EE5904 Neural Networks: Homework 3

Jin Lexuan A0232696W

March 11, 2022

Q1

a)

In this sub-question, exact interpolation method is implemented. Predict curve has similar tendency with test set. It shows in Figure 1 that the predict result pass through every data from training set exactly, which means there is an over-fitting problem. Since the training data is affected by noise with variance of 0.1, the predict result is also affected the predict result. In conclusion, exact interpolation method can generate approximation, but is easily influenced by noise. The mean-square error of predict curve and test set is 0.0810.

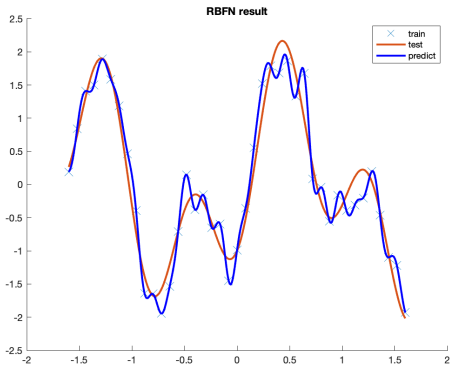

Figure 1: RBFN(exact interpolation method)

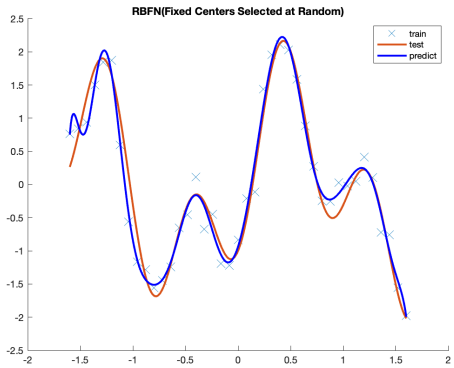

Figure 2: RBFN(Fixed Centers Selected at Random)

b)

"Fixed Centers Selected at Random" method is applied in this question. It means that not all the data in the training set is selected for training, but random 20 is chosen. From the comparison of Figure 1 and Figure 2, it show that the predict curve in Figure 2 is more smooth and has a higher compatibility with test set. Since method 2 does not consider all the data in train set, it solves over-fitting problem to a certain degree. The mean-square error of predict curve and test set is 0.0258, which is much smaller than that of exact interpolation method.

c)

Regularization is another method to solve over-fitting problem. Figure 3 to Figure 8 shows effect of different regularization factor. it shows that even small factor have notable effect to make the predict curve smooth. When the regularization increases,the larger the slope is. It controls the balance between s smooth mapping and fitting the data points exactly. In this case, parameter smaller than 1 generates good performance. When parameter is larger than 1. The prediction is too smooth to fit the test set.

Table 1 shows the performance of different RBFN method quantitatively. Method with smaller mse value has a better performance.

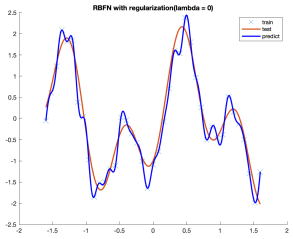

Figure 3: RBFN(re=0)

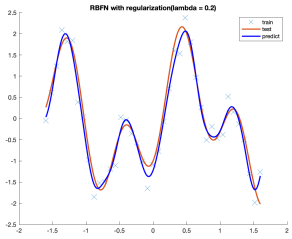

Figure 4: RBFN(re=0.2)

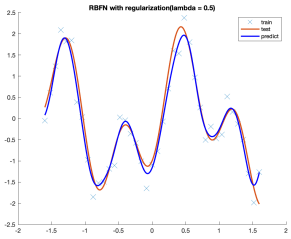

Figure 5: RBFN(re=0.5)

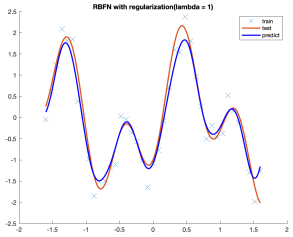

Figure 6: RBFN(re=1)

