Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Ming Jiang, Ziyuan Wang

February 10, 2021

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to build the RBF network and analyze its performance under different parameters setting
- to understand and analyze different methods for weights initialisation in RBF network
- to implement SOM network and apply it for multiple tasks

The scope of this particular assignment is well-structured and fulfilled for our capacity. And the limitation for the given project is that some tasks are similar but are separated into different sections.

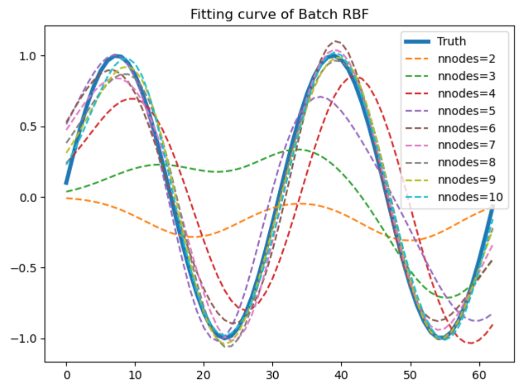
2 Methods

The coding language for our task is built upon Python 3.6 due to the overall universality for dependences. For the entire project, we conduct the code based on NumPy 1.19.2 and Matplotlib 3.3.3 for visulization.

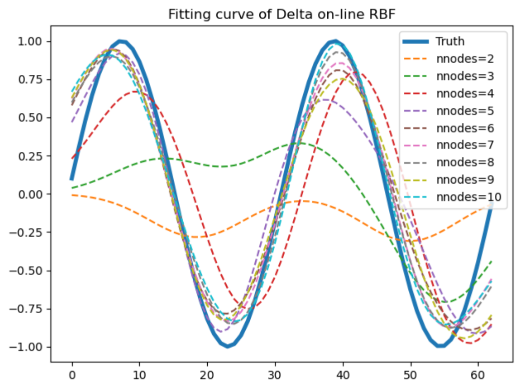
3 Results and discussion - Part I: RBF networks and Competitive Learning

3.1 Function approximation with RBF networks

We initialize the width $\sigma = 2 * \pi \ / \ \mathit{nodes\_number}$, and vary the hidden nodes number within the range [2, 11). And for the sin(2x) function we plot the fitting curves on test sets with training



(a) Batch Mode



(b) Delta On-line Mode

Figure 1: Fitting Curve with Different Hidden Node Numbers($width = 2 * \pi / nodes\_number$)

in batch mode and delta on-line mode. The results figures are shown in Fig. 1.

Next, also with $\sigma = 2 * \pi / nodes\_number$, Table 1 shows the absolute residual error changed by hidden nodes numbers trained in batch mode for the sin(2x) function.

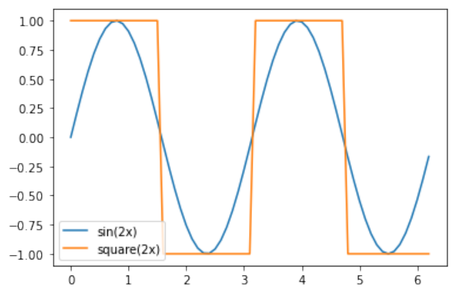
Table 1: MAE Thresholds and Hidden Nodes Number

Threshold	Absolute Residual Error	Hidden Nodes
0.1	0.0824	9
0.01	0.0078	17
0.001	0.0008	34

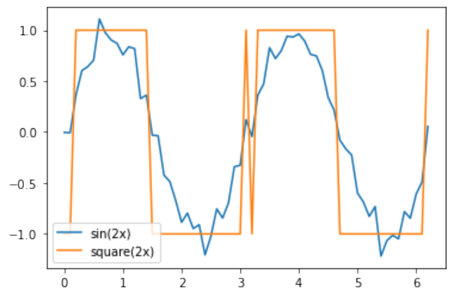
- Within the range we set, the curve tends to fit better with the increase of hidden nodes number.
- When the hidden nodes is beyond 34, the absolute residual error is less than 0.001, which means that the nodes number is enough to fit with the dataset.
- For the square(2x) function, we can set the output threshold to reduce the residual error to 0. And this kind of output from RBF is actually a classification problem.

3.2 Regression with Noise

Within the instruction, we can see the distribution of dataset with or without noise in Fig 2. The comparison between different parameter on both clean data and noisy data is given in Table 2. The foundation parameter is $epochs : 1000, learning\_rate : 0.01$. We can see that, on clearn data, with the same amount of RBF units, the smaller the width within rationale, the best the performance. However, in the noisy data, we can observe more fluctuation. As for the best performance configuration, the parameter is given $nodes : 20, sigma : 0.2$ on clean data, and it



(a) Clean Data



(b) Noisy Data

Figure 2: Data Comparison, Noise Level: $\sigma = 0.1$

is close to the best performance on the noisy dataset, so we decided this set of parameter is the best overall.

