Chua HongWei (A0074741E)
Homework 3

Q1)

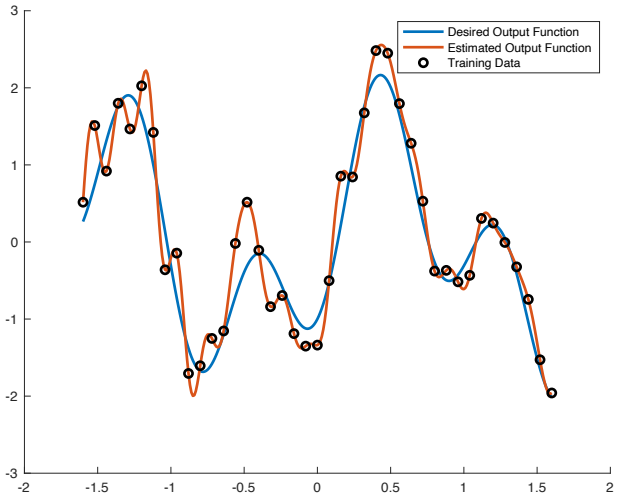
In this question, RBFN is used to approximate the function

$$y = 1.2sin(\pi x) - cos(2.4\pi x)$$

With x ranging from [-.16,1.6]. For training set, the step length is set at 0.08 while for testing set, the step length is set at 0.01. A noise is also introduced for training set observed outputs. Three variations of RBFN are experimented and results are documented below.

Q1a)

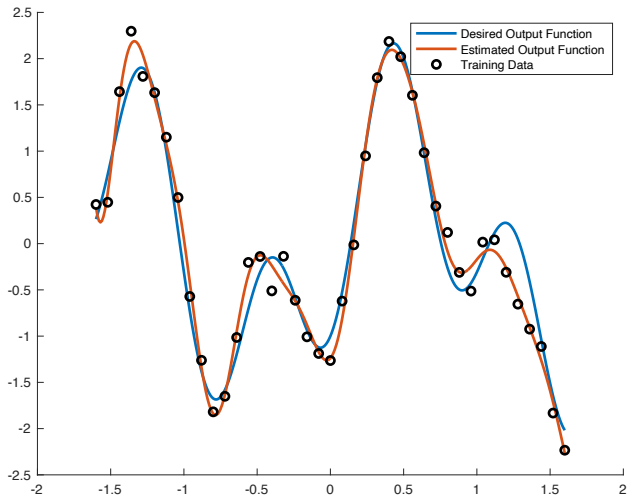
The weight vector derived from the training set is a 41x1 matrix. It could be obtained by running Matlab code Q1a.m. The plot below shows the approximated function from the testing set with the weight matrix obtained above.



It is shown that approximated function is not a good fit to the actual function. In fact, the predicted function is overfitting the testing data as the exact interpolation method tries to fit all the data. The mean square error between the estimated function and the desired function is 0.11.

Q1b)

The weight vector derived from the training set is a 20x1 matrix. It could be obtained by running Matlab code Q1b.m. The plot below shows the approximated function from the testing set with the weight matrix obtained above.

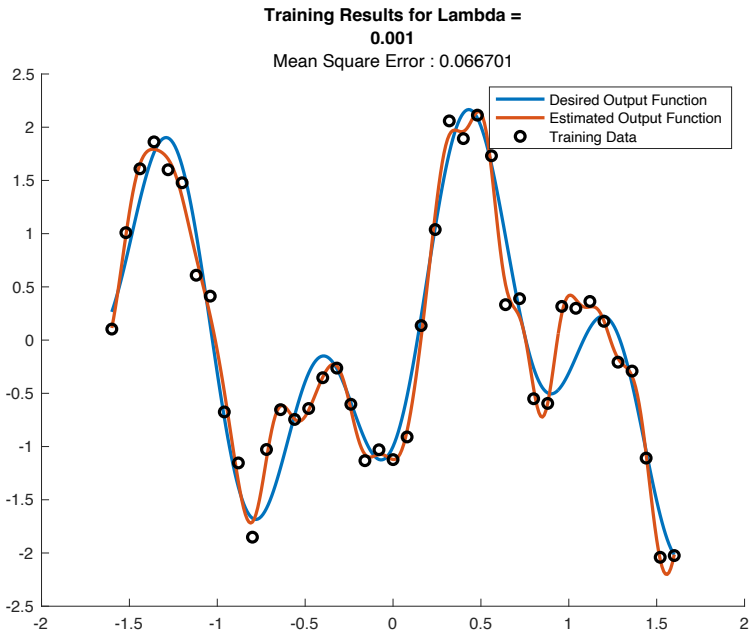


It is observed that the estimated output function matches more closely to the desired output function when compared to Q1a. The mean square error has also reduced to 0.05. This shows that the strategy of "Fixed Centers Selected at Random" improved the performance of RBFN in function approximation.

