EE7207 Neural & Fuzzy Systems

Student Name: Wei Zhifeng
Student ID: G2002825F

1. RBF network

(1) Train an RBF neural network classifier, assuming Gaussian basis function is used. The structure of a RBF is shown below:

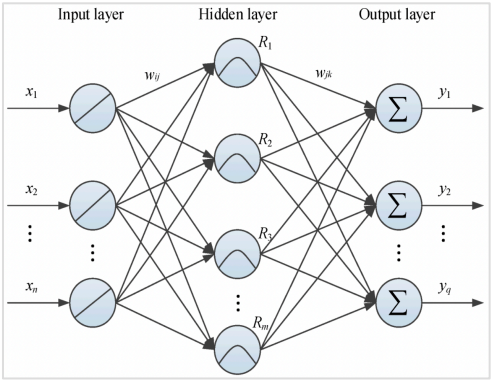

Figure 1.1 RBF Structure

We have 330 data as training sample, each data has 33 dimensions. These data are divided into 2 group, each group are labeled as 1 or -1.

```
>> unique(label_train)

ans =

    -1
     1

>> size(data_train)

ans =

   330    33
```

Figure 1.2 Data Structure

We use PCA algorithm to perform dimension reduction and visualize the data. As is shown in Figure 1.3

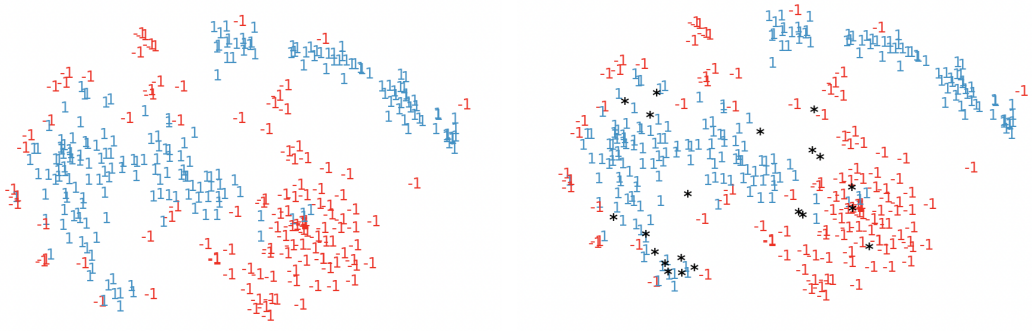

Figure 1.3 Data Distribution

The left figure is the data distribution of training data, and the right figure is training data combined with test data. The black * is test data. From figure 1.3 we could observe roughly the distribution of data

In this assignment, we first use SOM algorithm to choose the center neurons, the structure of SOM is shown below.

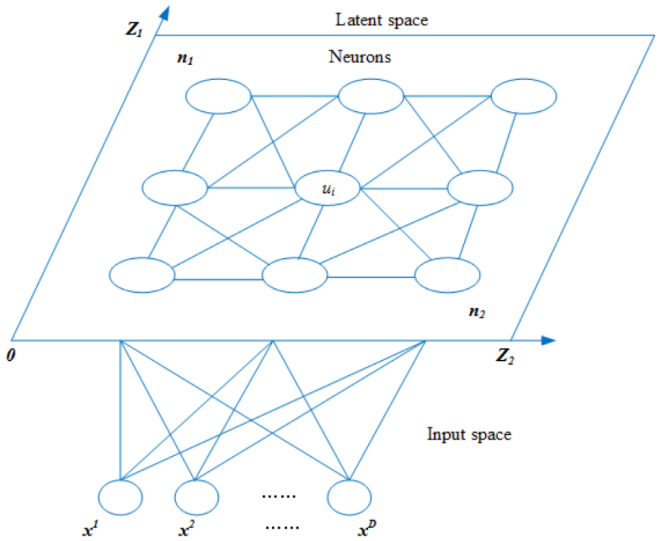

Figure 1.4 SOM Structure

The parameter setting in SOM network:

Self-organizing phase:

$$\tau_1 = 1000 \tag{1.1}$$

$$\tau_2 = \frac{1000}{\ln(2.121)} \tag{1.2}$$

$$\text{learning}_{\text{rate}} = 0.1 * e^{\left(-\frac{iter}{\tau_1}\right)} \tag{1.3}$$

$$\sigma = 2.121 * e^{\left(-\frac{iter}{\tau_2}\right)} \tag{1.4}$$

Iter means the iteration number, it is less than 1000.

