Student name: Jin Hoontae (A0243155L)

Question 1

a) Use the exact interpolation method and determine the weights of the RBFN.

In this question, the exact interpolation method was employed to investigate how the RBFN would predict the desired output. **Fig. 1** shows three different graphs: Training sample, desired output and RBFN output. Mean Square Error (MSE) for the training set was $1.1266e^{-27}$, which was nearly 0. This value could be obtained due to the weights that were derived to fit all the training outputs. On the other hand, the MSE for the test set was 0.0987, which was not as close to 0 as the former one. This could be because the *Gaussian* noise was not considered in the test outputs, which caused a relatively high MSE value as compared to that of the training set.

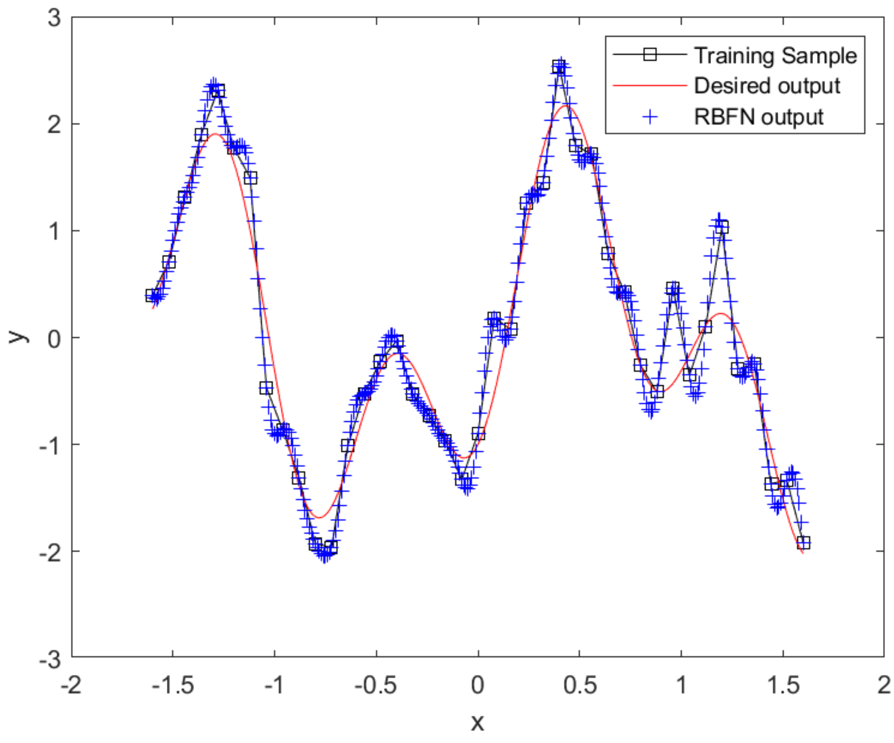


Figure 1. Regression result obtained by the exact interpolation method

Table 1. MSE Comparison between training set and test set

	Training set	Test set
MSE	$1.1266e^{-27}$	0.0987

b) Follow the strategy of "Fixed Centers Selected at Random" (as described on page 37 in the slides of lecture five), randomly select 20 centers among the sampling points.

In this part, 20 centers of RBFN were selected randomly from the training set instead of considering the whole sampling points as **Q1(a)**. **Fig. 2** shows three different graphs: Training sample, desired output and RBFN output with 20 centers. As for the training MSE, it increased to 0.0564, which was significantly higher than the previous one ($1.1266e^{-27}$) obtained in **Q1(a)**. This proves that the number of centers has a great influence on any training set; that is, the less centers are chosen, the lower the training MSE might become. However, the test MSE showed an improvement compared to the previous test MSE. The improvement can further be found in **Fig. 2** where the RBFN fitting curve became much smoother. Hence, to increase the performance accuracy of the test set, selecting the number of centers is in more favor than using the full training input data.

