EE5904/ME5404 Neural Network Assignment3

Wang Jiangyi
National University of Singapore

1 Question 1: Function Approximation with RBFN

Consider the following function:

$$y = 1.2sin(\pi x) - cos(2.4\pi x) \quad for\ x \in [-1.6, 1.6]$$

The figure of the function is:

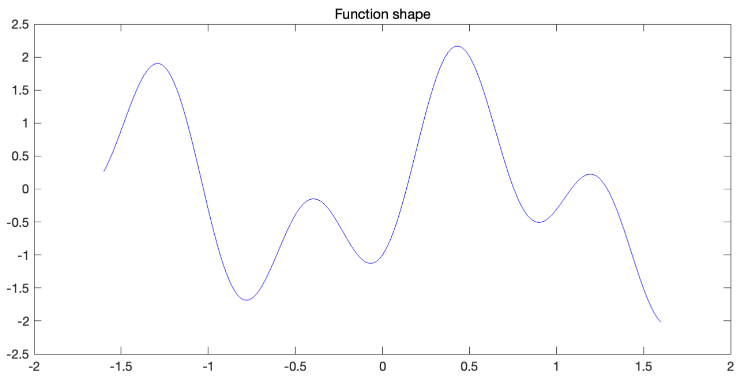

Figure 1: Shape of function

To discuss the robustness of RBFN, we introduce **Random Noise** to training set, that is,

$$y(i) = 1.2sin(\pi x(i)) - cos(2.4\pi x(i)) + 0.3n(i)$$
$$n(i) \sim Gaussian(0, 1)$$

Note that there is only Random Noise on training set. The target of testing set is exact function value.

1.1 Question a) Exact Interpolation Method

Here, we choose Gaussian function as RBF (Radial Basis Function) whose width (standard deviation) equal to 0.1. The Approximate Curve is shown as follows:

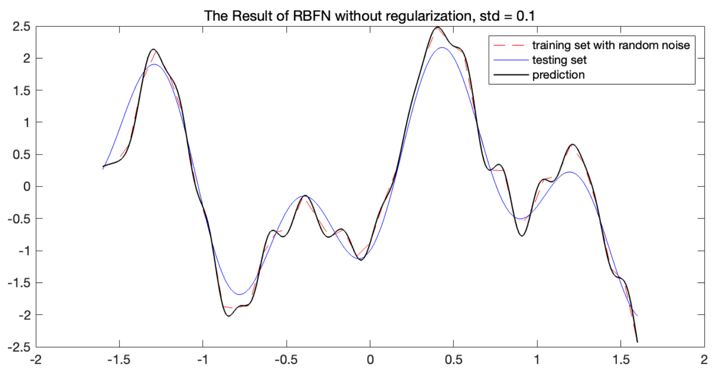

Figure 2: Approximate Curve By RBFN

To be more precise, we use the **MSE quantity** to justify the performance of Exact Interpolation RBFN. In this model (with this achievement of Random Noise), the **MSE of training set is exactly 0 and that of testing set is 10.1166**. This shows that Exact Interpolation Method without regularization leads to **over-fitting dramatically**.

Moreover, we can check the error of each testing sample, and the result is shown in Figure 3:

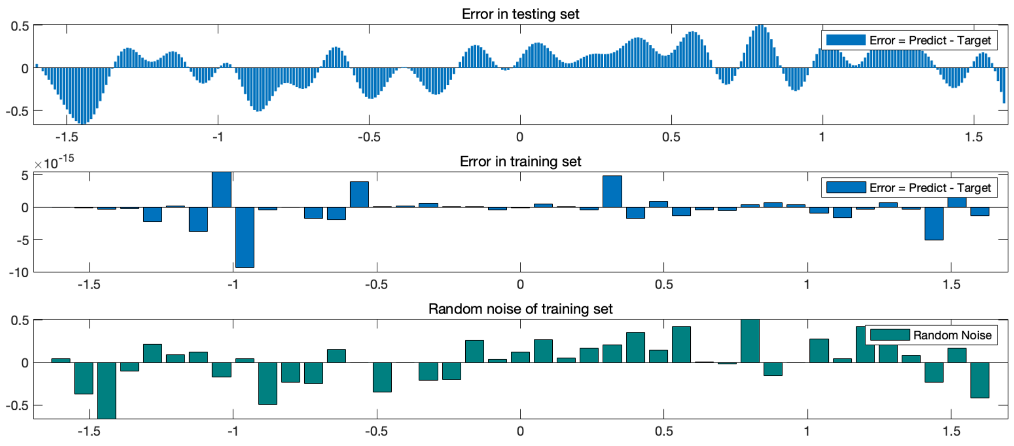

Figure 3: Relation between Error and Random Noise

In the second subfigure, it shows that RBFN perfectly fits the training set. Then, we focus on the first and third subfigure. It can be observed that, the **Error of Testing set** is **highly related** to the **Random Noise of Training set**. We can draw the conclusion that, **RBFN (with Exact Interpolation Method) will be influenced greatly by Random Noise.**

1.1.1 Extra Part: Width of RBF (Guideline)

In the previous experiment, we fix the standard deviation of RBF at **0.1**. In this part, we want to consider the influence of standard deviation for RBF. We choose 2 special std, 0.001 and 10 respectively and the result is shown in Figure 4:

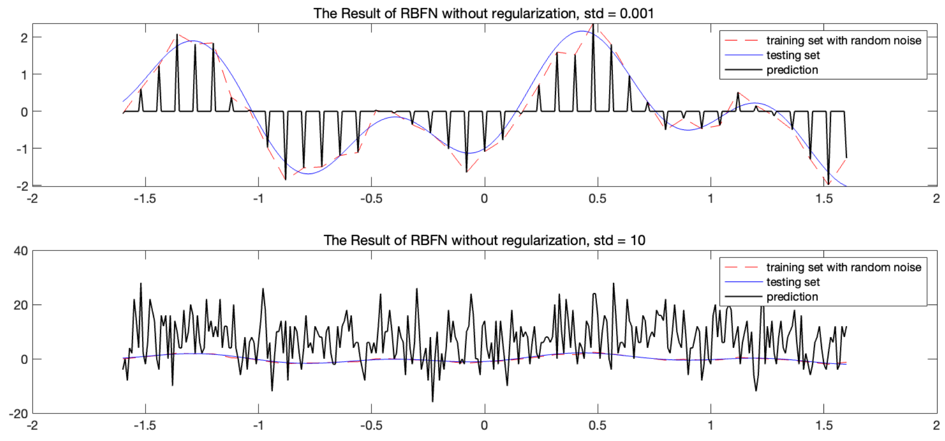

Figure 4: Different width of RBF, 0.001 and 10 respectively

Conclusion: Small std leads to over-fitting and large std leads to under-fitting.

Reason: For small std (width), each RBF tends to be shaper, which means **it is only active in very small region near the center**. Therefore, if we choose the number of center equal to training sample, it will

defintely **achieve 0 training error** (strongly over-fitting).
For large std (width), each RBF tends to be flatter, which means it will **behave almost the same in all the domain**. If so, it is very difficult for us to use this kind of feature to fit any curve (under-fitting).

Therefore, we should choose the **intermediate value** of std (width) like 0.1 (guideline of choosing std).

