HW10
Machine Leaning

21600004
Kang Seok-Un

Introduction

Unlike the previous homework, this homework uses 'circle in the square' data and 'function approximation' data, not 'mnist data'. In the case of 'circle in the square' data, data is configured in the form of [Feature 1, Feature2, Label]. In the case of 'function approximation' data, the data is configured in the form of [x, f(x)]. After constructing the radial basis function network model, train using the above two datasets. Predictions are carried out using the trained model.

Experiment

Data Preparing

As mentioned in the Introduction Section, this problem uses two types of datasets. Each dataset consists of train1, train2, and test. Therefore, it is necessary to load data so that each txt file can be used for model training and testing. The process of loading data is shown in Figure 1. And the process of preparing to be used for learning based on the loaded data can be found in Figure 2.

Table 1. Information of dataset about # of samples and dimension

Dataset	cis_train1.txt	cis_train2.txt	cis_test.txt	fa_train1.txt	fa_train2.txt	fa_test.txt
[N x d]	[100 x 3]	[1000 x 3]	[10000 x 3]	[20 x 2]	[100 x 2]	[1000 x 2]

```
100  def load_data(dataset):
101      train1 = dataset + "_train1.txt"
102      train2 = dataset + "_train2.txt"
103      test = dataset +"_test.txt"
104
105      train_dataset1 = pd.read_csv(train1, sep="\t",header=None)
106      train_dataset2 = pd.read_csv(train2, sep="\t",header=None)
107
108      test_dataset = pd.read_csv(test, sep="\t",header=None)
109
110      return train_dataset1.to_numpy(), train_dataset2.to_numpy(), test_dataset.to_numpy()
```

Figure 1. Data Loading

```
117      train_cis_1, train_cis_2, test_cis = load_data("cis")
118
119      test_cis_x_1, test_cis_y_1 = train_cis_1[:, :2], train_cis_1[:, 2:]
120      test_cis_x_2, test_cis_y_2 = train_cis_2[:, :2], train_cis_2[:, 2:]
121      test_cis_x, test_cis_y = test_cis[:, :2], test_cis[:, 2:]
```

Figure 2. Data Splitting

RBFN Model Constructing

In the case of the RBFN model, it was made of one class object. According to RBFN's constructor, K to be used for K-Means Algorithm, learning rate used for back-propagation, and epoch to determine how many trains to be used were received as parameters. And in the case of weight, it was randomized with K-dim. The above process can be seen in Figure 3.

There may be no value in a particular cluster, and even if there is a value in the cluster, there may be cases where the standard deviation is 0. In this case, since a division by zero error occurs, an exception was processed as shown in line 74 of Figure 1.

Next, I will explain training. The training was repeated as much as epochs. Gaussian basis function was implemented at RBF() function. In Line 86, the difference between the actual data and the predicted value was calculated, and then back-propagation was performed by reflecting the learning rate.

```
63  class RBFN():
64      def __init__(self, K=2, lr=0.01, epochs=100):
65          self.k = K
66          self.lr = lr
67          self.epochs = epochs
68
69          self.w = np.random.randn(K)
70          self.b = np.random.randn(1)
71
72      def RBF(self, x, centorid, std):
73          if std == 0.0:
74              return 1
75          else:
76              return np.exp(-1 / (2 * std * std) * np.sum((x-centorid)**2))
77
78      def fit(self, X, y):
79          self.centroids, self.stds = kmeans(X, self.k)
80
81          for epoch in range(self.epochs):
82              for i in range(X.shape[0]):
83                  pi = np.array([self.RBF(X[i], centorid, std) for centorid, std, in zip(self.centroids, self.stds)])
84                  y_hat = self.w.T @ pi + self.b
85
86                  error = -(y[i] - y_hat)
87
88                  self.w = self.w - self.lr * pi * error
89                  self.b = self.b - self.lr * error
90
91      def predict(self, X):
92          y_pred = []
93          for i in range(X.shape[0]):
94              pi = np.array([self.RBF(X[i], centorid, std) for centorid, std, in zip(self.centroids, self.stds)])
95              y_hat = self.w.T @ pi + self.b
96              y_pred.append(y_hat)
97
98          return np.array(y_pred)
```