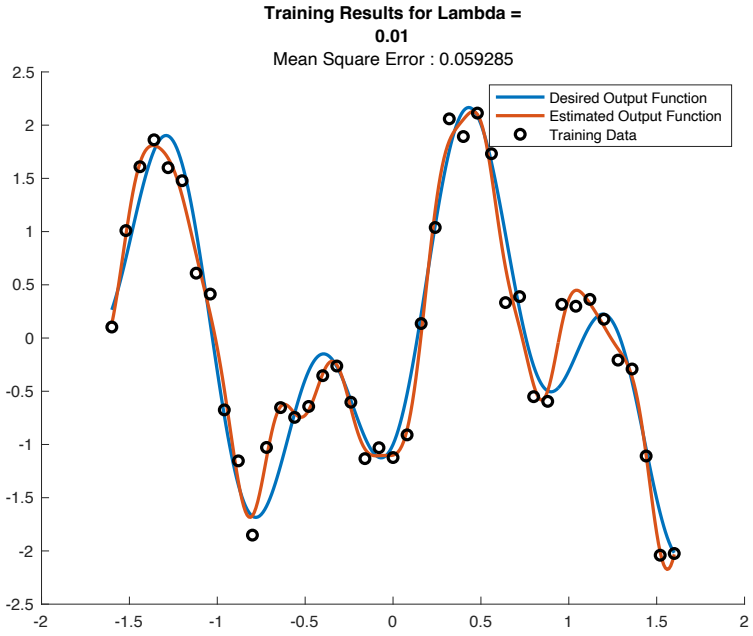
Q1c)

In this question, the center and width used is the same as Q1a but with regularization term added. The regularization factors used to demonstrate the effect of regularization is 0.001, 0.01, 0.1, 1, 2 & 10. The results are listed in the graph below with the mean square error indicated under the title. For fair comparison of each regularization factor, a random seed of 42 is set for the random number generator.