1.2 Question b) Fixed Centers Selected at Random

Here, we select 20 Random Centers from the training set, and in order to make comparison, we continue to use the **previously generate Random Noise**. What's more, we determine the std adaptively, that is,

$$std = \frac{d_{max}}{\sqrt{2M}}$$

With this special parameter setting, the Approximate Curve is shown in Figure 5:

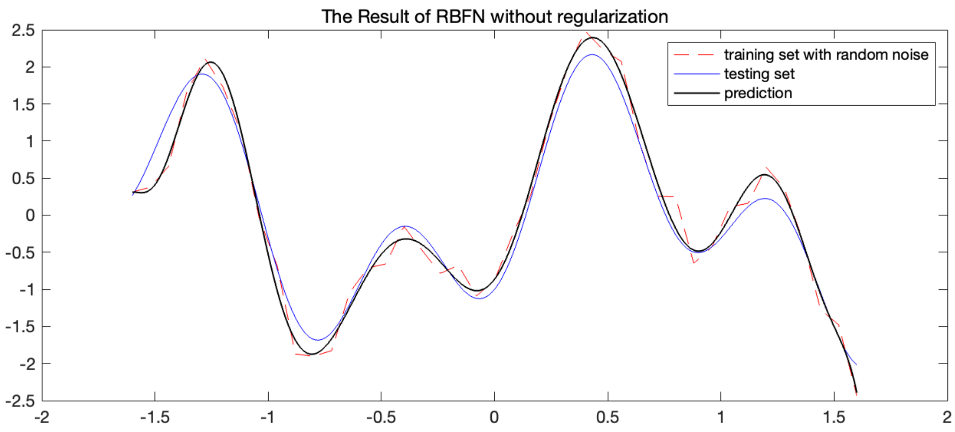

Figure 5: Approximate Curve by RBFN

Here, MSE for training set is **0.4931**, which is no longer MSE for testing set is **6.4599**, which is much smaller than **10.1166**. This shows that if we decrease the number of centers (Fixed Centers Selected at Random), the **problem of over-fitting releases**. However, it **still suffers from the over-fitting** problem because 6.4599 MSE is still very large.

We can still check the error of each testing sample, and the result is shown in Figure 6:

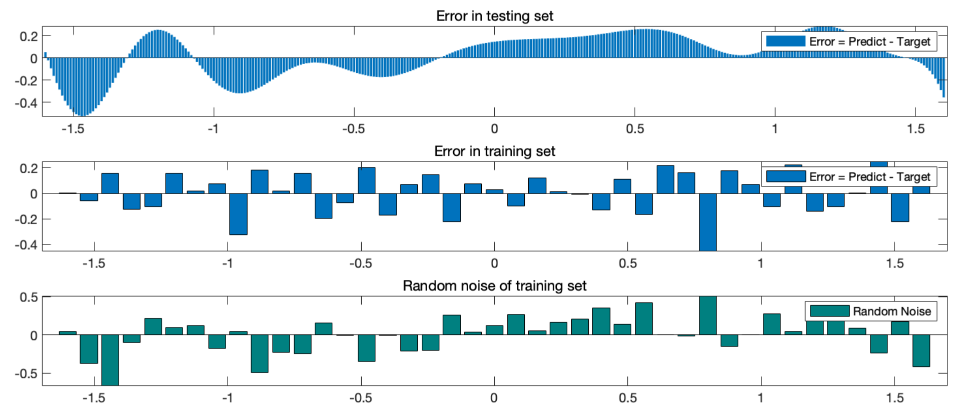

Figure 6: Error of training and testing set

From Figure 6, it can be observed that, with FCSR (Fixed Center Selected at Random), the **error in the interval** $[-1, 1]$ **has been flatter**, compared with Exact Interpolation Method. And this is the reason that the total MSE is smaller.

To conclude, with FCSR Method, the perfomance (according to MSE) of testing set is **much better** compared with Exact Interpolation Method. However, it still **suffers from the over-fiiting** problem, **especially in the interval** $[-1.6, -1.3]$ **and** $[1.3, 1.6]$.

1.2.1 Extra Part: Regularization for FCSR

Consider the Regularization for FCSR Method. In order to make comparision, we keep all other parameters the same, except adding the regularization term. Here, we choose **6** regularization factor, which is 1, 0.0001, 0.00001, 0.000001, 0.0000001 and 0 respectively.

The perfomance (MSE) of 6 different regularization factors is shwon in Figure 7, including the training and testing set:

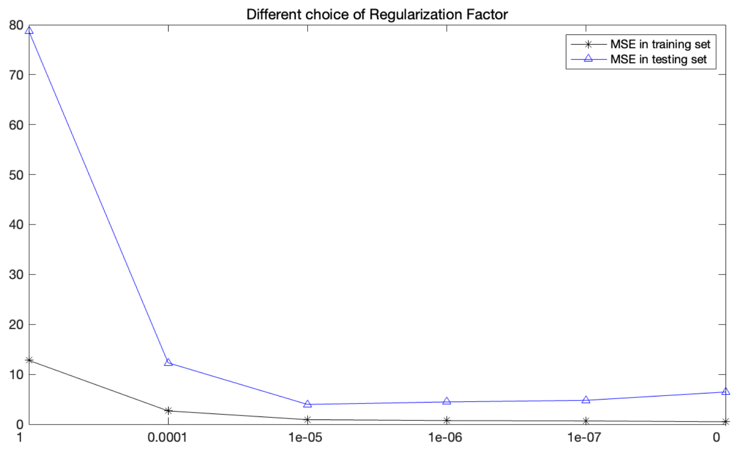

Figure 7: Different choice of regularization factors

It is obvious from Figure 7 that, for large regularization factor, it will be under-fitting. If we choose regularization factor small, but not too small, i.e., 0.00001, then we may achieve the best performance, which is **much better than the case without regularization**.

The Approximate Curve achieved by regularization factor = 0.000001 is shown as Figure 8 (compared with the case without regularization term):

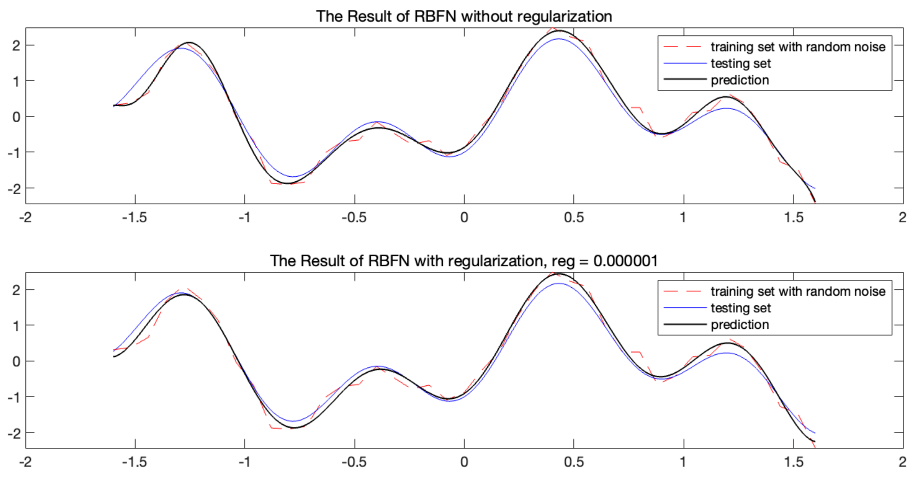

Figure 8: Different choice of regularization factors

From Figure 8, we can observe that with the help of regularization, the curve is smoother, **especially in the interval** $[-1.6, -1.3]$ **and** $[1.3, 1.6]$. And this is the problem we met in the previous case without regularization. That is, the prior knowledge of Regularization Factor greatly releases the over-fitting problem in these 2 intervals, **guiding the curve to be smoother**.