Figure 3. RBFN Class

Result

In carrying out this homework, the learning rate is set to 0.01, the epoch of cis*.txt is 300, and the epoch of fa*.txt is 500. In addition, a total of k=1 to k=50 were performed differently. However, it is notified that a total of 200 output result images were selectively reflected because there was too large an amount to be included in this report.

cis*.txt

Figure 4 shows the distribution of test data. In the case of train1.txt, compared to train2.txt, there is a difference in the number of data samples, which is considered to have an effect on the accuracy. In addition, it can be seen that the overall accuracy increases as K increases. However, as K increases, accuracy does not continue to increase, and as soon as K exceeds a certain moment, it has been confirmed that the accuracy is tied or lowered.

Table 2. The Performance about cis_*.txt

	cis_train1.txt					cis_train2.txt				
K	1	11	21	31	41	1	11	21	31	41
Accuracy	0.8914	0.8356	0.9152	0.9124	0.9466	0.9100	0.9279	0.918	0.9229	0.9217

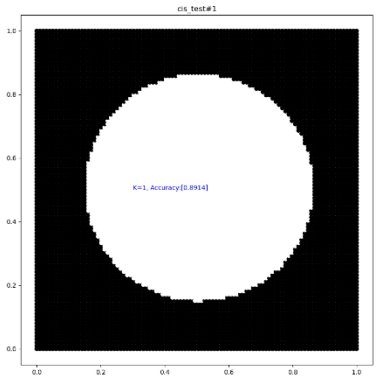

Figure 4. Scatter Image of Actual Data of cis_test.txt

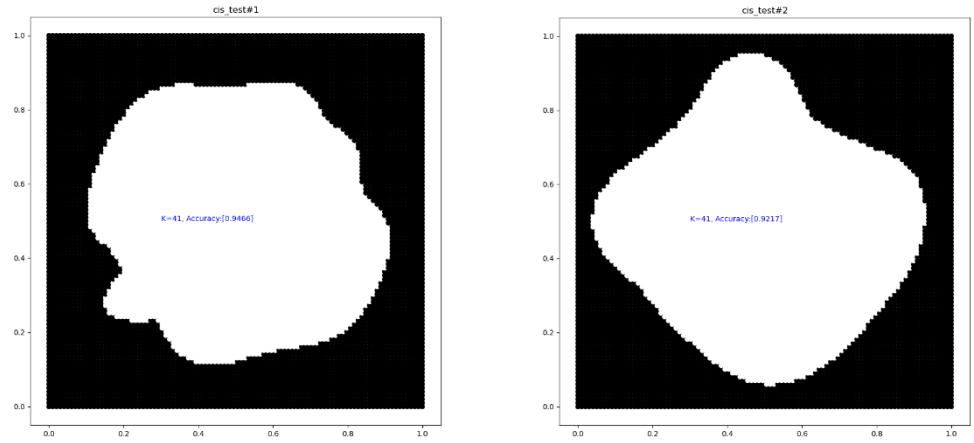

Figure 5. Left: trained by train1.txt(acc:0.947); Right: trained by train2.txt(acc:0.922) when K is 41

fa_*.txt

Figures 6 below are images that are trained with fa_train2.txt and then MSE measured for fa_test.txt and plotted using a predicted value. The red line is the actual data and the blue line is the predicted values.

In the case of this dataset, the number of samples was not large, so even if K increased, it did not work well. In addition, as k increases, it is overfitting. The reason is that when looking at Figures 6, it becomes too complicated as K increases.