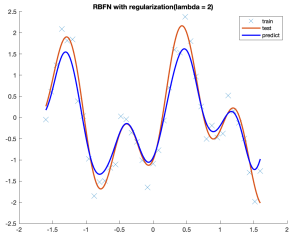

Figure 7: RBFN(re=2)

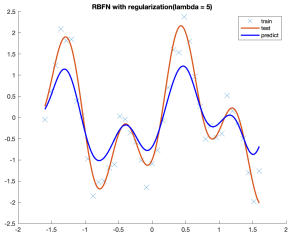

Figure 8: RBFN(re=5)

method	exact	random	re=0	re=0.2	re=0.5	re=1	re=2	re=5
mse	0.0810	0.0258	0.0857	0.0332	0.0330	0.0430	0.0815	0.2290

Table 1: Performance of different RBFN method

Q2

My matric number is A0232696W, so according to the requirement, classes 9 and 6 are labeled as 1 and other classes are labeled as 0.

a

Figure 9 shows the exact interpolation method without regularization. Although the performance of train set accuracy is 100%, the accuracy of test set is just about 50%, which means over-fitting happens.
Figure 10 to Figure 15 shows effect of different regularization parameter from 0.001 to 100. The involvement of regularization parameter solve the over-fitting problem, for different value of threshold, train set and test set have almost same accuracy. With regularization increasing, for different threshold, it has a more smooth slope to grow, and finally reach accuracy of 80%. Generally, parameter around 1 is suitable.

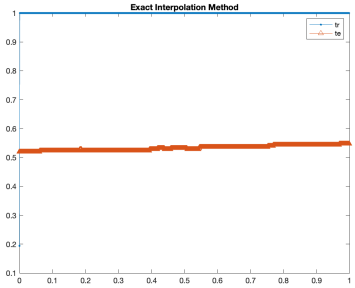

Figure 9: Exact interpolation method

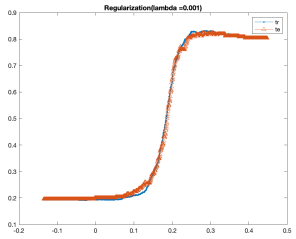

Figure 10: Re factor(0.001)

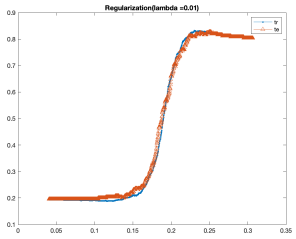

Figure 11: Re factor(0.01)

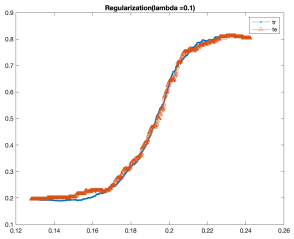

Figure 12: Re factor(0.1)

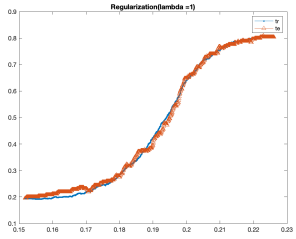

Figure 13: Re factor(1)

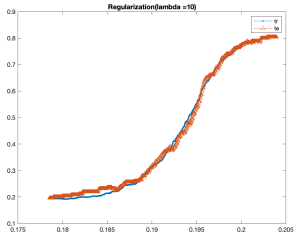

Figure 14: Re factor(10)

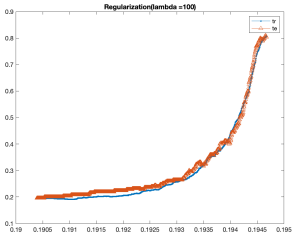

Figure 15: Re factor(100)

b

In this part, strategy of "Fixed Centers Selected at Random" is applied. For Figure 16, the appropriate size is defined by the maximum distance from 100 selected centers. Compared with exact interpolation method without regularization in question a, it has a better performance that can reach accuracy of 80%.
From Figure 17 to Figure 22(next page), Different value of width is applied. For small width which is under 100, since centers are very closed to each other, so data are easily classified as same class. In this question's data set, data is divide into 2 classes and the percentage of them is 20% and 80%, so it shows that for every threshold, the accuracy keeps at 80%. When the width is large, centers are separate. For different values of threshold, accuracy has different performance, but all finally reach 80%.