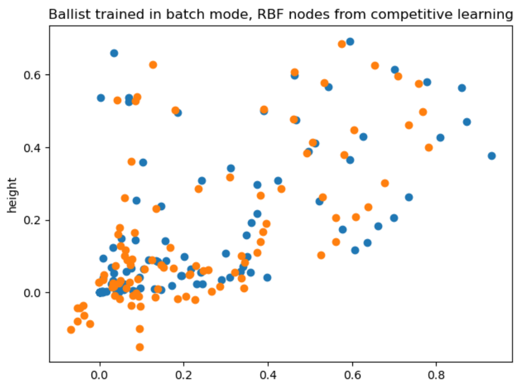
Table 2: Grid Search for $sin(2x)$ and $square(2x)$ Approximation

Functions:		$sin(2x)$				$square(2x)$			
		Clean Data		Noisy Data		Clean Data		Noisy Data	
Hidden Nodes	Sigma	Batch	Online	Batch	Online	Batch	Online	Batch	Online
8	0.2	0.1019	0.1053	0.1456	0.1445	0.0	0.0	0.0317	0.0317
8	0.6	0.0862	0.116	0.1237	0.1255	0.0635	0.0952	0.0635	0.0952
8	1.0	0.0722	0.1951	0.1234	0.1757	0.0317	0.0952	0.0317	0.0635
16	0.2	0.0239	0.0898	0.1028	0.1383	0.0952	0.0635	0.0635	0.0317
16	0.6	0.0064	0.1026	0.1013	0.1153	0.0635	0.0317	0.0317	0.0317
16	1.0	0.0032	0.15	0.0979	0.2068	0.031	0.127	0.0635	0.0317
20	0.2	0.019	0.0625	0.1046	0.1364	0.0635	0.0635	0.0317	0.0
20	0.6	0.0038	0.1173	0.1002	0.1206	0.0952	0.0	0.0317	0.0317
20	1.0	0.0029	0.1242	0.0994	0.145	0.0952	0.0317	0.0	0.0317

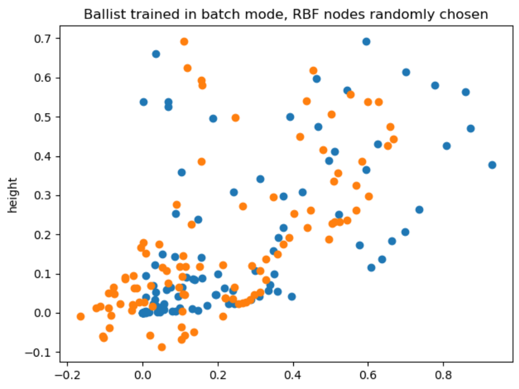
As for the error measurement methods, we perfer the MAE due to is do not reduce the error for given model and stay consistence with the network itself, we do believe that this could be the better choice instead of MSE. And to compare the importance of eta, we given the parameter of $eta : 0.01, 0.001, 0.0001$, we the error decline with the incline of error from 0.0914 to 0.1799, and we could say that the training process is not trapped in a local minimal.

The meaning of sigma in RBF Kernel is that it makes it easier for each hidden layer neuron to realize the ability to perceive local information, which is beneficial to improve the local response ability of RBF neural network. And we found out that with evenly distribution of RBFs centers is most suitable for the generalization and localization. We set the centers evenly distributed acorss the input space and randomly choose various nodes in the input space, we observe a significant gap in performance with the parameter of $Nodes : 20, eta : 0.01, epoch : 20$, and for the evenly distributed set, the error is 0.0952, as for the random set, the error is 0.9524.

As for performing on the clean data with a model trained on noisy data, we found out that the error of clean data is 0.0444 which lower that on noisy data itself, stays in 0.0861. And since the noise is not disturbing the distribution of given dataset, or the flow of given dataset, the result is understandable.



(a) Good Initialization of RBF Nodes



(b) Bad Initialization of RBF Nodes

Figure 3: Predictions on the 2-D Ballist Data(σ = *1.0, nodes_number* = *12*)

And for the comparison with the single-layer of perception network, we can see the result in Table 3, As we can see that in this case, with projecting the input space into a higher dimension space, we could see that the robustness of a RBF network is much higher that the perceptron network.

Table 3: Compassion between RBF and Perceptron

Function	Hidden Nodes	RBF MSE	Perceptron MSE
$sin(2x)$	8	0.0509	0.4428
$sin(2x)$	12	0.0338	0.4362
$sin(2x)$	20	0.0277	0.4389
$square(2x)$	8	0.381	1.3968
$square(2x)$	12	0.3175	1.3333
$square(2x)$	20	0.127	1.3333

3.3 Competitive learning for RBF unit initialisation

We first use the competitive learning in batch mode for the sin(2x) function. Here we set the hidden nodes number as 8, and σ = 1.0. Compared with Table 2, when the parameter configuration remains the same, predictions with competitive learning achieves the MAE as 0.0912, which is smaller.

