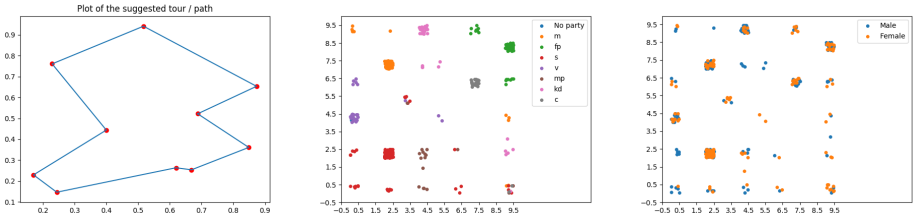
1. spider	9. pelican	17. bat	25. rabbit
2. moskito	10. ostrich	18. lion	26. elephant
3. housefly	11. penguin	19. cat	27. kangaroo
4. butterfly	12. seaturtle	20. dog	28. giraffe
5. dragonfly	13. crocodile	21. walrus	29. horse
6. beetle	14. frog	22. ape	30. camel
7. grasshopper	15. rat	23. bear	31. pig
8. duck	16. skunk	24. hyena	32. antelop

3.2 Cyclic tour

The cyclic tour can be interpreted as a variant of the travelling salesman problem. The training data are cities with two-dimensional coordinates and the SOM algorithm provides a two-dimensional output with a curve that corresponds to the tour. The output grid is 10 nodes (which corresponds to the ten cities) and the neighbourhood is circular since a cyclic tour is desired. The number of epochs used is 20, the learning rate is 0.2 and the start size of the neighbourhood is 2 which decays in the same way as for the the topological ordering of the animal species. In Figure 6 (a) the cyclic tour is shown. It seems like the shortest path is chosen.

3.3 Clustering with SOM

The SOM algorihtm is used to position 349 MPs on a 2D 10×10 output grid according to their votes during 2004-2005 based on party, gender and district. The idea is that voting patterns that are similar to each other should be close to each other. 50 epochs, start size of the neighbourhood 3 and learning rate of 0.2 were used. In Figure 6 (b) it can be seen that the traditional blocks of M-KD-C-FP as well as S-MP-V groups together. There are a few MPs that seems to deviate from the party line, which is probably due to missing votes and the fact that not all MPs always vote in line with the party. In Figure 6 (c) it can be seen that the distribution of votes over different genders seems to be randomly distributed. In Figure 7 it can be seen that it does not seem to be any systematic differences between the votes of the districts.



(a) (b) (c)

Figure 6: (a) Plot of the training points (city coordinates) and the suggested tour / cycle. (b) Plot of the 2D 10×10 grid for every MPs 31 votes, by party. (c) Plot of the 2D 10×10 grid for every MPs 31 votes, by gender.

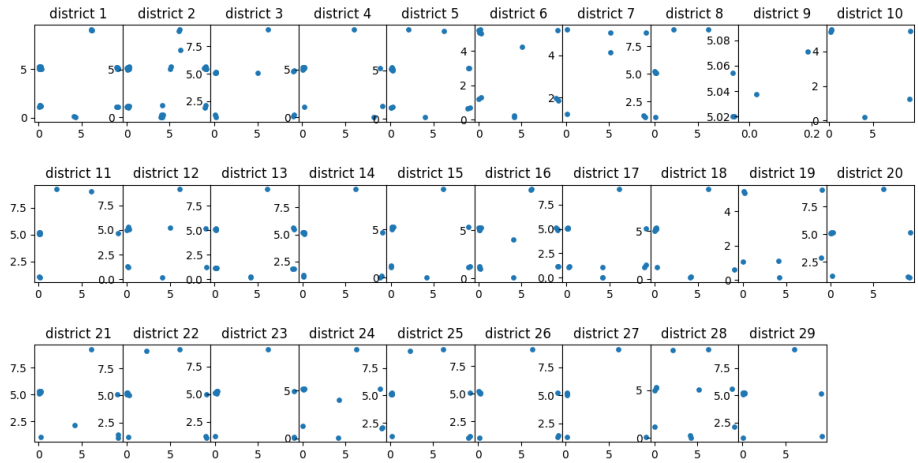


Figure 7: Plot of the 2D 10×10 grid for every MPs 31 votes, by district.