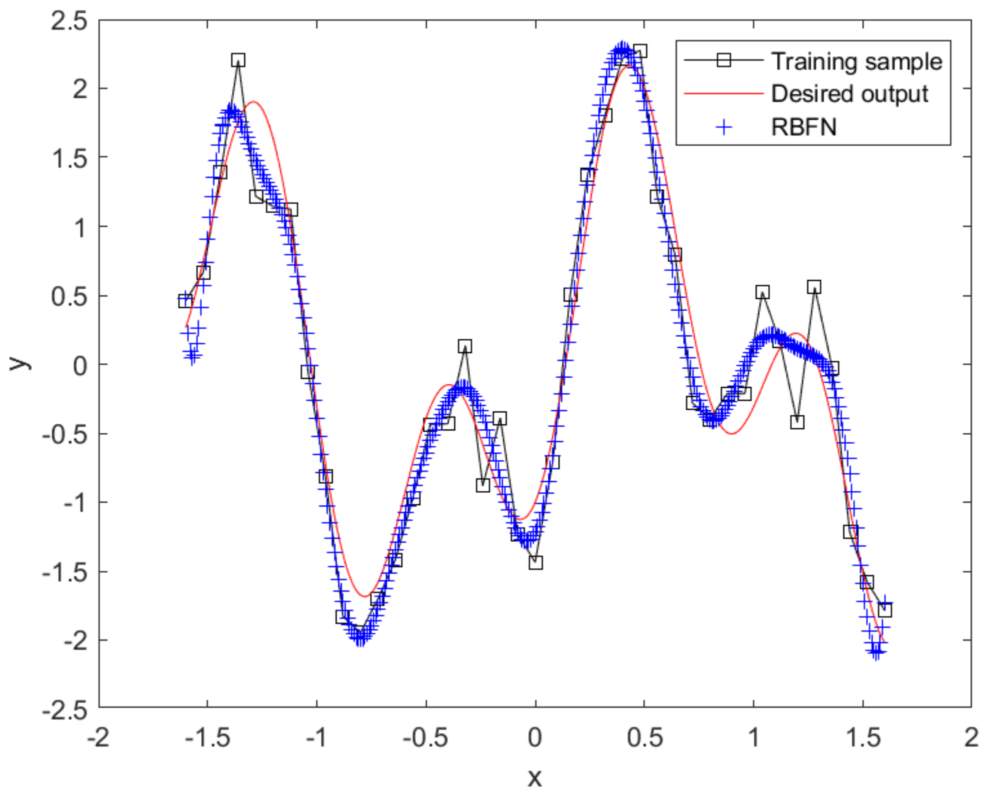


Figure 2. Regression result obtained with 20 centers of RBFN

Table 2. MSE Comparison between training set and test set with 20 centers

	Training set	Test set
MSE	0.0564	0.0551

c) Use the same centers and widths as those determined in part a) and apply the regularization method.

The term, *Regularization* (λ), was added to the function, and 6 different values were used to examine the behavior of the RBFN function with respect to a regularization value. As can be observed in **Fig. 3**, the fitting curve becomes smooth when the regularization value increases. However, when it is too high (in this case, when λ is 10), it completely loses the ability of fitting the desired output. Hence, a conclusion could be drawn from this finding that a proper value should be used for maximizing the fitting accuracy as well as the MSE result. From the observation of the figure, the best fitting could be obtained when λ was equal to 0.5.