Table 3. The Performance about fa_*.txt

	fa_train1.txt					fa_train2.txt				
K	1	11	21	31	41	1	11	21	31	41
MSE	0.04038	0.03643	X	X	X	0.03958	0.01014	0.0151	X	X

Conclusion

I have experience dealing with MLP before. However, RBFN is similar to MLP, but there are different parts, so the difficulty of this homework felt very high. In addition, I felt that if there were many data sets, they could be measured more accurately. The accuracy lower than the accuracy of the pdf provided by the professor was measured. I wonder what part made this difference.

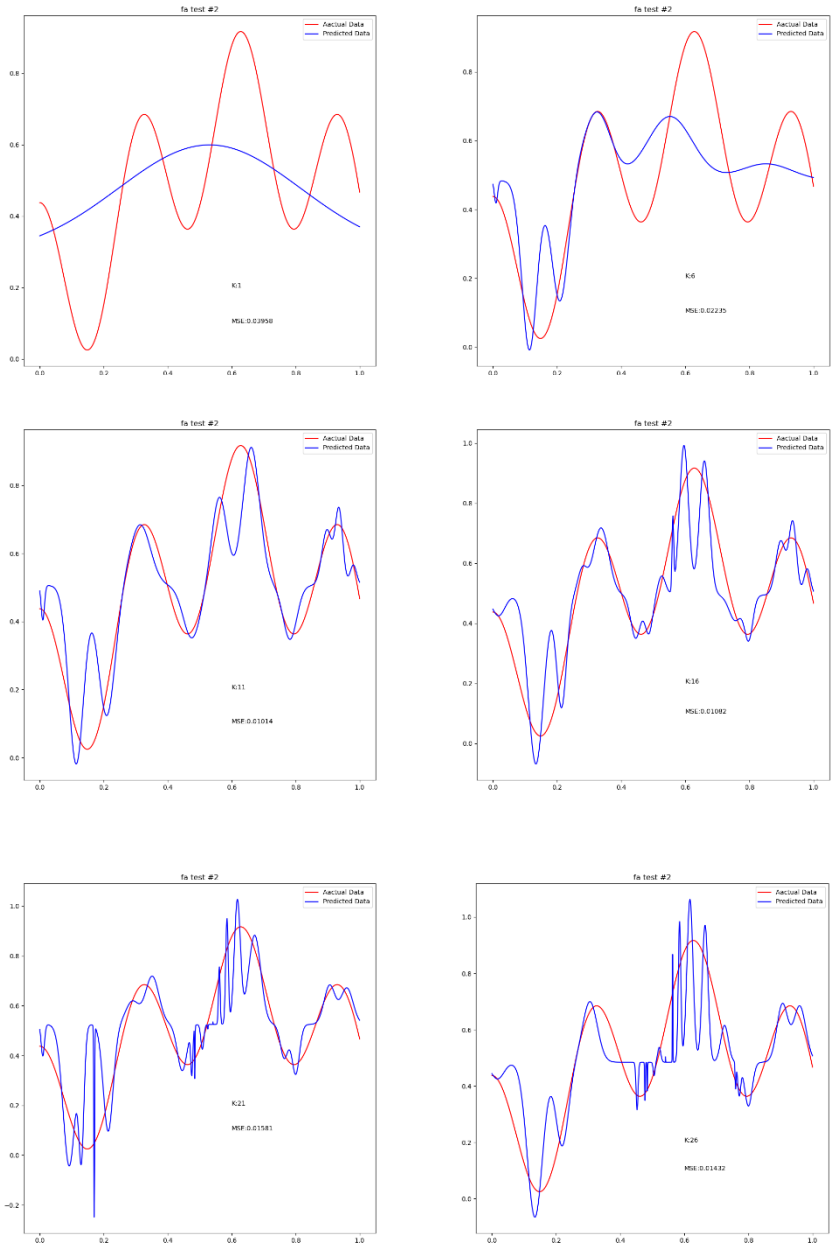

Figure 6. output of fa_test.txt by trained fa_train2.txt (K=1, 6, 11, 16, 21, 26)