Then we check the perfomance of this case:

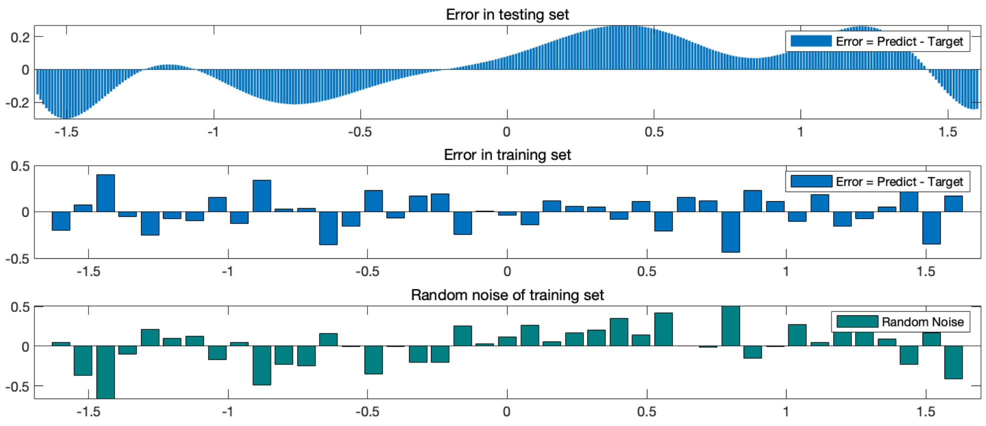

Figure 9: Error of training and testing set

In Figure 9, notice that **the range of testing set error shrinks to** $[-0.2, 0.2]$, which is $[-0.5, 0.5]$ previously without regularization. Additionally, **MSE for testing set is 3.7063**, which is much smaller than **6.4599 (without regularization)**.

To conclude, **with appropriate choice of regularization factor, we can release the over-fitting problem greatly.**

1.3 Question c) Regularization for Exact Interpolation Method

Lastly, we consider **Regularization** for Exact Interpolation Method. Here, we choose the regularization factors from [0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.8, 0.9, 1, 1.1, 1.2, 2]. We want to find the **best regularization factor** among the 13 candidates. The Performance of training and testing set is shown in Figure 10:

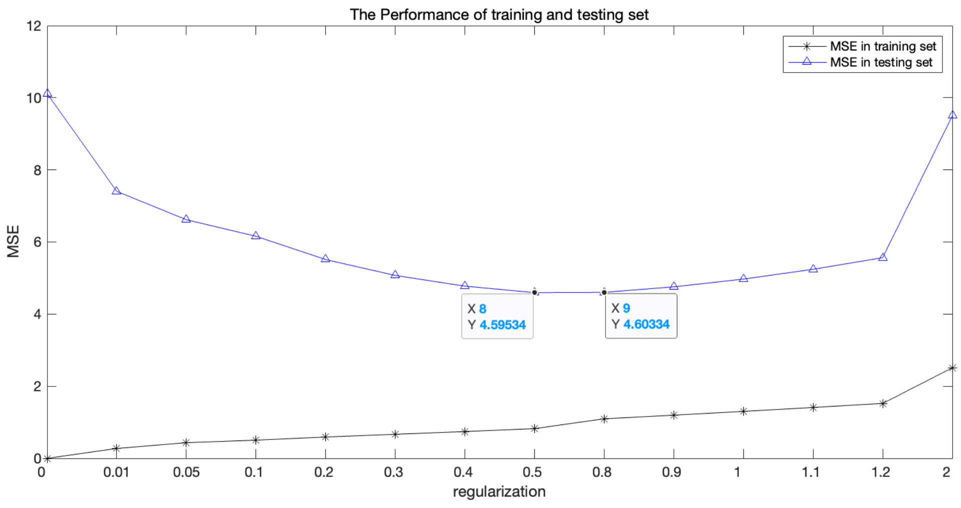

Figure 10: Error of training and testing set

From this figure, we can deduce the Best reugularization factor is around 0.5 and 0.8. Compared with the

testing set MSE without regularization, 10.1166, that for **Best regularization factor is around 4.59**, which is **much better.** The cost for this improvement is the increase in training set MSE. But, actually, what we concern is just the generalization ability of our model, i.e., the testing set MSE.

We can further check the Curve of the fitting result:

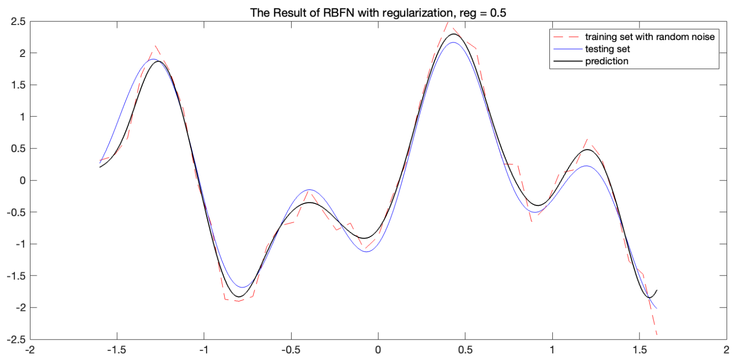

Figure 11: Fitting Curve for Regularization Method

Compared with Exact Interpolation without regularization, this curve is **much smoother**, implying the **better generalization ability for regularization method**. Although it is still **slightly worse** than the result of FCSR with regularization (**3.7063 Testing Set MSE**), it is still a good result since it releases the extent of over-fitting when **using around 300 centroids**.

To conclude, the regularization term releases the extent of over-fitting greatly, while better results can be achieved by decreasing the number of centers (FCSR with regularization).

2 Question 2: Handwritten Digits Classification

My student ID is: A0236307J. Therefore, in my experiment, class 0 and class 7 should be assigned label 1, while remaining classes should be assigned label 0.

2.1 Question a) Exact Interpolation Method and Regularization

In question part, we choose Gaussian RBF with fixed std of 100. In extra part, we will discuss the best choice of std.

2.1.1 Without Regularization

After using **Exact Interpolation RBFN without regularization**, we can achieve the performance which is shown in Figure 12:

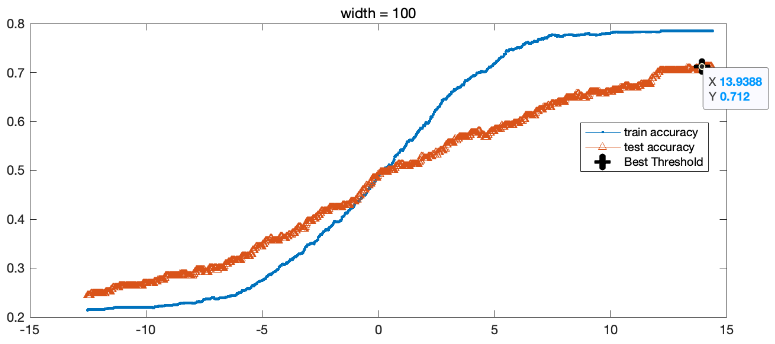

Figure 12: Performance changing with threshold

Explanation:
In Figure 12, the RED line represents the change of testing accuracy according to different threshold, and the BLUE line represents the change of training accuracy according to different threshold. R**BFN only gives us the Approximate Function of the Target**, In order to translate the value of Approximate Function (RBFN) into the predicted label, we use **step function with threshold**:

$$\begin{aligned} predicted\ label(x) &= step(x - threhold) \\ &= \begin{cases} 1, & x > threshold \\ 0, & x \le threshold \end{cases} \end{aligned}$$

The Best Threshold is chosen according to the testing accuracy. Here, from Figure 12, we can achieve that the **Best Threshold is 13.9388** and the **corresponding training accuracy is 0.712**.