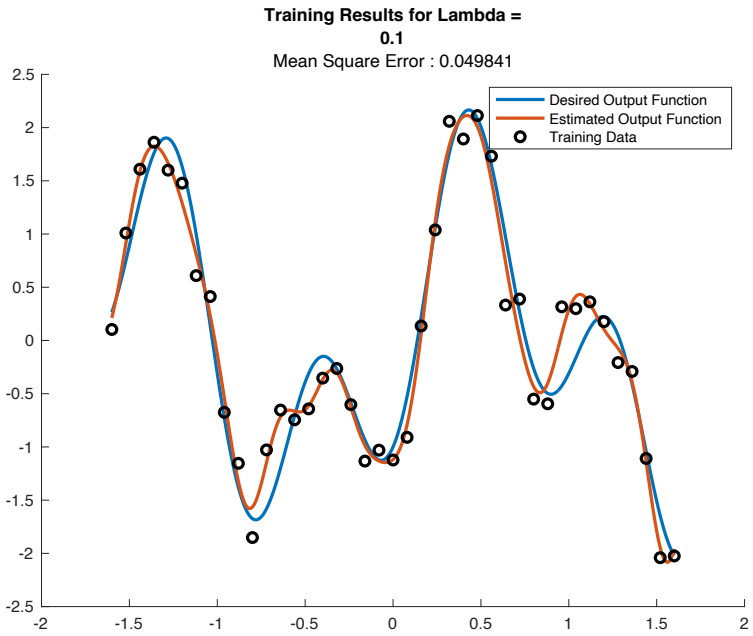
Regularization factor = 0.001



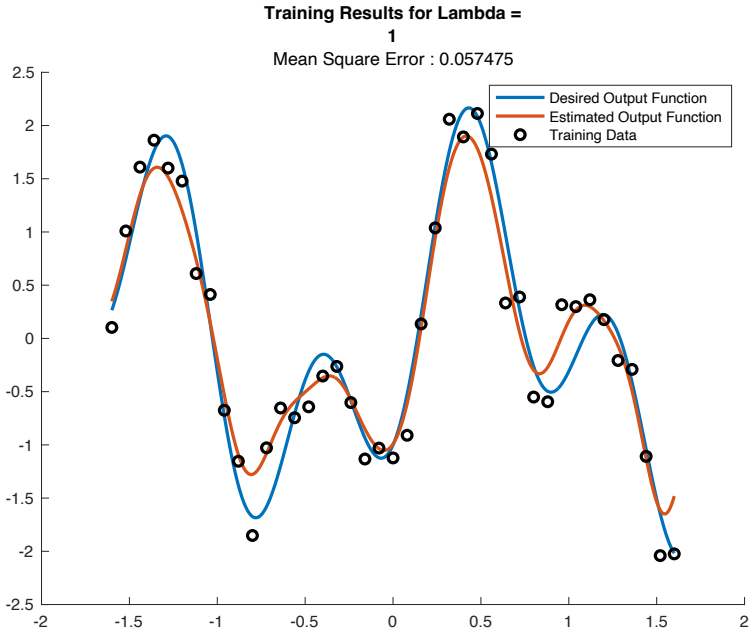
Regularization factor = 0.01



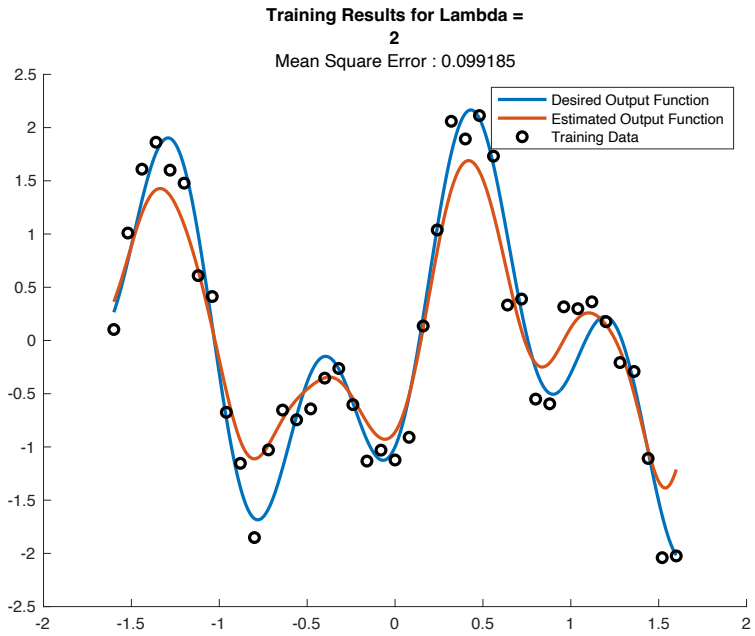
Regularization factor = 0.1



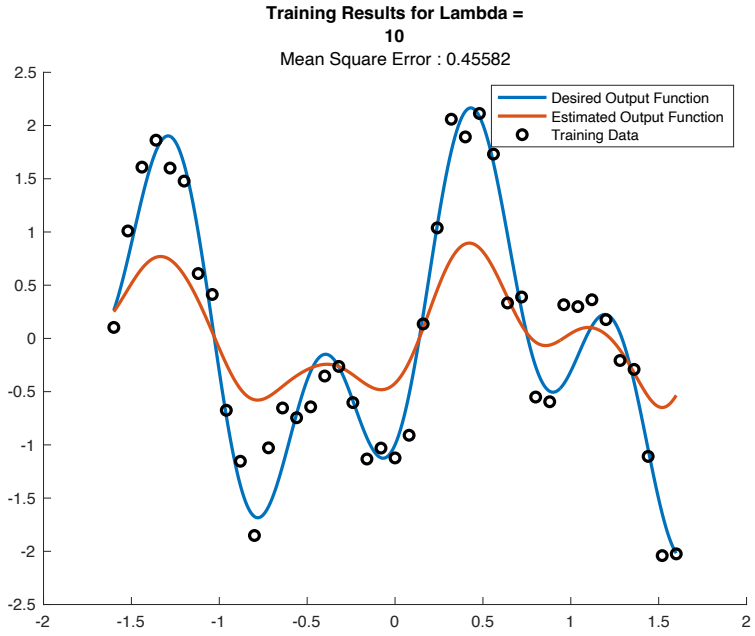
Regularization factor = 1



Regularization factor = 2



Regularization factor = 10



From the results above, it is observed that for small regularization factor from 0.001 to 0.01, the predicted output function produced is overfitting to the training data. When the regularization factor starts to increase in magnitude, the predicted function produced a proper fit to the training data, as observed from graph with regularization factor of 0.1 and 1. However, when the regularization factor increases to 2 and above, the predicted function will underfit the training data.

For this question, the best regularization factor is 0.1, with the MSE being the lowest at 0.049841.

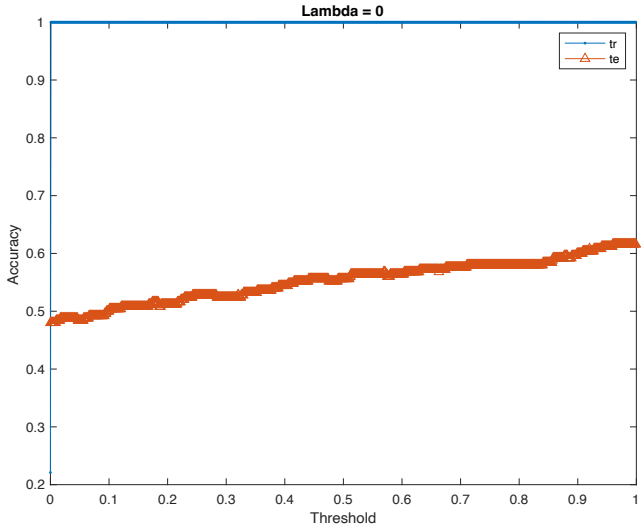
Q2)

The two classes that is chosen based on two different digits of my matric number is **classes 1 and 4**.

A function named loadmnist.m is created to load the mnist_m.mat data across all question and to re-label image of class 1 & 4 to '1' and the other image classes to '0' per requirement from the question.