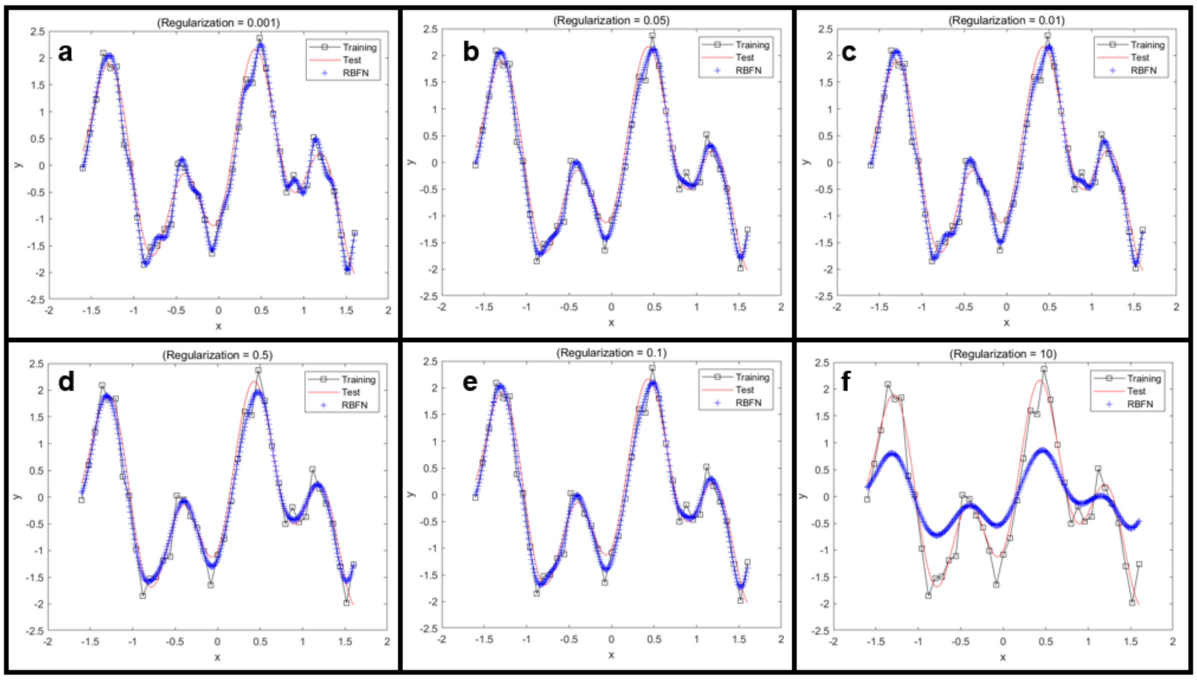


Figure 3. Regression result obtained with different regularization values

Table 3 shows the MSE values for the training and test sets for different regularization values. In the MATLAB script, the values would change for each run because of the noise created by the *randn* function. However, for every run, it was observed that the test MSE reached its lowest value when λ was 0.5, which correlated with the previous observation based on the plots.

Table 3. MSE Comparison between training set and test set with different regularization values

		Regularization (λ)					
		0.001	0.05	0.01	0.5	0.1	10
MSE	Training	0.0084	0.0170	0.0118	0.0360	0.0203	0.4681
	Test	0.0525	0.0405	0.0461	0.0374	0.0379	0.4518

Question 2

a) Use Exact Interpolation Method and apply regularization.

The exact interpolation method was applied to the given dataset without/with a regularization factor. **Fig. 4** demonstrates the effect of the regularization on the accuracy performance.

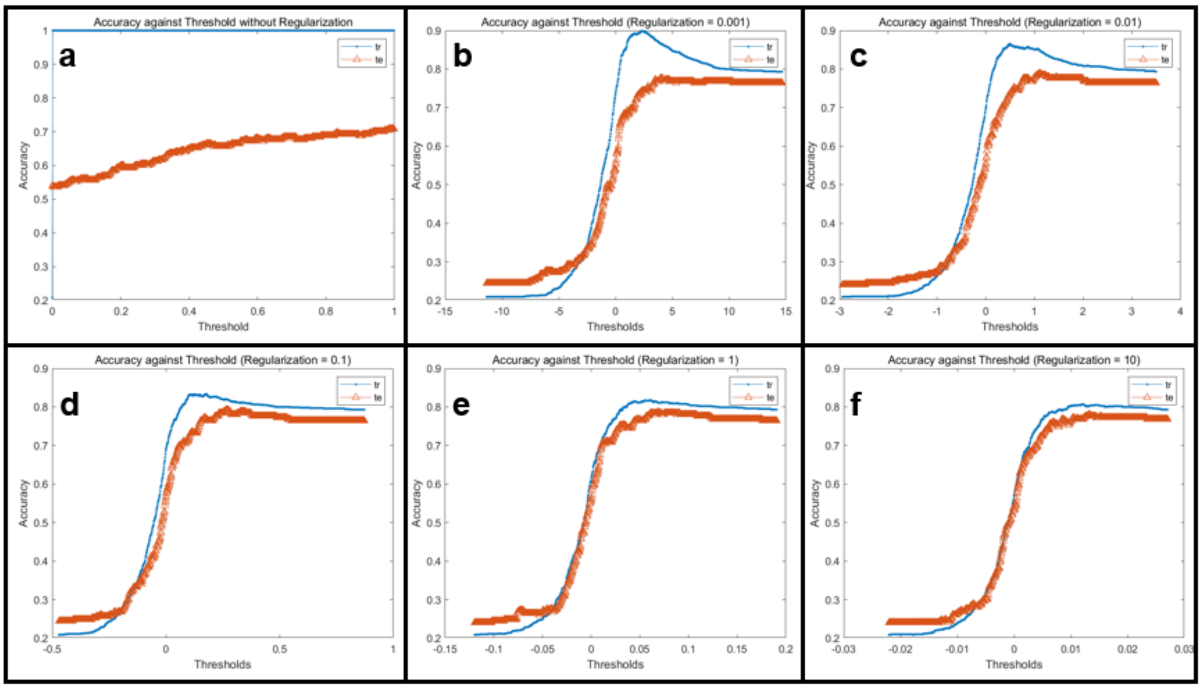


Figure 4. A comparison between the RBFN accuracy plots with/without the regularization

As can be observed from graph (a), the training accuracy remains at 100% throughout the whole thresholds from 0 to 1. From this result, a conclusion can be drawn that each individual training sample is passed when the exact interpolation method is used for RBFN. However, despite the training accuracy being 100%, it does not perform very well on the test set. This proves that the 100% accuracy is obtained due to the overfitting, which is not desirable.

When the regularization is applied, it greatly prevents RBFN from creating the overfitting phenomenon as can be seen in the other graphs (b-f), thereby keeping the accuracy balance between the training set and the test set stable. However, its value does affect the performance to a great extent. For example, when it is too low (=0.001), the training accuracy soars near 90% where the threshold is around 0~5. Moreover, when it is too large (≥0.1), the threshold margin becomes narrow.

b) Follow the strategy of "Fixed Centers Selected at Random" (as described in page 37 of lecture five). Randomly select 100 centers among the training samples. Firstly, determine the weights of RBFN with widths fixed at an appropriate size and compare its performance to the result of a) and then vary the value of width from 0.1 to 10000 and study its effect on the resulting RBFNs' performance.