We next utilize the RBF network for the given two dimensional data. The result is displayed in Fig. 3a, in which the absolute residual error is 0.0334 for the *distance* label, and 0.0232 for the *height* label. The RBF nodes for competitive learning we use here is randomly chosen. We also plot the fitting result when the randomly chosen nodes are not good enough in Fig. 3b. And in this situation, the the absolute residual error is 0.0628 for the *distance* label, and 0.0378 for the *height* label.

- Competitive learning can help us to correct the initialized RBF nodes for a better training.
- To avoid the dead units, we update the worst node, which has the largest distance, with the square of learning rate(smaller than the original learning rate).
- The performance of RBF network relies much on the initialization of RBF nodes. If we randomly choose them, then the final results will vary with the selection.

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

4.1 Topological ordering of animal species

We set the parameter of $radius : 50, eta : 0.2, epochs : 20$ training all the animals into an order that make sense. The result could be seen in Table 4. And we can see that the ordering is given by: the first cluster is Insect, followed by Birds, merching into Amphibians, Reptiles and Mammals. The start of each clusters are bold in Table 4. However, the ordering do vary based on different initialization.

Table 4: Re-ordered Animal List

Animal	**spider**	housefly	moskito	butterfly	dragonfly
Animal	grasshoppe	beetle	**pelican**	duck	penguin
Animal	ostrich	**frog**	**seaturtle**	crocodile	**horse**
Animal	rabbit	elephant	bat	rat	skunk
Animal	hyena	bear	dog	walrus	lion
Animal	cat	ape			

4.2 Cyclic tour

Based on the SOM algorithm, we could get a relatively decent result for a visiting tour in Fig. 4, however, the tour varies based on the initialization.

4.3 Clustering with SOM

We use the SOM model built before for the *Votes* dataset. Fig. 5 display the clustering results, with different colors represent different labels. For visualization of the results, we add some noise into the clustering results to reduce the overlapping.

- The gender clustering and district clustering results, as shown in Fig. 5a and Fig. 5b, have a lot overlapping. The clustering results represent the voting pattern, thus the overlapping explains that these two attributes have no obvious relation to one's voting pattern.

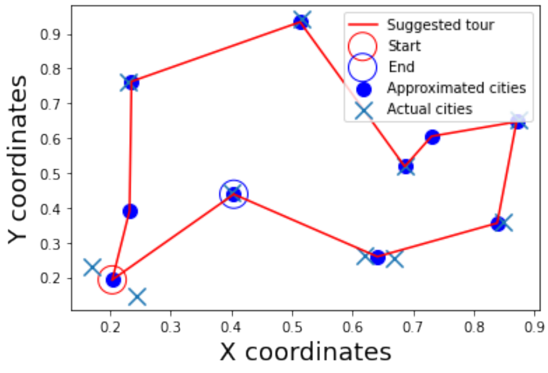
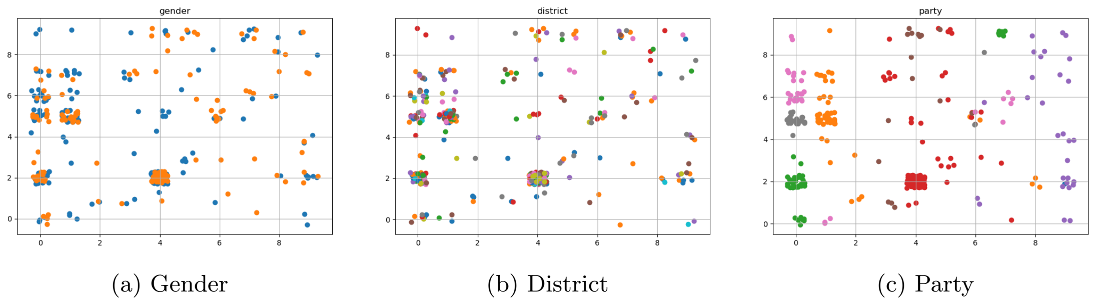


Figure 4: The recommended Tour Plot



(a) Gender (b) District (c) Party

Figure 5: Voting Clustering Results with Different Attributes(*epoch=100, eta=0.2, radius=50*)

- In the party clustering result, as shown in Fig .5c, same color points(means the same party) are close to each other and not spread a lot. This might implies some relation between the party and voting pattern of an individual.

5 Final remarks

In this lab session, we have learnt the RBF network and SOM network from a methodology perspective as well as a practical perspective. The overall experience is significantly beneficial for understanding the foundation of RNN. And we have a better understanding of the limitation of percptron network, the robustness and limitation of RBF, SOM networks.