2.1.2 Extra Part: Different std without regularization

Actually, we can choose different std, and here we choose 4 different std, which is 0.1, 1, 10, 1000 respectively. The results of performance are as follows:

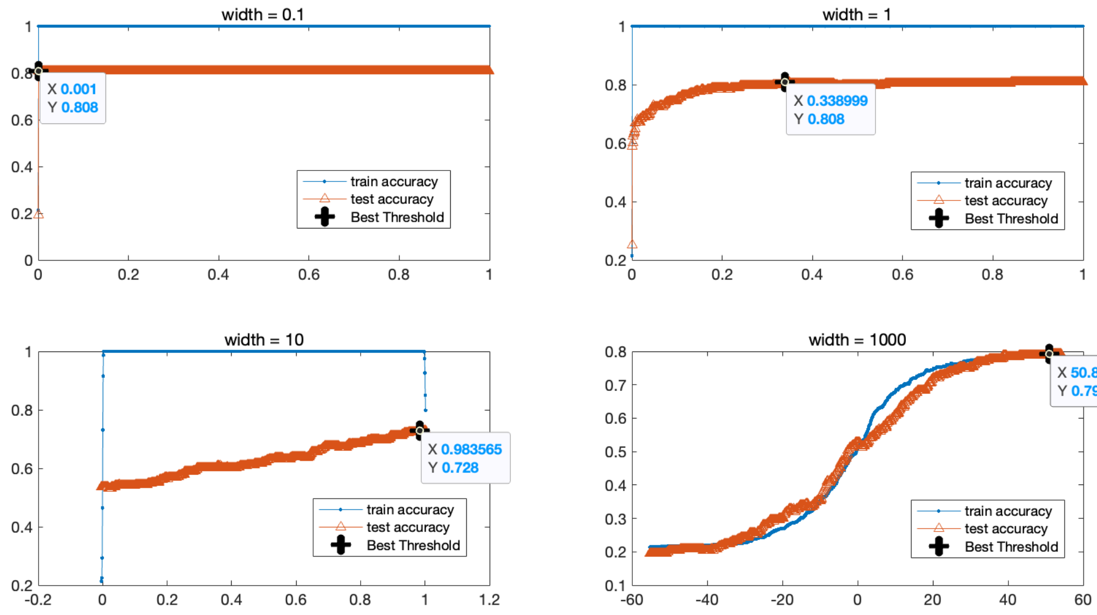

Figure 13: Performance for different choices of std

Explanation(Comments):
1. Subfigure 1 proves that, as for Exact Interpolation Method, if we choose std small enough, then we will definitely achieve 0 training error (simply by set weight equal to training target for binary classification problem). This will lead to the **disaster** that, **all testing samples will be predicted to label 0**. This conclusion corresponds to the conclusion we attain in Question 1 (**Small std will lead to over-fitting dramatically**).
This is the case we met in subfigure 1. Since this problem is **Category-Imbalance**, which has around 80% samples whose labels are 0. Therefore, predicting all testing set to label 0 will achieve 80% training accuracy. This is what we do in Subfigure 1.

2. Furthermore, for all choices of std, if we set the threshold **big** enough, we can always achieve 80% training accuracy. Therefore, the results here are not good actually although the testing accuracy is quite high.

3. In principle, Exact Interpolation Method will definitely give 0 training error. However, when std=100 and std=1000, **we cannot achieve that**. The reason is that, while std is big enough, then the matrix

$\Phi_{(i,j)} = \phi(|x_i - x_j|)$ has **extremely large condition number**, implying that we are unable to solve the equation (for Exact Interpolation Method):

$$\Phi w = d$$

Therefore, we are unable to achieve the exact solution w. **Exact Interpolation Method breaks down!**

Next, we are supposed to focus on the small value of std. Then, we take std = 4.3 as an example. We can achieve a deeper intuition of this problem by visualizing the value of RBFN according to different target:

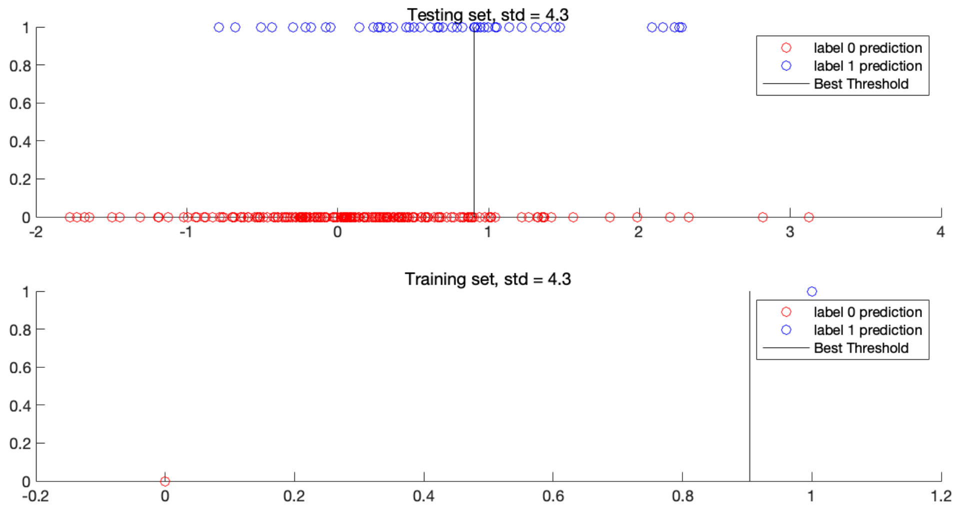

Figure 14: The mapping value of RBFN for different classes

From this figure, we can observe that, we can achieve 0.804 accuracy on testing set. Meanwhile, we can also **classify 20 Class-1 samples correctly**. Of course we can achieve 100% training accuracy on training set since the std is small.

We can also get the intuition that, in our model (with threshold step activation function), RBFN is trying to **distinguish these two-classes images from each other as far as possible**.

To conclude, Exact Interpolation Method without regularization depends much on the choice of std.
If we choose **std=100**, **it performs rather bad on testing set**;
If we choose **std=4.3**, **it performs relatively better**. Actually std=4.3 is almost the best parameter we can obtain only use Exact Interpolation Method without regularization.

2.1.3 With regularization

Here, we still fix std=100, and consider the influence of regularization factors, which are [0.001, 0.01, 0.05, 0.1, 0.2, 0.4] respectively. The testing accuracy of them is shown as Figure 15:

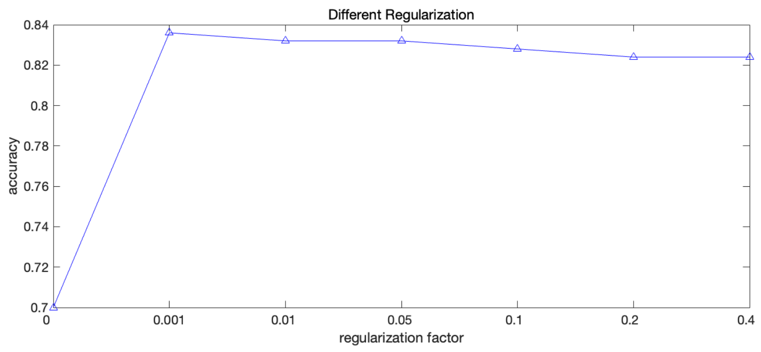

Figure 15: Performance of different regularization factors

According to Figure 15, we should choose **regularization factor=0.001**, and its perfomace curve is shown as follows:

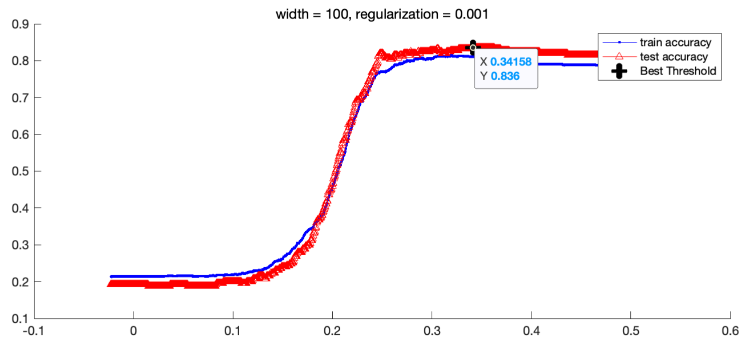

Figure 16: Performance curve of regularization factor=0.001

It shows that, when we add regularization factor=0.001, the testing accuracy **increases from 0.7 (without regularization) to 0.836**. Therefore, for fixed std=100, the **perfomance of RBFN with Exact Interpolation Method has been improved greatly** by adding the regularization term.

2.1.4 Extra Part: Better choice of std

In the previous part of discussion, we draw the conclusion that **std=4.3 is a rather good choice of parameter**. Then, we will consider the regularization term on this model.