When 100 training samples were randomly selected as the centers of the RBFN model with the fixed value of the width according to the equation ($\sigma = \frac{d_{max}}{\sqrt{2M}}$), not a significant improvement of the classification performance was not discovered because its accuracy remained around 70-80% similarly as compared to the ones in a), which can be seen in **Fig. 5(a)**.

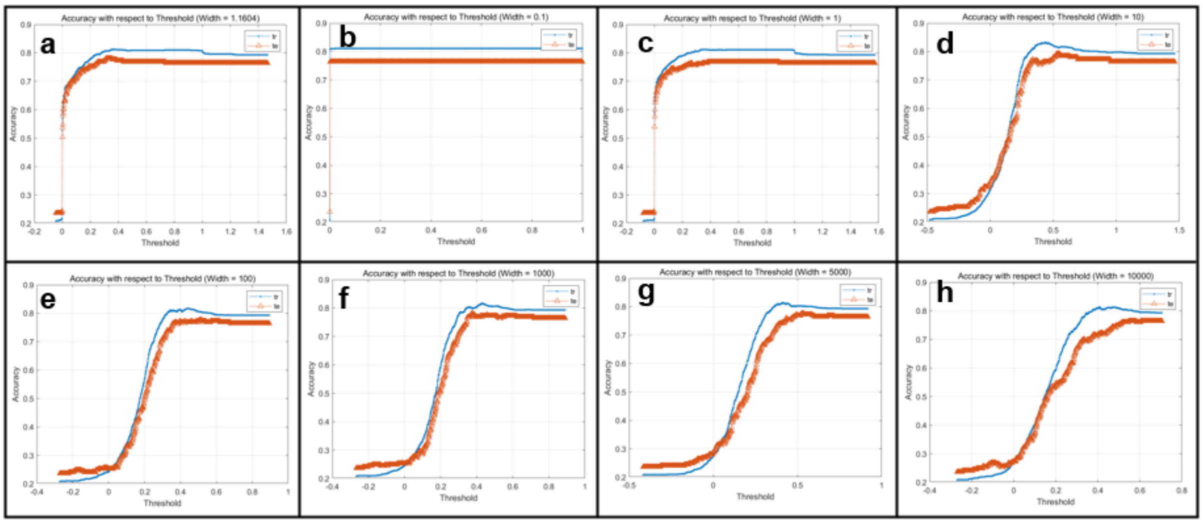


Figure 5. Comparison of the classification performance with respect to width values

The other graphs (**Fig. 5(b-h)**) illustrate how the classification accuracy differs depending on the value of width ranging from 0.1 to 10,000. From the graphs, it can be seen that the value of width does indeed affect the performance to some extent. To investigate more in detail, maximum accuracy values for training and test sets were extracted for each width graph, and a new graph was plotted as shown in **Fig. 6**.

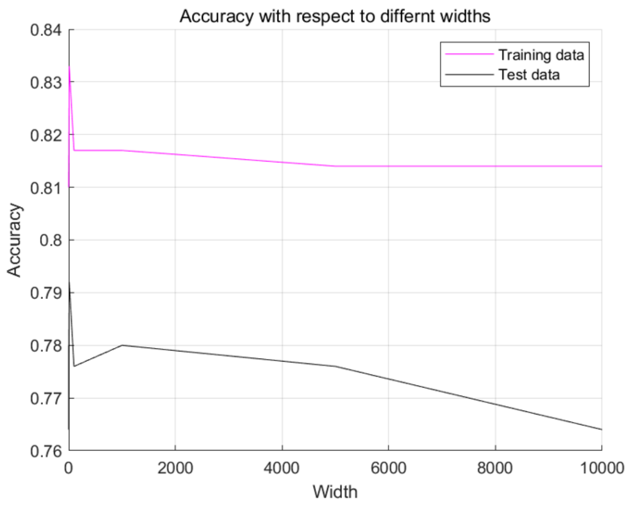


Figure 6. The performance of RBFN according to width values

The figure clearly shows that the deterioration of the performance occurs continuously as the value of width increases. Furthermore, the divergence of the accuracy between them exacerbates with the increasing with values, which implies that the overfitting corrupts the model's performance. Hence, as its value directly and significantly affect the classification accuracy, it is advisable to select a right one to maximize its performance.

c) Try classical "K-Mean Clustering" (as described in pages 38-39 of lecture five) with 2 centers. Firstly, determine the weights of RBFN and evaluate its performance; then visualize the obtained centers and compare them to the mean of training images of each class. State your findings.