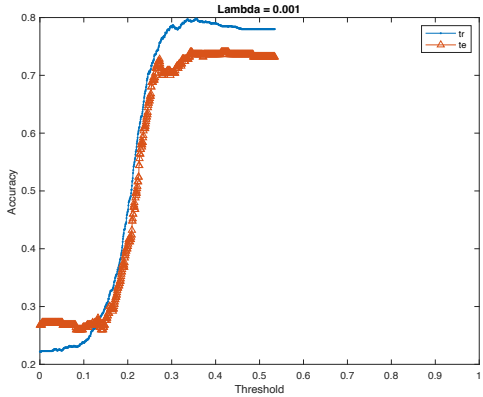
Q2a)

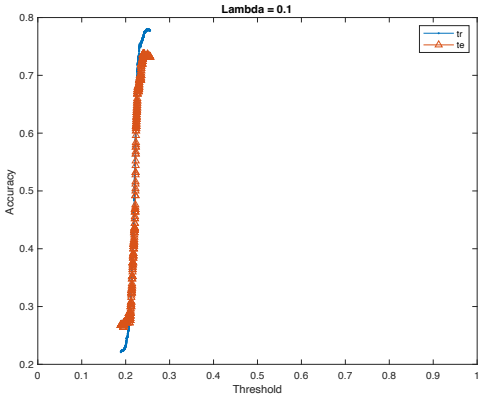
Question 2a aim to solve the classification problem via exact interpolation method. The method is first run without regularization and further done with regularization. The graph below shows the plot for exact interpolation method without regularization.

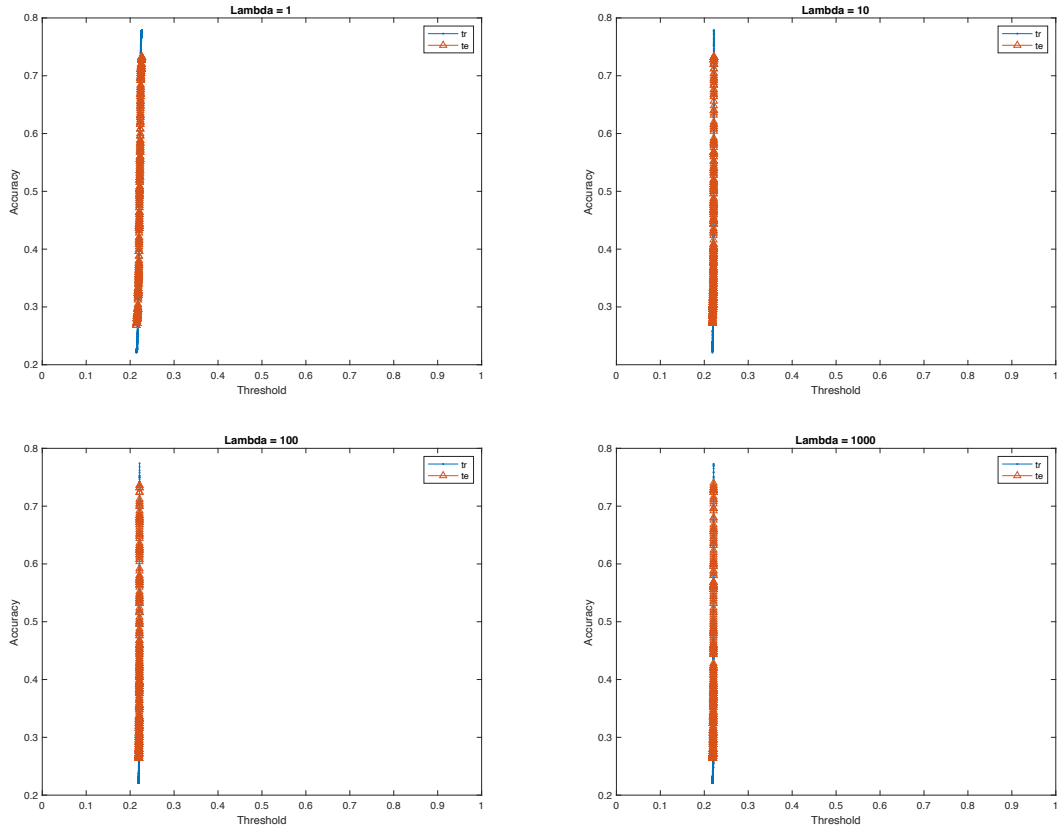


The training set obtained an accuracy of 1 due to exact interpolation method. Similar weights are used for testing set and the accuracy obtained range from 0.48 to 0.6 with increasing threshold.

Similar experiment was done but with extra regularization term added. The regularization factor used are [0.001 0.1 1 10 100 1000] respectively. The resulted plots are listed below.



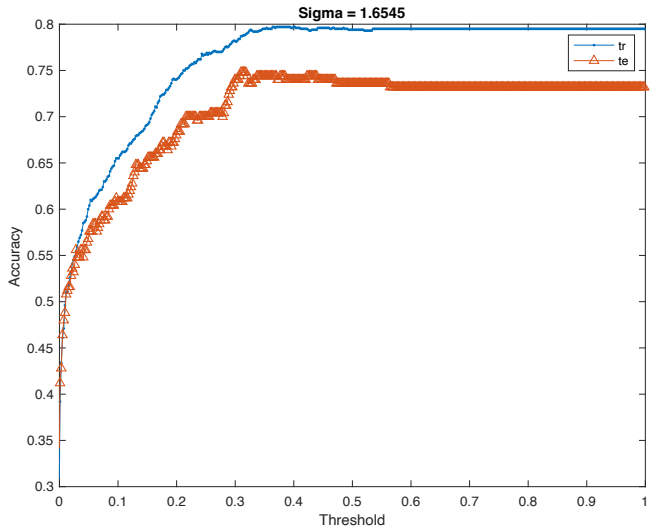




From the plots listed above, it is observed that a low regularization factor improves the accuracy of the testing set. When the regularization factor is at 0.001, the testing accuracy increases to above 0.7 within the threshold range 0.25 to 0.55. However, as the regularization factor increases, the performance of this RBFN for this classification task decreases drastically. The effective threshold range become narrow and it's around 0.23 to achieve 0.75 classification accuracy for the testing set.