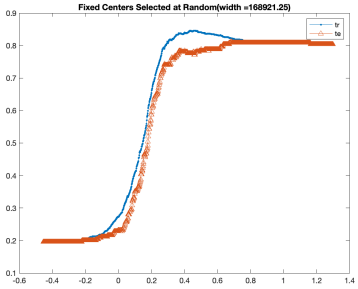

Figure 16: appropriate size

c

"K-Mean Clustering" is used in this question. Iteration stops at when all train set data stays in the same group compared with last iteration. With experiment, 20 times of iteration is enough.
When obtain 2 centers of 2 classes, then RBFN weights can be calculated. Performance is shown in Figure 23. It can achieve accuracy of 80%.

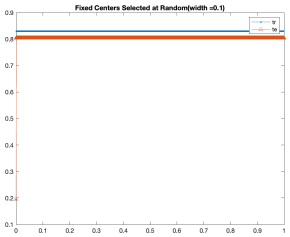

Figure 17: Width(0.1)

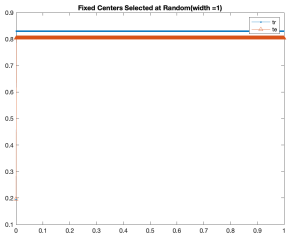

Figure 18: Width(1)

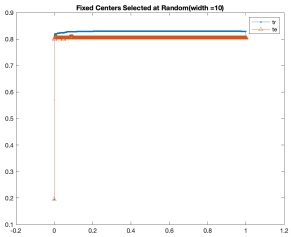

Figure 19: Width(10)

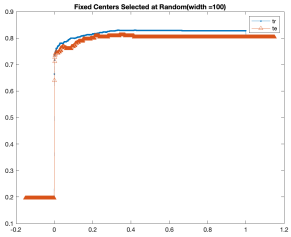

Figure 20: Width(100)

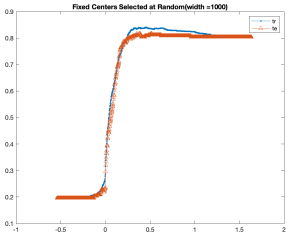

Figure 21: Width(1000)

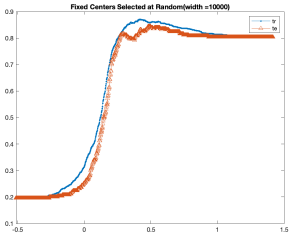

Figure 22: Width(10000)

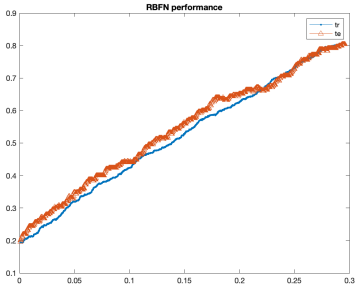

Figure 23: RBFN performance

Figure 24 and Figure 25 shows the visualization of clustering centers and mean of training images of each class respectively. It shows that two clustering centers has a more obvious contract compared with mean of training image. Two visualization result is quite different. For clustering, there are 434 images for one class and 566 images for another. For train set, there are 194 images for one class and 806 images for another. Since K-Mean clustering is strictly base on distance, its has a better effect.

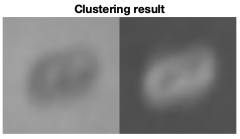

Figure 24: Clustering result

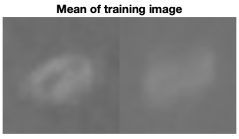

Figure 25: Mean of training image

Q3

a

Figure 26 to Figure 34 shows the self-organizing process within 50 iteration. Figure 26 presents the initial random weights. During the organizing process, weights become gathering first, then become a line and finally fit in the "hat". When these 1-dimensional output layer linking together, they just describe the sinc function.