In a similar manner to the previous question (**Q2(b)**), the classification performance of the K-Mean Clustering was evaluated according to different width values. Although the maximum accuracy does not decrease with the increasing width value, it does not display any improvement at all as shown in **Fig. 7**. Instead, the accuracy remains the same regardless of the width values for both the training set and the test set; that is, the model is not learning at all regardless of the width values.

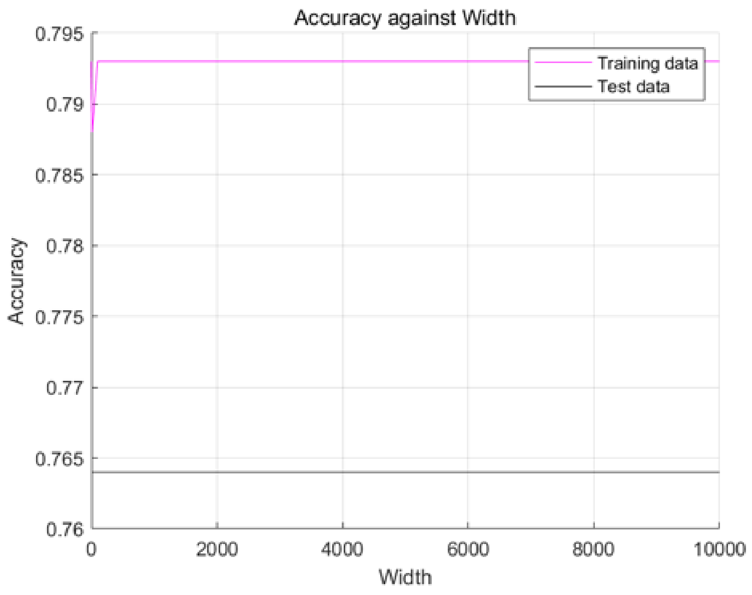


Figure 7. The performance of RBFN with 2 centers (K-Mean Clustering) with different width values

The poor performance can be observed by directly examining/comparing the images created by K-Mean Clustering with the mean of training images for each class. **Fig. 8** shows four images (two images created by K-Mean Clustering method, and the other two created by computing mean values of the training set for each class). As can be observed from the figure, the clustering algorithm failed to group two different labels. Both clustered images look like *Label 0*, even though one of them is supposed to look like *Label 1*. The cause of the poor performance, perhaps, is because of the quality of the images in the given training set, as the objects are too blurry and unclear. Thus, in order to improve the classification performance, collecting data with better quality could be a solution.

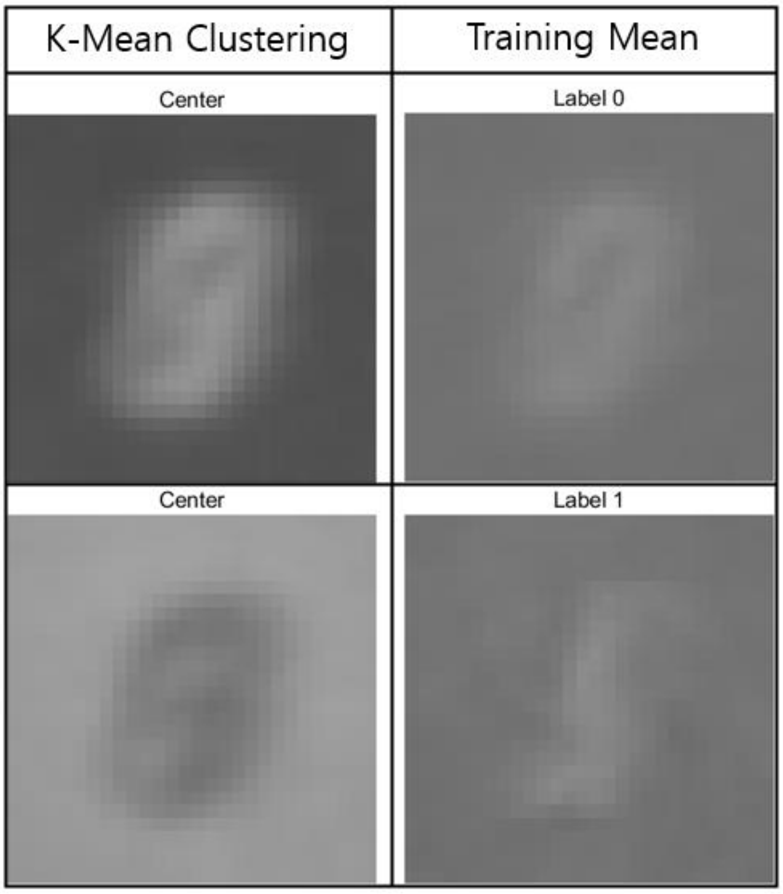


Figure 8. Comparison between K-Mean Clustering images and Training Mean images

Question 3 (SOM)

The following steps are taken for n-dimensional input space and m output neurons:

(1) Randomly initialize the weight vector w_i for neuron, i, i=1,…,m.