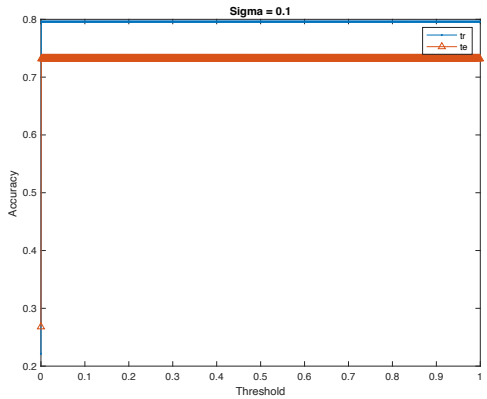
Q2b)

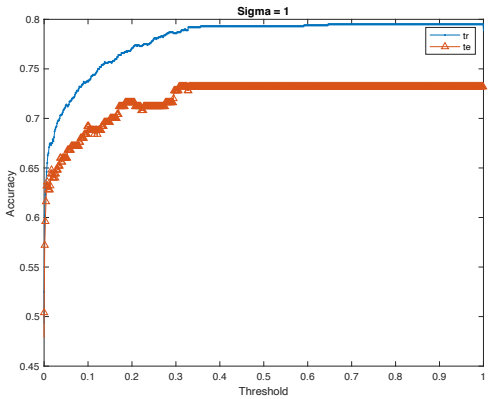
The question uses the 'Fixed center selected at random' method with varying sigma value. 100 centers are randomly selected. For consistent results, a random seed of 42 is chosen. The sigma calculated through the random selection of 100 centers is 1.6545 for my case. The plot below shows the performance of this method with the sigma calculated above.

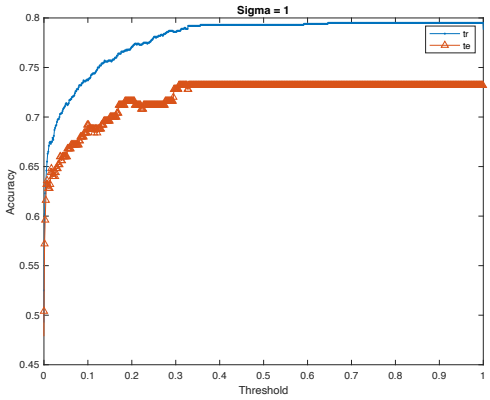


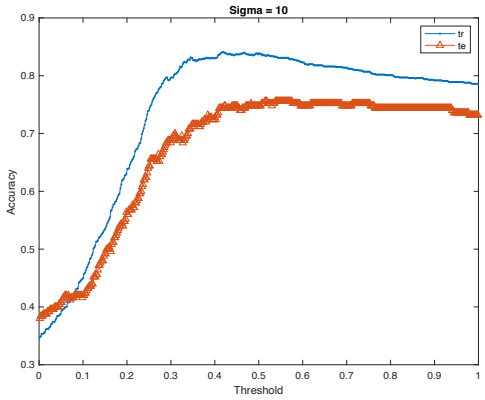
The performance of the RBFN with this method is better than Q2a with the testing accuracy improved to 0.75. The threshold range has also widened.

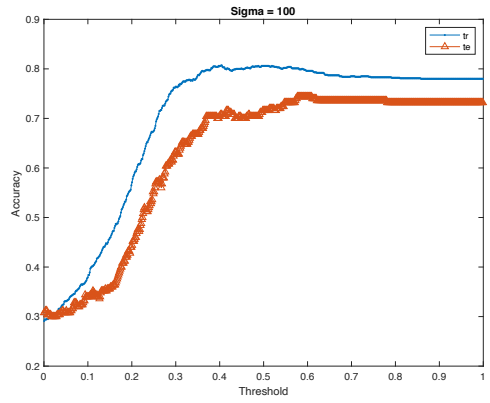
Next, an experiment with varying sigma value of [0.1 1 1 10 100 1000 10000] is perform on similar algorithm. The resulted plots are listed below.