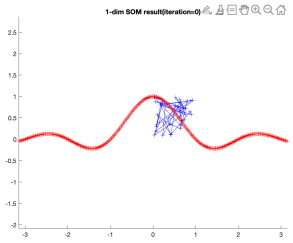

Figure 26: 1-dim SOM(0)

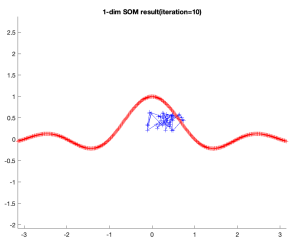

Figure 27: 1-dim SOM(10)

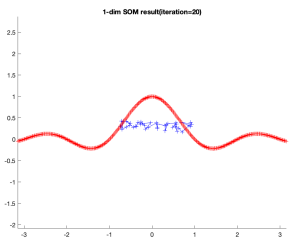

Figure 28: 1-dim SOM(20)

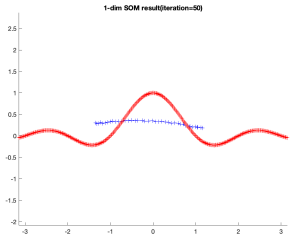

Figure 29: 1-dim SOM(50)

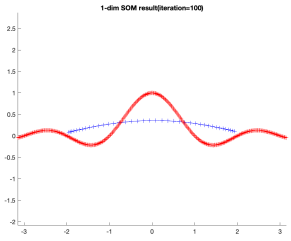

Figure 30: 1-dim SOM(100)

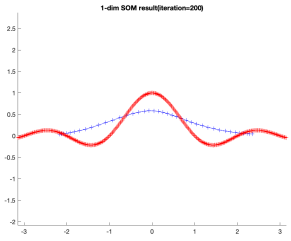

Figure 31: 1-dim SOM(200)

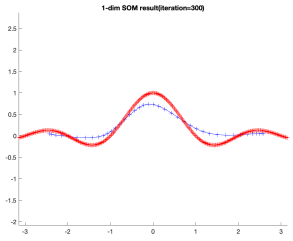

Figure 32: 1-dim SOM(300)

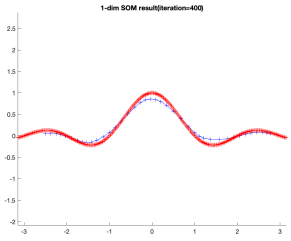

Figure 33: 1-dim SOM(400)

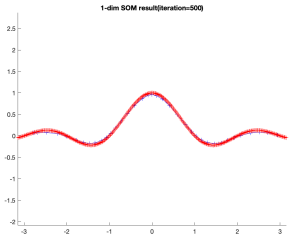

Figure 34: 1-dim SOM(500)

b

Figure 35 to Figure 43(next page) shows the self-organizing process within 50 iteration. Figure 35 presents the initial random weights. During the organizing process, weights become gathering first at the center of the circle, then become a line. Then the line extends to a "square block", gradually the block grows and has a tendency to cover the "circle". When these 2-dimensional output layer linking together, they generate a network to describe the "circle".

c

My matric number is A0232696W, so according to the requirement, classes 1 and 2 are omitted. Then do SOM for class 0, class 3 and class 4.

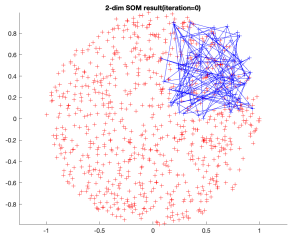

Figure 35: 2-dim SOM(0)

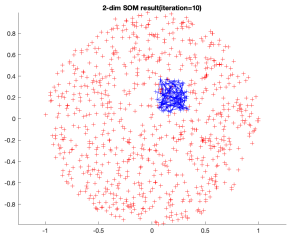

Figure 36: 2-dim SOM(10)

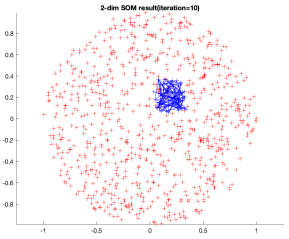

Figure 37: 2-dim SOM(20)