Convergence phase:

$$\text{learning}_{\text{rate}} = 0.01 \tag{1.5}$$

$$\text{iter} \leq 500 \tag{1.6}$$

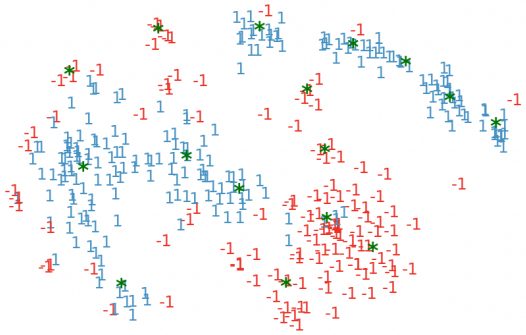

Figure 1.4 SOM Center Distribution

As figure 1.4 shown, the green * is the SOM center we calculated (we choose 16 neurons). From the figure, we could observe that the som center can roughly represent the distribution of the training data.

Then, we used 1,4,16,49,100,225 neurons chosen by SOM respectively as the hidden neurons of RBF network, and use **10-fold** cross validation to evaluate the performance the RBF network, the result is shown as the table below:

Table 1.1 validation Result

Neurons	Average accuracy	Best accuracy
1	0.64848	0.81818
4	0.85455	0.9697
16	0.88182	0.9697
49	0.87273	1
100	0.85455	0.90909
225	0.74242	0.87879

From table 1.1, we can find out that the network with 16 hidden neurons has the best average accuracy. Therefore, we use this model to perform prediction.

The MATLAB code is shown in appendix.

2. Kernel SVM

(2) Train a kernel SVM classifier, assuming Gaussian kernel function is used.

We use the training data to train a kernel SVM, use Gaussian kernel function to map the data to high dimension and then perform classification. We also use 10-fold cross validation to evaluate the performance of SVM classifier. The evaluation result in training data is shown below:

Table 2.1 SVM validation Result

Average Accuracy	Best Accuracy
0.94545	1

From the table, we can find out that the performance of SVM is slightly better than RBF in these training data.

The MATLAB code is shown in appendix.

3. Comparison

(3) Compare and discuss the performance of the two classifiers on the training data.

In our experiment， we use **10-fold cross validation** to evaluate the accuracy of our model. We divided the training data into 10 groups, each time we use 1 group as validation data and train the model using the other 9 groups.

From our experiment, we can see that the average accuracy of RBF is nearly **0.88** (using 16 hidden neurons) while the SVM is **0.94**. Therefore, the SVM's performance might better than RBF in this dataset. However, the best accuracy of these two model during 10-fold validation is both 1, which means that these 2 models both have excellent performance in classify these data.

In practice, accuracy is sometimes not the best criteria of the performance of the model, so we introduced f1 score as the evaluation criteria to evaluate the model.

Table 3.1 model comparison

Model	Acc	precise	Recall	F1-score
RBF	0.969697	0.972028	0.969697	0.969933
SVM	0.969697	0.971014	0.969697	0.969312

We choose 1 fold which RBF and SVM have the same accuracy, and calculate their f1 score. From the table, we could find out that RBF might slightly better than SVM. In conclusion, both classifiers have good performance in classification task, SVM perform better than RBF, but when the data are near the decision boundary, svm is slightly prone to misclassification.

4. Prediction

(4) Predict class labels for testing data using the two classifiers

We use 4 models to predict the test data:

RBF_all is the model trained by all training data, with 16 hidden neurons.

RBF_best is the model using the best weight matrix that obtain during the 10-fold validation. We keep the weight that has the highest validation accuracy and use it to predict the result. The model also has 16 hidden neurons.

SVM_all is the SVM model that trained by all training data.

SVM_best is the SVM using the best parameter that obtain during the 10-fold validation. We keep the parameter that has the highest validation accuracy and use it to predict the result.