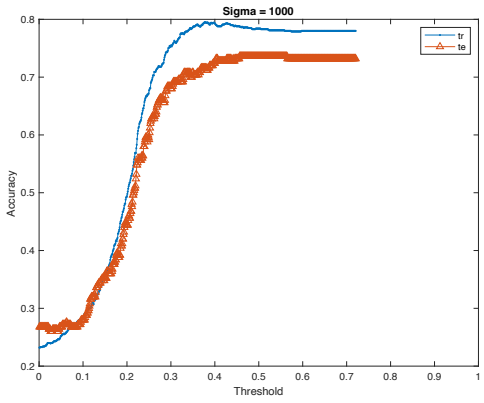


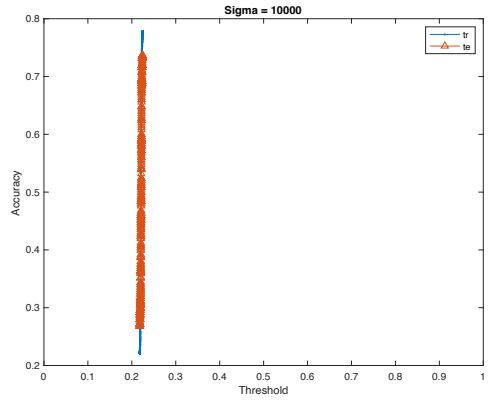










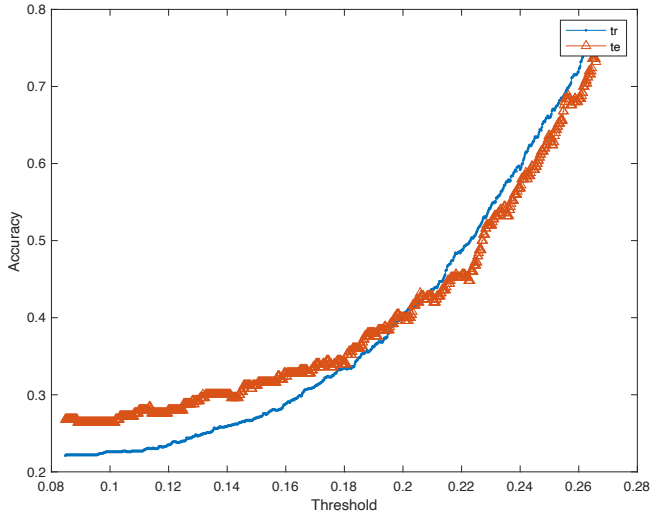


It is shown that for sigma less than 1, the accuracy didn't change for every increasing threshold. Further investigation shows that for low sigma, the algorithm keeps predicting class '0' for all the test data as the predicted value are very small.

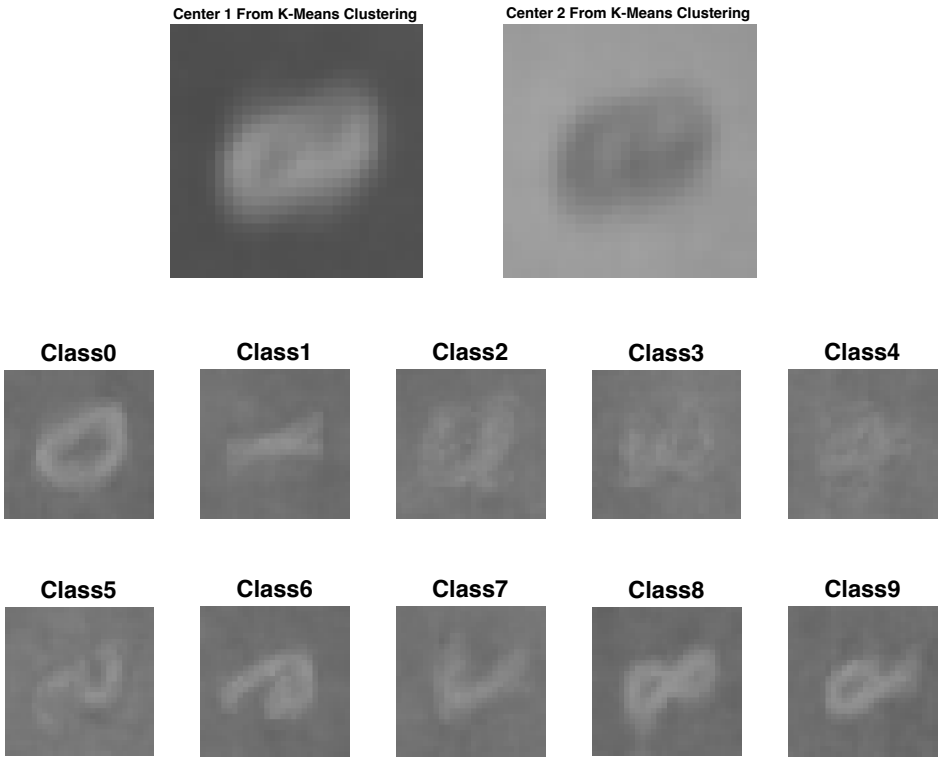
When sigma increases to above 1, the algorithm is shown to have good prediction accuracy of around 70% for testing set. However, when the sigma is huge at 10000, the effective threshold range become narrow and it's around 0.23 to achieve 0.75 classification accuracy for the testing set, displaying similar result with Q2a when the regularization factor keeps increasing.

Q2c)

For this question, K-means clustering with 2 centers was first used to detect the 2 centers from all the training data. Then RBFN is used to for the classification task. The performance of this method can be view in the plot below.



The testing accuracy is around 0.732 with a threshold of around 0.26. The performance of this method is comparable to the method used in Q2b. The figure below shows the image obtained from the 2 centers after 15 iterations with the K-means clustering method and its comparison with the mean of each class of the training data.