(2) Sampling: Choose an input vector x from the training set.

(3) Determine winner neuron k:

$$||w_k - x|| = min_k||w_k - x|| \text{ (Euclidean distance)}$$

(4) Update all weight vectors of all neurons i in the neighborhood of the winning neuron i(x): $w_j(n+1) = w_j(n) + \eta(n)h_{j,i(x)}(n)\left(x - w_j(n)\right), j = 1, \dots, m$

(5) If convergence criterion is met, Stop. Otherwise, go to (2).

The values of the variables such as the number of iterations, the width, the initial learning rate, etc. were given in the homework. One noticeable outcome that was observed for all the sub-questions was that, with respect to the increasing number of iterations, the performance accuracy increased.

a) Write your own code to implement a SOM that maps a 1-dimensional output layer of 40 neurons to a "hat" (sinc function).

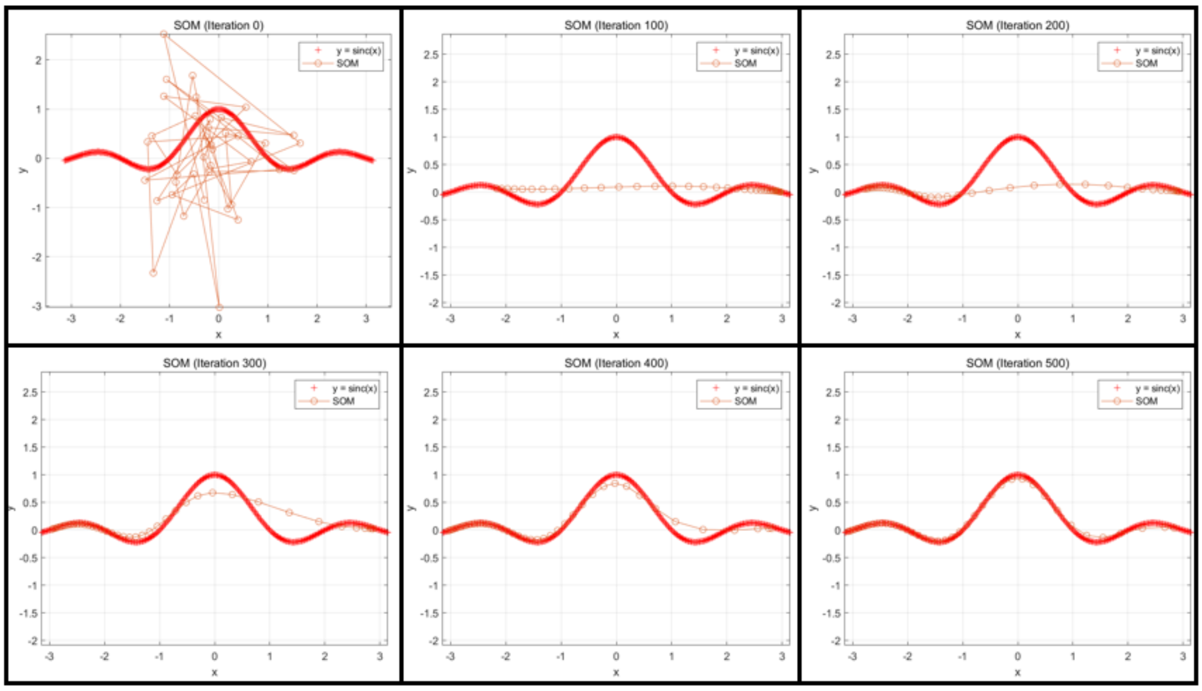


Figure 9. SOM for the sinc function

b) Write your own code to implement a SOM that maps a 2-dimensional output layer of 64 (i.e. 8×8) neurons to a "circle".