Table 4.1 Predict Result

Model	Predict Result
RBF_best	1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1, 1, -1, 1
RBF_all	1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1, 1, -1, 1
SVM_best	1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1,1, -1, 1
SVM_all	1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1,1, -1, 1

In general, these models have similar prediction result. And there are 3 different prediction result in the test data, these 3 data might be in the decision boundary:

12: [1,1,-0.5421,1,-1,1,-1,1,0.36217,1,-0.41119, 1,1,1,-1,1,-0.29354,1,-0.93599,1,1,1,1,1,-0.40888,1,-0.62745,1,-1,1,-1,1,-1]

14: [1,1,-0.867,1,0.2228,0.85492,-0.39896,1,-0.1209,1,0.35147,1,0.07772,1,-0.14767,1,-1,1,-1,0.61831,0.158030,1,0.62349,1 -0.17012,1,0.35924,1,-0.66494,1,0.88428,1,-0.18826]

16: [1,1,-0.867,1,0.2228,0.85492,-0.39896,1,-0.1209,1,0.35147,1,0.07772,1,-0.14767,1,-1,1,-1,0.61831,0.158030,1,0.62349,1 -0.17012,1,0.35924,1,-0.66494,1,0.88428,1,-0.18826]

Appendix

SOM:

```
clc,clear
load data_train
data=data_train;%%330*33 data
m=15;n=15;%%m*n network
rand('seed',5);
disp(size(data,2))
W_som=rand(m*n,size(data,2));%%25*33 weight   -
radom generate m*n center

%normalize to make sure the weights small
W_som=W_som./(sum(W_som,2)*10);
%Self-organizing phase
learning_rate_start=0.1;learning_rate=learning_rate_start;tao1=1000;%
% learning rate change
sigma_start=2.121;sigma=sigma_start;tao2=1000/log(sigma);%% variance
change
iterations = 1000;
[I,J]=ind2sub([m,n],1:m*n);% change to array index
length(I)
for iter=1:iterations
    for i=1:size(data,1)%330,data
       Sample=data(i,:);%%ith Sampleple
       % Compute W_somining neurons
       [minDist,win_index]=min(dist(Sample,W_som'));
       [win_row,win_col]=ind2sub([m,n],win_index);
       % distance
```

```
       for x=1:length(I)% I=m*n 遍历每个 nerual 计算 hij
           distance(x)=exp(((I(x)-win_row)^2+(J(x)-win_col)^2)/(-
2*sigma^2));%gussion distance
       end
       % iterate W_som
       for j=1:m*n
           W_som(j,:)=W_som(j,:)+learning_rate*distance(j)*(Sample-
W_som(j,:));%update w

       end
    end
    %change learning_rate and sigma
    learning_rate=learning_rate_start*exp(-iter/tao1);
    sigma=sigma_start*exp(-iter/tao2);
    if (learning_rate<0.01)
        learning_rate=0.01;
```

```
    end
end
save W_som;
%Convergence phase
W_som_Con=W_som;
iterations2=size(W_som,1)*500;
learning_rate2=0.01;%fixed
for iter=1:iterations2
    for i=1:size(data,1)%330,data
        Sample=data(i,:);%%ith Sampleple
        % Compute W_somwining neurons
        [minDist,win_index]=min(dist(Sample,W_som_Con'));
%         W_som_Con(win_index,:)=W_som_Con(win_index,:)+learning_rate2*(Sample-W_som_Con(win_index,:));
        [win_row,win_col]=ind2sub([m,n],win_index);
        % distance
        for x=1:length(I)
            if((I(x)==win_row)&&J(x)==win_col)
                distance(x)=1;
            else
                distance(x)=0;
            end
        end
        % iterate W_som
        for j=1:m*n
            W_som_Con(j,:)=W_som_Con(j,:)+learning_rate2*distance(j)*(Sample-W_som_Con(j,:));
```

```
        end
    end
    k=iter/iterations2;
    for i=1:10
        if(k==(i*0.1))
            disp(i)%print i
        end
    end
end
save W_som_Con;
xlswrite('center_vectors_Con.xls',W_som_Con)
[idx,Center_kmeans] = kmeans(data_train,m*n);% choose the number you need
save  Center_kmeans;
xlswrite('center_vectors_kmeans.xls',Center_kmeans)
```

RBF:

```
clear
clc
load label_train
load data_test
load data_train
load W_som_Con
train_data=data_train;%% 330*33
cvindex = crossvalind("Kfold",size(train_data,1),10);
center=W_som_Con;
sum_accuracy=0;
accuracy_best=0;
%10-fold
for i=1:10
    test_index=(cvindex == i);
    train_index = ~test_index;
    valid_data=train_data(test_index,:);
    valid_label=label_train(test_index,:);
    label=label_train(train_index,:);
    data=train_data(train_index,:);

    max_disance=0;
    %Compute the max distance
    for i=1:size(data,1)
        for j=1:size(data,1)
            if max_disance<norm(data(i,:)-data(j,:))
                max_disance=norm(data(i,:)-data(j,:));
            end
        end
    end
    %Compute sigma
    sigma=max_disance/sqrt(2*size(center,1));
    %computer fhi %330*16
    for i=1:size(data,1)
        xk=data(i,:);%% 1*33
        for j=1:size(center,1)%16
            sum=0;
            for k=1:size(center,2)%33
                sum=sum+(xk(k)-center(j,k))^2;
            fhi(i,j)=exp(sum/(-2*sigma^2));
            end
        end
    end