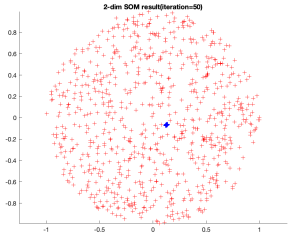

Figure 38: 2-dim SOM(50)

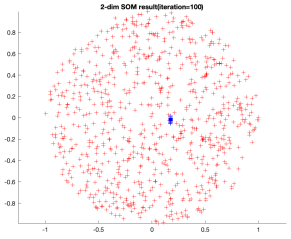

Figure 39: 2-dim SOM(100)

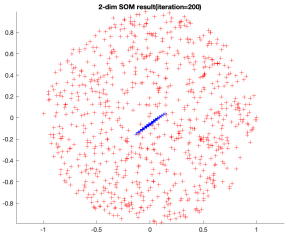

Figure 40: 2-dim SOM(200)

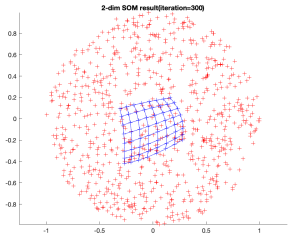

Figure 41: 2-dim SOM(300)

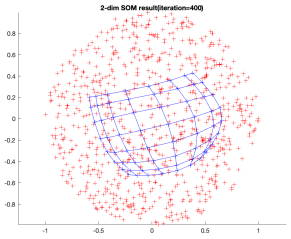

Figure 42: 2-dim SOM(400)

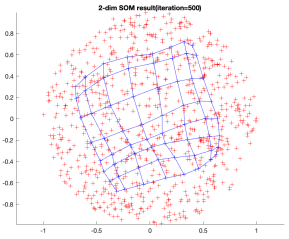

Figure 43: 2-dim SOM(500)

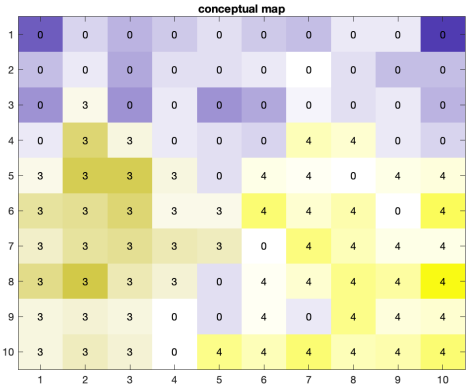

Figure 44: conceptual map

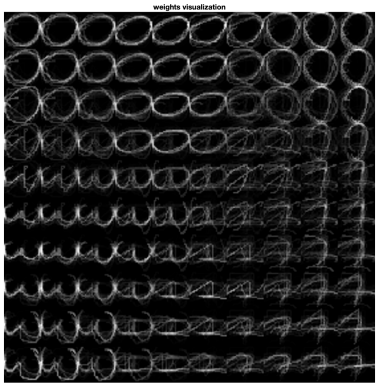

Figure 45: weights visualization

Figure 44 and Figure 45 present conceptual map and weights visualization respectively. In conceptual map, purple represents class 0, green represents class 3 and yellow represents class4. The deeper the color is, the higher possibility of these block(weights) to represent this class. It shows that the map is generally divided into three parts belonging to each class.

In weights visualization, it can tell that there are three parts and the visual effect is just the number.

Numbers close to the edge is more clear while numbers at junction of two classes are more blurry.In the comparison of conceptual map and weights visualization, the distribution of classes is similar.

Figure 46 shows the trajectory of accuracy according to iteration. Accuracy increases as iteration grows. Although the recommended iteration is 1000, accuracy has a better performance at 1500th iteration(about 95% for both train set and test set). For the visualization of weights, it is also observed that, contour of number in each weights block becomes more clear with iteration increasing.

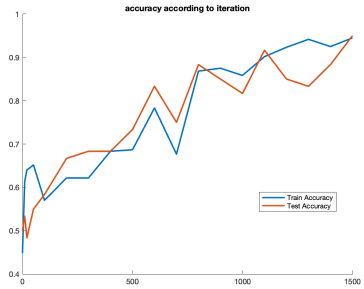

Figure 46: accuracy according to iteration