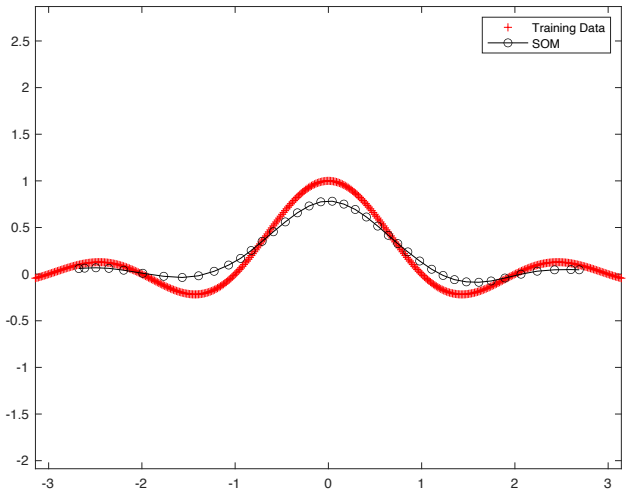


The 2 centers form a shape of more like ‘0’ or ‘8’. This is probably due to a similarity of most training data to the shape ‘0’ and ‘8’, as seen from the mean of each class. In order to improve the clustering performance, one could use a sharper image with more pixel count for the algorithm to make a clear distinction between classes.

Q3)

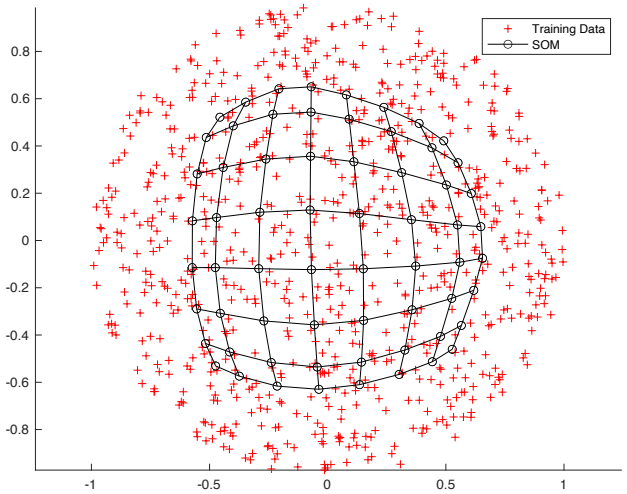
Q3a)

The function approximation with SOM can be seen in the plot below.



The SOM can map the function well with close approximation to the training data.

Q3b)

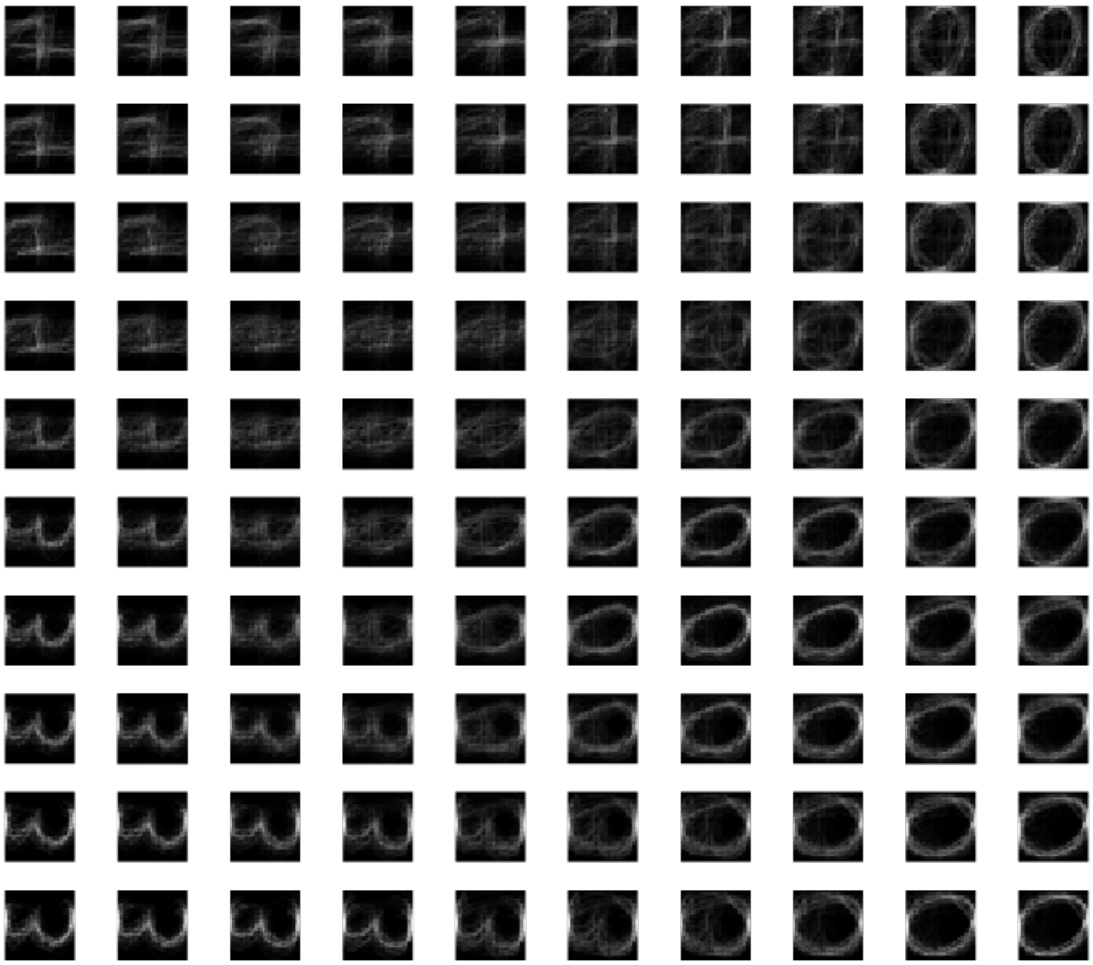


The SOM can map the circle function well after around 500 iterations. More iterations is required for the mapping to be more accurate.