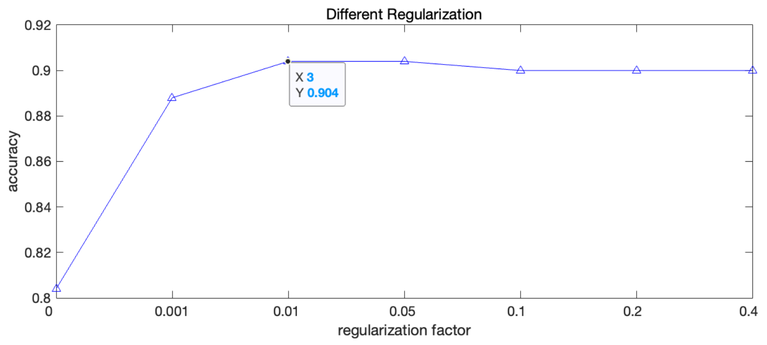

Figure 17: Performance of different regularization factors

According to Figure 17, we should choose **regularization factor=0.01**, and its perfomace curve is shown as follows:

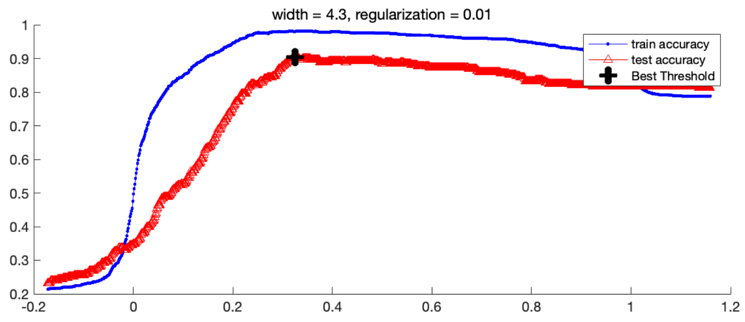

Figure 18: Performance curve of regularization factor=0.01

From Figure 18, with appropriate choice of threshold, we are able to **achieve 0.904 testing accuracy**, which is even higher than 0.804 (without regularization). To check whether this model is able to predict the Class-1 testing samples correctly, we visualize the corresponding RBFN:

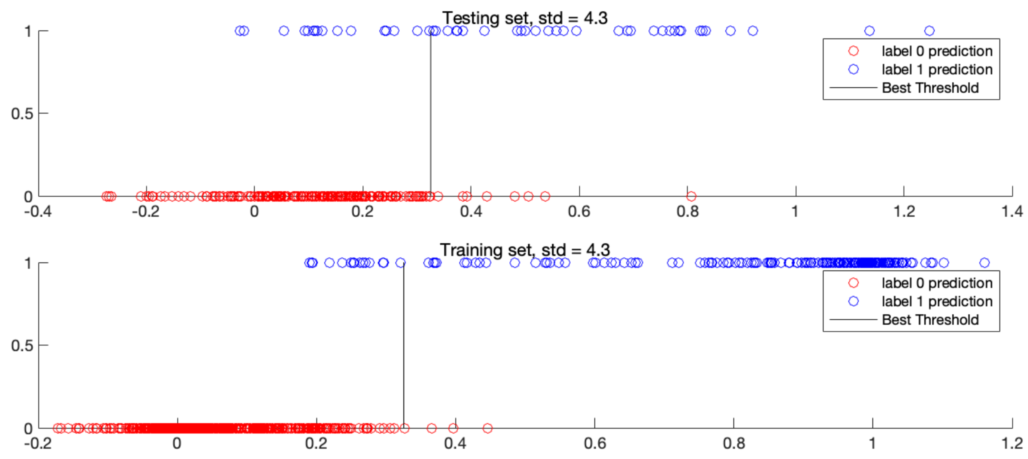

Figure 19: The mapping value of RBFN for different classes

In Figure 19, we can attain the conclusion that, there are all **32 Class-1 testing samples are predicted correctly** and there are all **48 Class-1 testing samples**! Previously, the number is 20 (without regularization). It is actually a **wonderful result** for this problem!

From this example, the power of regularization totally demonstrates. Moreover, Exact Interpolation Method greatly depends on the choice of std. If the std is appropriate, i.e., 4.3, then we will have great performance.

2.2 Question b) Fixed Centers Selected at Random (FCSR)

Here, we apply FCSR Method with RBFN and **fix the number of centers at 100**.

2.2.1 Adaptive width

Firstly, we choose an appropriate width by th following formula:

$$std = \frac{d_{max}}{\sqrt{2M}}$$

where M represents the number of centers and d_{max} represents the maximum distance between centers.

The perfomance curve according to threshold is shown as follows:

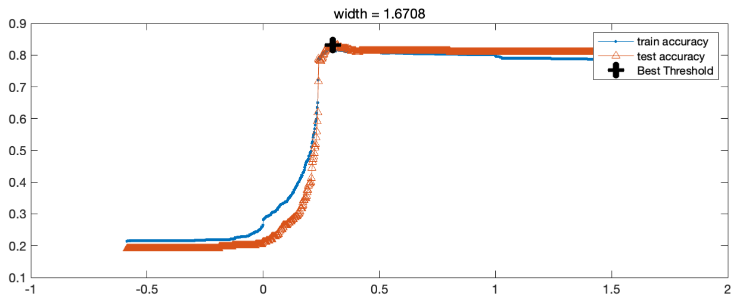

Figure 20: Performance Curve for Adaptive Width

In Figure 20, the best testing accuracy is around 0.832. Compared with the result in a), actually its performance **is similar to** the case that **width=100 with regularization factor=0.001**. And its performance is obviously **poorer** if compared with the case that choosing **width=4.2 and regularization factor=0.01**.

Also, we can visualize the value of RBFN mapping:

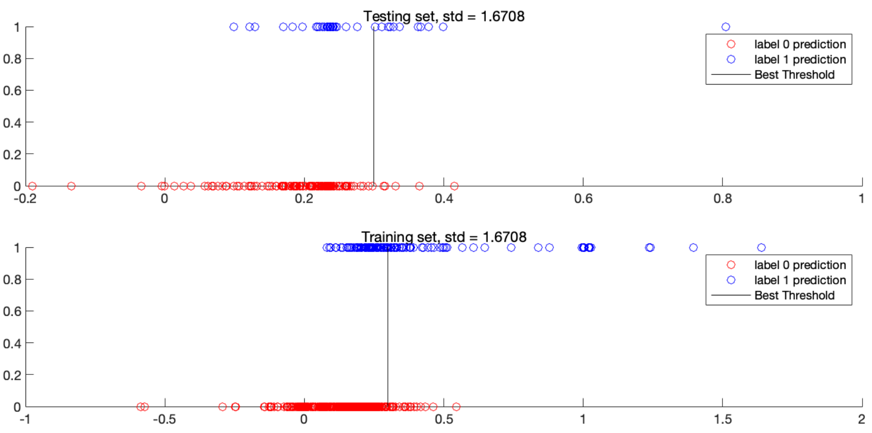

Figure 21: The mapping value of RBFN for different classes

There are all **11 Class-1 testing samples** which are correctly predicted.

To conclude, with adaptive width, the performance of RBFN is relatively moderate. It is **slightly better** than the case that std=100, but **much worse** than the case that std=4.2 in Question a).