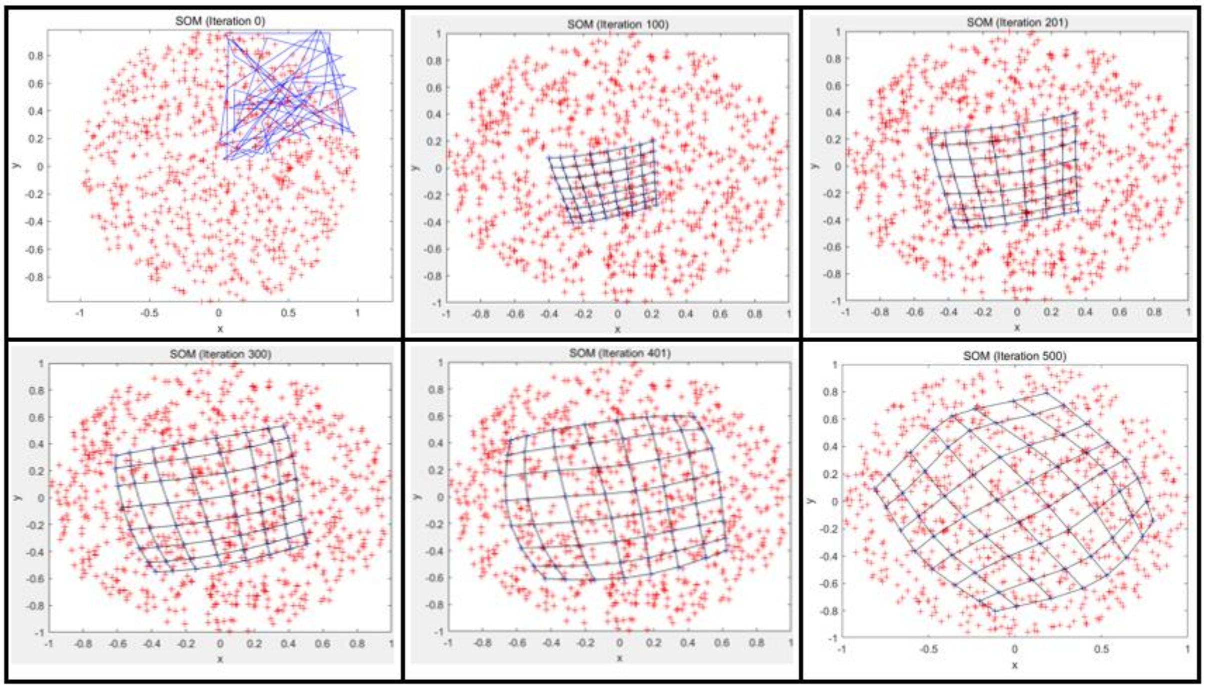
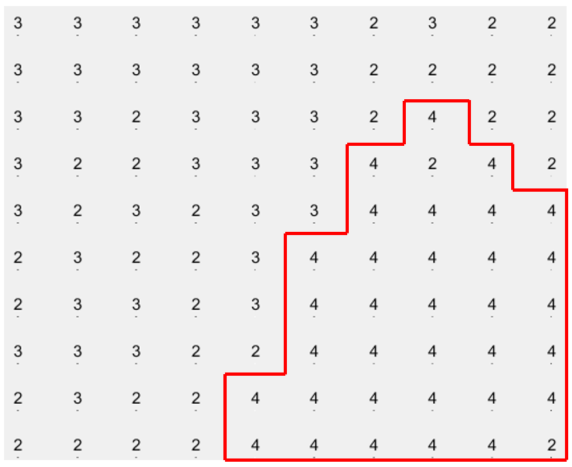


Figure 10. SOM for a 2-dimensional circle

c-1) Print out corresponding conceptual/semantic map of the trained SOM (as described in page 24 of lecture six) and visualize the trained weights of each output neuron on a 10×10 map.

As shown in **Fig. 11**, the labels (2, 3 and 4) tend to be grouped with their own numbers, although it's not 100% accurate. A group of 4's was the most well-grouped one among the others with only two different labels inside. The other numbers (2 and 3) failed to group up successfully. The reason for this erroneous result could be because their curvy shapes share some similarity in the given images.



3	3	3	3	3	3	2	3	2	2
3	3	3	3	3	3	2	2	2	2
3	3	2	3	3	3	2	4	2	2
3	2	2	3	3	3	4	2	4	2
3	2	3	2	3	3	4	4	4	4
2	3	2	2	3	4	4	4	4	4
2	3	3	2	3	4	4	4	4	4
3	3	3	2	2	4	4	4	4	4
2	3	2	2	4	4	4	4	4	4
2	2	2	2	4	4	4	4	4	2

Figure 11. Labels created by SOM

The erroneous result can be discovered more in detail in **Fig. 12**. It shows that the images labeled as “2” oftentimes fail to look how they are supposed to look; that is, some look like “3” or “4”. A similar result can be seen for the other numbers as well.