Q3c1)

For this question, based on my matric card number, the classes that need to be omitted is class ‘1’ and ‘2’, which leave me three classes **‘0’, ‘3’, & ‘4’** for this classification task.

The figure below shows a semantic map of the trained SOM weights for this question.



From the semantic map, I observed that handwritten digits with similar orientation are group together. Several groups of the same handwritten pattern were observed from the semantic map.

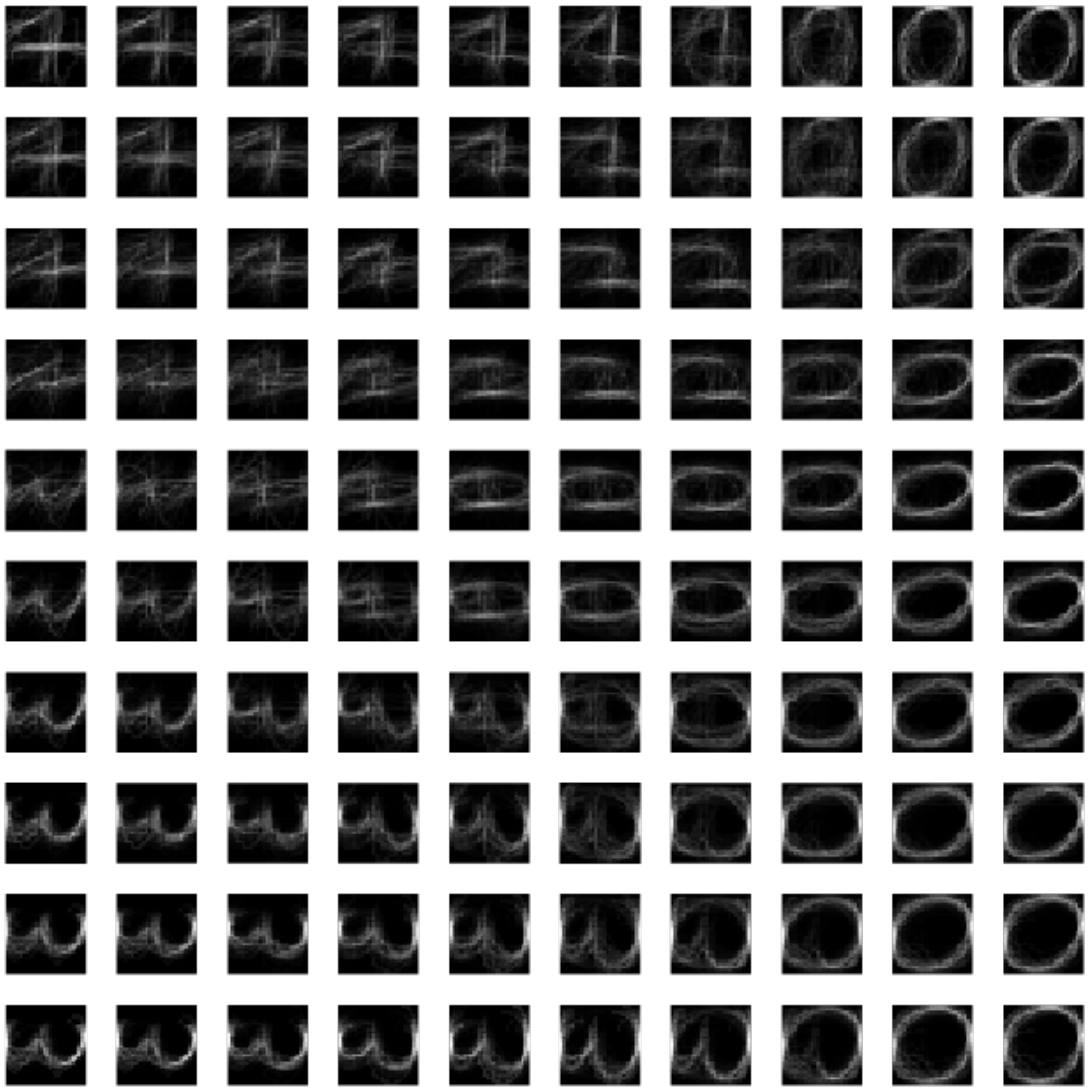
Q3c2)

The training and testing accuracy were both calculated in this question with the trained SOM. The results are as follows.

Training Accuracy: 81.5%
Testing set: 70%

The SOM did a pretty good job in the testing set with a high win rate.

I further tested the performance of the SOM with 10,000 iterations. The semantic obtained is shown below.



The semantic map doesn’t show significant changes, but the training and testing accuracy increases to the 83.2% and 76.7% respectively, proving that by giving more iterations to the SOM will produce a more accurate classification result.