2.2.2 Selecting the best width

Here, we aim to select the width from [0.1, 0.5, 1, 3.8, 4.3, 5, 10, 100, 1000, 10000]. The training accuracy of each choice of width is shown in Figure 22:

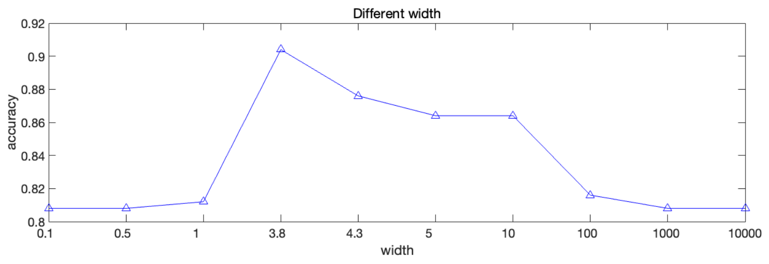

Figure 22: Performance of different widths

From Figure 22, obviously we should choose **width=3.8** to achieve the largest training accuracy, which is **0.9**. Its performance curve is shown as follows:

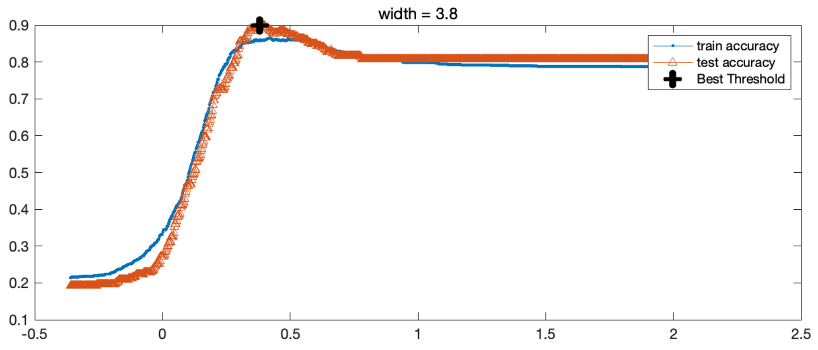

Figure 23: Performance Curve of width=3.8

We can still check the mapping value of RBFN for different classes:

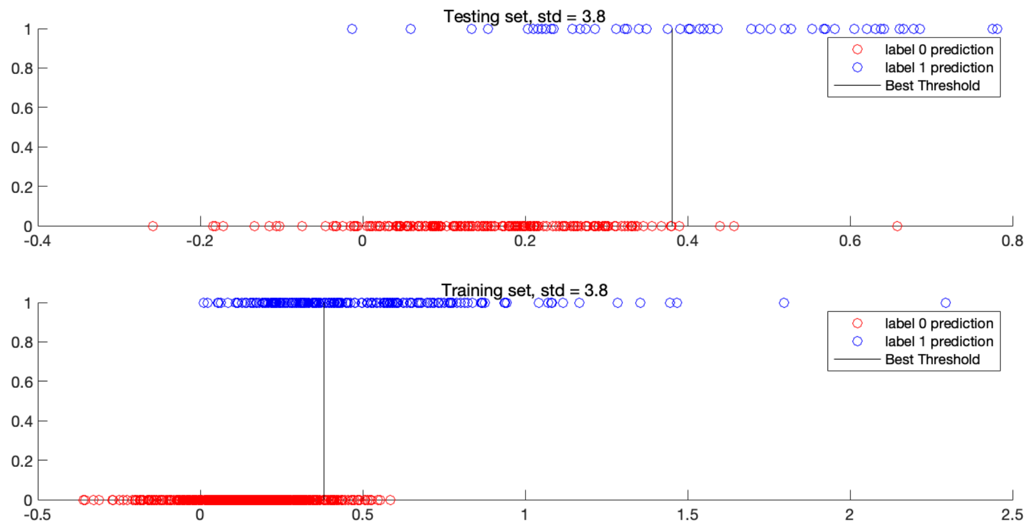

Figure 24: the mapping value of RBFN for different classes

There are all **29** Class-1 testing samples which are correctly predicted, which is **very similar to** the performance of std=4.3, regularization factor=0.01 case in Question a).

To conclude, if we vary the value of width from 0.1 to 10000, the **best choice of width is around 4**. In this case, the performance is very similar to that of std=4.3, regularization factor=0.01 case in Question a). **Both of these two cases are really great results.**

2.3 Question c) K-means Clustering

Here, we fix K equal to 2 (K-means algorithm with 2 centers).

2.3.1 Performance Evaluation

Here, we use adaptive std to train our model (without regularization term). The result is shown in FIgure 25:

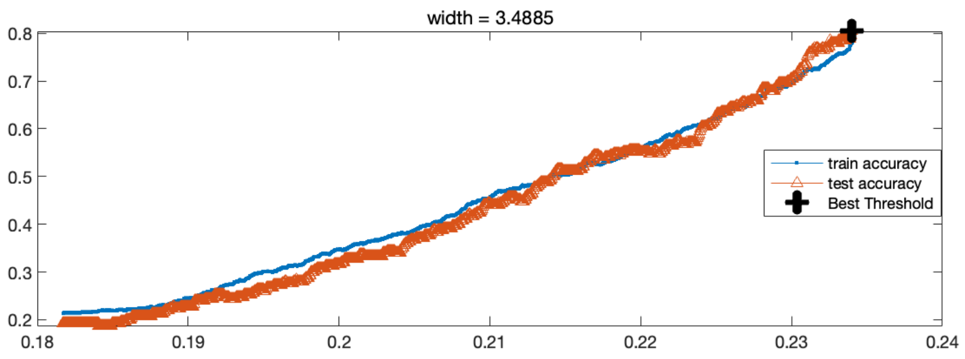

Figure 25: Performance Curve for K-means RBFN

Actually, this Performance Curve is **not good**. Although we can achieve 0.8 accuracy, this is simply attained by predicting all the points to Class-0, which is very naive.

Explanation:

The main reason might be, although we use K-means algorithm to derive the centers, which is a more **reasonable** way than Fixed Centers Selected at Random, we **just pick 2 centers**! The freedom of our model is quite small, and it is very difficult for such kind of model to attain good a good generlization ability.

Therefore, we try a bigger K value, i.e., **K=100**. The result is shown as follows:

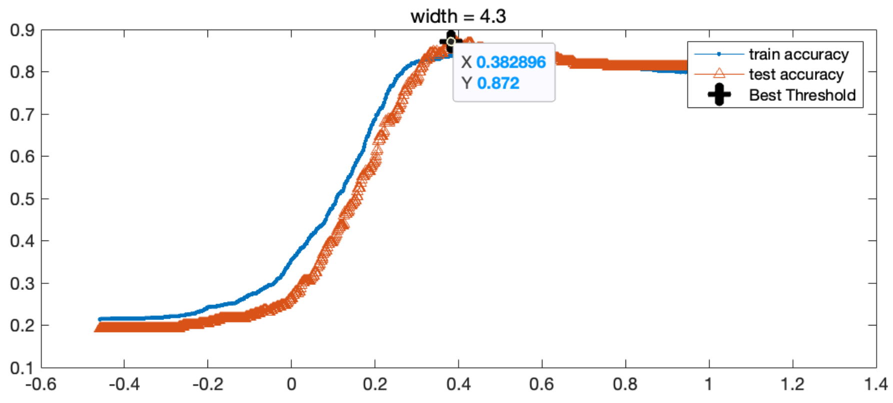

Figure 26: Performance Curve for K-means RBFN

Here, compared with the previous case K=2, the performance curve is **much better** now. Like before, we can further check how many class-1 testing samples can be predicted correctly with this model. The visualization is shown as follows:

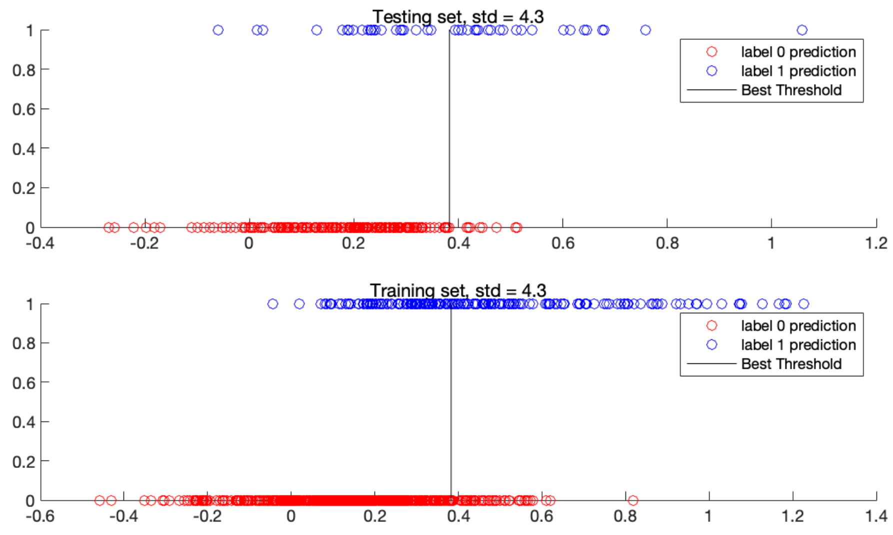

Figure 27: The mapping value of RBFN for different classes

Actually, we are able to classify **25 Class-1 testing samples** correctly. Meanwhile, we can achieve about **0.88 testing accuracy**. This result is very similar to that of FCSR with std=3.8 and 100 centers, which is rather good!