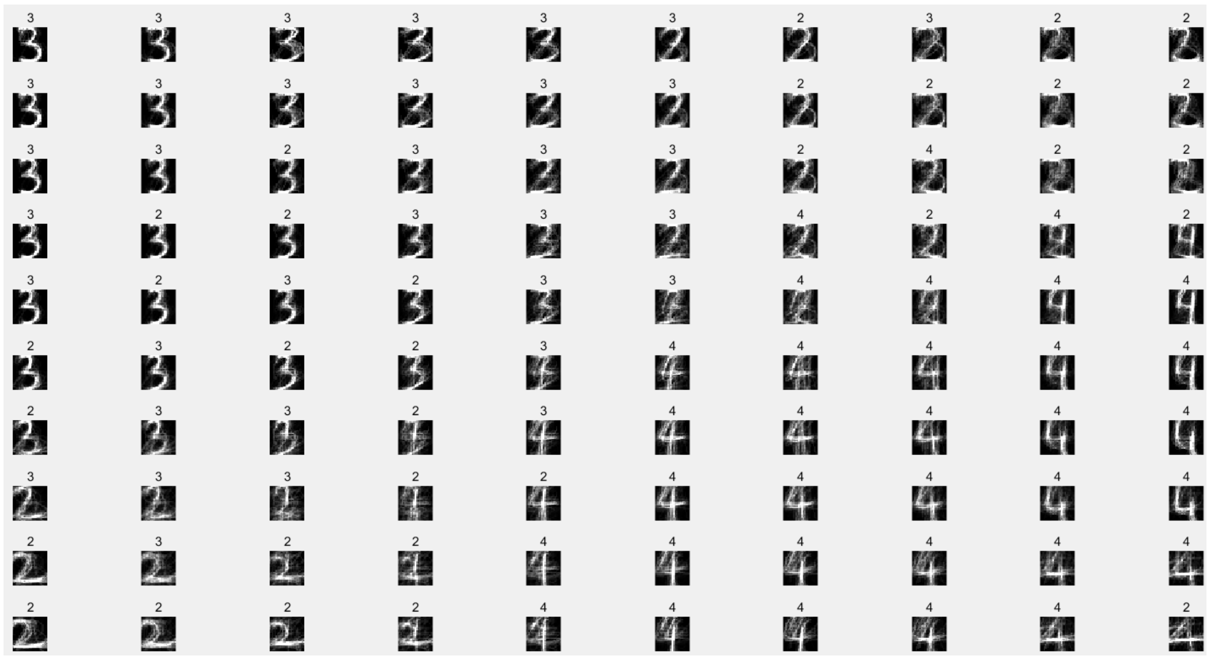


Figure 12. Labels and images by SOM

c-2) Apply the trained SOM to classify the test images (in test_data). The classification can be done in the following fashion: input a test image to SOM, and find out the winner neuron; then label the test image with the winner neuron’s label.

The trained SOM was applied to classify the test images as shown in **Fig. 13**. The ones in red boxes are the incorrectly classified images. The classification accuracy for the test set was 73.3% in this case. **Table 4** illustrates the classification accuracy for each image. The results show that the images (4) were labeled with the highest accuracy (84.2%). As previously mentioned, due to the curvy shapes that the images (2 and 3) share in common, they showed relatively low classification accuracy (70% and 66.7%, respectively). However, as the SOM is built with many parameters such as the lattice size, iteration number, learning rate, etc., it is expected that the improvement can be made by tuning the parameters.

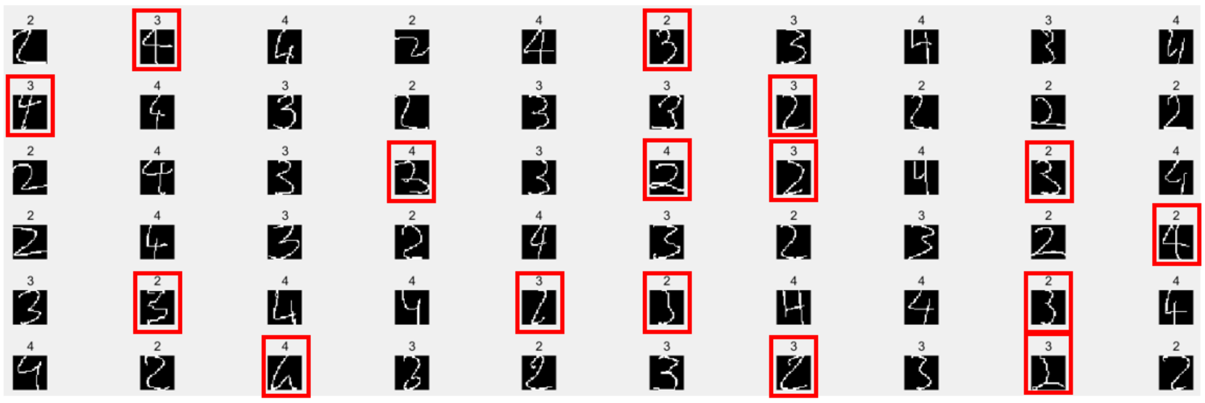


Figure 13. Test labels and images by the trained SOM

Table 4. Classification accuracy for each test image

	Label		
	2	3	4
Total	20	21	19
Error	6	7	3
Accuracy	70%	66.7%	84.2%