To conclude, RBFN with K-means algorithm still need to choose an appropriate value of K, at least not too small. **When K=2**, it is **difficult** for us to achieve a good generalization result, at least for the std chosen by the formula:

$$std = \frac{d_{max}}{\sqrt{2M}}$$

But **when K=100**, we can still easily **achieve a good generalization result** by setting std around 4 like previous case.

2.3.2 Visualization of obtained centers

Firstly, we consider **K=2** case:

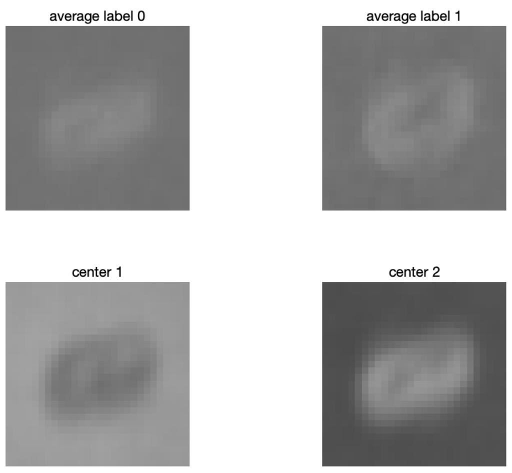

Figure 28: Comparison between centers and mean of two classes

Findings:

1. The **mean image** of twoo classes are **almost the same**.
2. The background (outside regions) of the centers are very different, one is **whiter** and the other is **blacker**. The **reason** might be that, in order to find 2 centers among all training samples, the 2 centers **should be very different, especially in respect to the background(outside regions)**.
3. All of the 4 figures **share the similar property** that, the center region is something between 0 and 8.

Lastly, we visualize the centers when K=100:

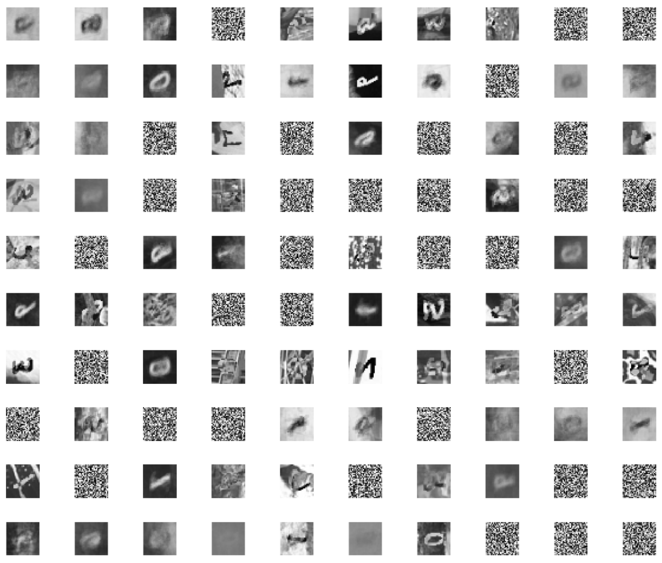

Figure 29: 100 centers when K=100

Findings:

1. In Figure 29, almost all the 10 digits appear.
2. **About half** of the 100 centers are Noise, which are those things we do not want K-means algorithm to learn.

To conclude, if **K is small**, then the centers we find contain **more different generalized** information. This is **our Finding 1 and 2** in K=2 case. **Different** means those two centers represent almost the opposite information (black and white background), and **generalized** means the digit of the center is **not very clear**. If **K is big**, then those centers will **contain more details**. For example, we can see some **clear digits** in those centers. Moreover, those centers may contain **Noise term**. These pieces of information may **be bad for** our task.

Therefore, when we use K-means algorithm, we should not use too big K since those centers may contain much noise (Guideline) ! That is, an intermediate value of K is recommended.

This also shows that, K-means Method **is not definitely better** than Fixed Centers Selected at Random Method supposed that we set the same number of centers. That is, the centers in K-means Method may contain much noisy information (local optimum). This will make no benefits to our result. In this sense, it is critical for us to attain better results from K-means algorithm, which can be achieved by setting reasonable initialization (Now there are some algorithms like K-means++ can achieve that).

3 Question 3: SOM

3.1 Quesiton a) 1-D Output Layer 'hat'

Here, we set the parameter as: 1×40 output layer and 500 iteration. My student ID is: A0236307J, which means I should **consider Claas 0, 1 and 4**. The result is:

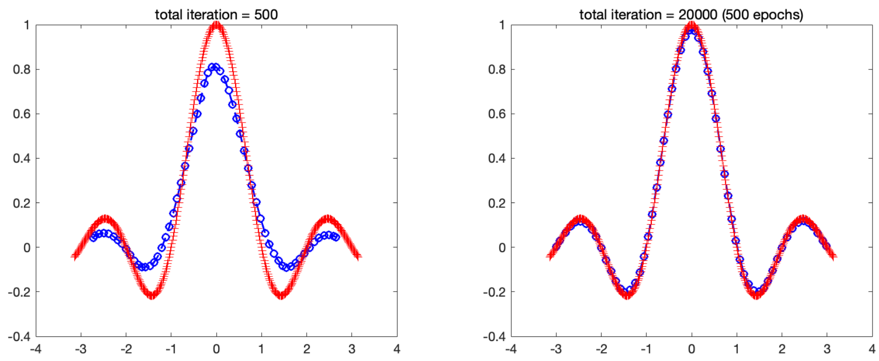

Figure 30: Hat

Comment:

In experiment, actually we try two different ways of initialization:

1. For each dimension, we initialize the weight from interval [0,1] uniformly.
2. For each dimension, we first calculate the range for each dimension. Then, we initialize the weigh within the range uniformly.

It seems to be more reasonable to use **the second way** to initialize, since the initialized weight will be distributed uniformly in the whole space, not just in a corner [0,1]×[0,1]. However, actually it turns out that **the first way** is better. We also **visualize the whole process step by step**, then we obsever that for the first few iteration, those weights tend to squeeze before fitting the curve. The first way will use fewer steps to squeeze with each other before starting to fit the curve.

Therefore, for such problems, we do not have to initialize the weights so that they can be distributed all over the space. It is actually a bad initialization.

3.2 Question b) 2-D Output Layer 'Circle'

Here, we set the parameter as: 8×8 output layer and 500 iteration. The result is:

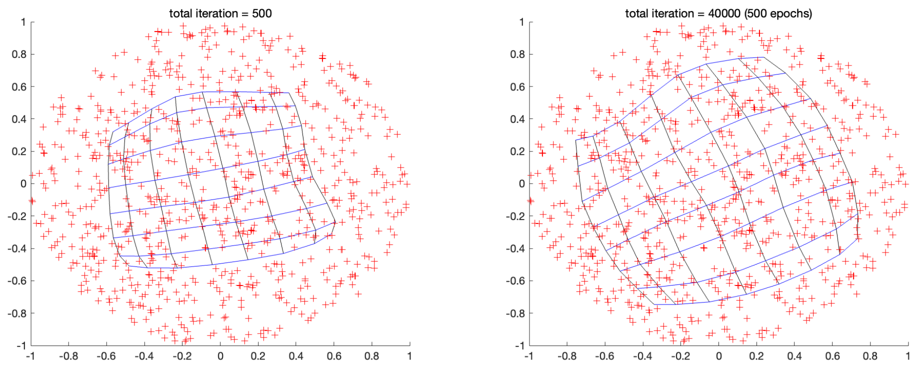

Figure 31: Circle

We visualize the whole fitting process dynamically in MATLAB.

3.3 Question c) Digits

Here, we set the parameter as: 10×10 output layer and 4000 iteration (since we find 1000 iteration will not give us a good result).

3.3.1 Conceptual Map

The Conceptual Map is shown as follows:

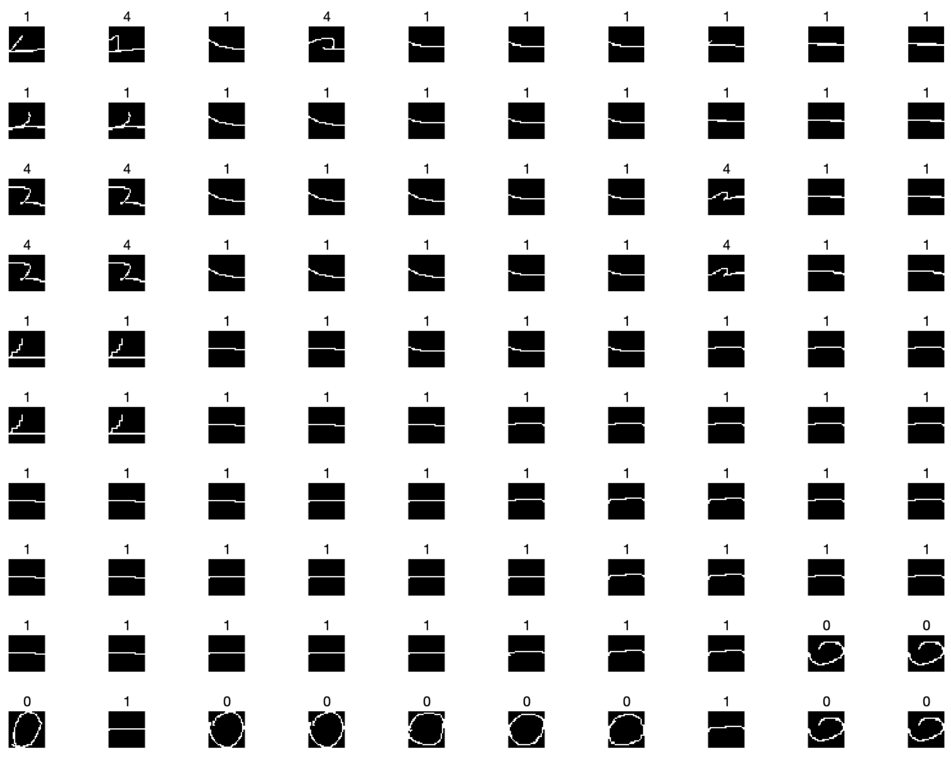

Figure 32: Conceptual Map

Comments:

1. Observation: Class 1 is more likely to be in the center region, while class 0 and class 4 are only appear in the corner region.

Explanation (Guess): Firstly, the property of 2-D SOM is, the center will **be more easily to be influenced by training samples** than the corner. Reason is, using a Mathematics perspective, when the neuron is at center, then its maximum distance with other neuron is about $5\sqrt{2}$ (as for this problem). However, when we consider the neuron in the corner, there are only about 40% neurons whose distance (with corner neuron) is less than $5\sqrt{2} \approx 7$. Based on this fact, we can deduce that the neuron in the center will be **more likely to have the same characteristic of the 3 classes**. That is, both 1 and 4 have a line!

However, why 4 is not in the center? The reason is, when we compare between 1 and 4, we find that **1 will have more black regions in the center than 4**, and **class 0 is black in this region**! Therefore, based on the fact that neuron in the center is more likely to be affacted by training samples, it is **reasonable** that Class 1 is more likely to be in the center region for the existence of Class 0 (will make the region tend to be black).

2. The same digits will be neighbous with each other.

Explanation: This is more understandable because, for the mechanism of SOM, while updating the weights, the winner neuron will be updated with its direct neighbours strongly. This will lead to the same prediction in Conceptual Map for those neighbours. Therefore, **it actually achieve the purpose of clustering**!

3.3.2 Classification accuracy

For this question, in 60 testing samples, only 31 of them are classified correctly. That is, **the accuracy is just around 50%.** We can check the weights with respect to the form of image:

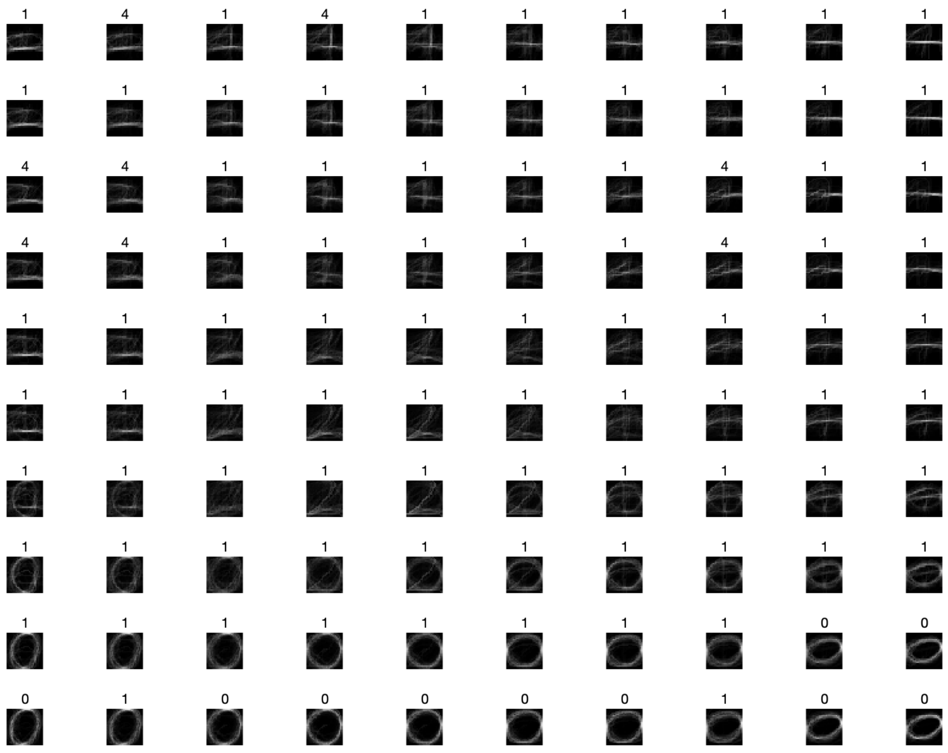

Figure 33: Weights

Findings: In Figure 33, all the images of weights are very **fuzzy**. The number ahead the image is its label determined by Conceptual Map previously. Actually, it can be observed that **for some images, its weight does not correspond to the label by our eyes**. That is because, this label is achieved by finding the winner over the training samples, which **may not give us the REAL label** of the weight in respect to image. **This is one big reason for the low accuracy in testing set.**

Actually, SOM is an **unsupervised learning algorithm**. Therefore, we do not expect high accuracy over the testing set since we do not have a teacher to give us the error signal. What we do is just, **updating the weights according to the inputs (no labels)**. Therefore, we can **assign those similar inputs to a similar region** in the N×N neurons. This is what we do in Conceptual Map, **a good way of clustering**, not Pattern Recognition. The result of clustering can be easily attained from the Conceptual Map.

4 Appendix

All matlab codes (*.m) will be attached in .zip.