```

```
    %compute w
    d=label;% 330*1
    %W_RBF=pinv(fhi'*fhi)*fhi'*d*10
    W_RBF=pinv(fhi'*fhi)*fhi'*d;
    Predict_result=fhi*W_RBF;
    length(Predict_result);
    for i=1:length(Predict_result)
        if (Predict_result(i)>=0)
            Predict_result(i)=1;
        else
            Predict_result(i)=-1;
        end
    end
    sum_Predict_result=0;
    for i=1:length(label)
        if(Predict_result(i)==label(i))
            sum_Predict_result=sum_Predict_result+1;
        end
    end
    accuracy=sum_Predict_result/length(label);
    disp(['traning_accuracy while RBF centers is computed by SOM:= ',
num2str(accuracy)]);

    %validation
    size(valid_data,1);
    for i=1:size(valid_data,1)
        xk=valid_data(i,:);%% 1*33
        for j=1:size(center,1)%16
            sum=0;
            for k=1:size(center,2)%33
                sum=sum+(xk(k)-center(j,k))^2;
            fhi2(i,j)=exp(sum/(-2*sigma^2));
            end
        end
    end
   validation_result=fhi2*W_RBF;
    for i=1:length(validation_result)
        if (validation_result(i)>=0)
            validation_result(i)=1;
        else
            validation_result(i)=-1;
        end
    end
    sum_Valid_result=0;
```

```
    for i=1:length(valid_label)
        if(validation_result(i)==valid_label(i))
            sum_Valid_result=sum_Valid_result+1;
        end
    end
    accuracy=sum_Valid_result/length(valid_label);
    disp(['validation_accuracy while RBF centers is computed by SOM:=
 ',num2str(accuracy)]);
    sum_accuracy = sum_accuracy+accuracy;
    if (accuracy > accuracy_best)
        accuracy_best=accuracy;
        best_w=W_RBF;
    end
end
  disp(['average_accuracy while RBF centers is computed by SOM:= ',n
um2str(sum_accuracy/10)]);
  disp(['best_accuracy while RBF centers is computed by SOM:= ',num2
str(accuracy_best)]);
  W_RBF=best_w;%use the best w
  %test data
  size(data_test,1)
  for i=1:size(data_test,1)
      xk=data_test(i,:);%% 1*33
      for j=1:size(center,1)%16
          sum=0;
          for k=1:size(center,2)%33
              sum=sum+(xk(k)-center(j,k))^2;
          fhi3(i,j)=exp(sum/(-2*sigma^2));
          end
      end
  end
  Predict_result2=fhi3*W_RBF;
  xlswrite('Predict_result_test_data_before.xls',Predict_result2)
  for i=1:length(Predict_result2)
      if (Predict_result2(i)>=0)
          Predict_result2(i)=1;
      else
          Predict_result2(i)=-1;
      end
  end

  Predict_result2;
  plot(Predict_result2,'rx')
  xlswrite('Predict_result_test_data.xls',Predict_result2)
```

SVM:

```
load data_train
load label_train
load data_test
train_data=data_train;
cvindex = crossvalind("Kfold",size(train_data,1),10);
sum_accuracy=0;
best_accuracy=0;
for i=1:10
    i
    test_index=(cvindex == i);
    train_index = ~test_index;
    valid_data=train_data(test_index,:);
    valid_label=label_train(test_index,:);
    label=label_train(train_index,:);
    data=train_data(train_index,:);
    svmClassifier = fitcsvm(data,label,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');
    [predic_labels, scores] = predict(svmClassifier, valid_data);
    count=0;
    for i=1:size(valid_data,1)
       if valid_label(i) == predic_labels(i)
           count=count+1;
       end
    end
    accuracy=count/length(valid_label);
    disp(['validation_accuracy  by svm:= ',num2str(accuracy)]);
    sum_accuracy = sum_accuracy+accuracy;
    if (accuracy > best_accuracy)
        best_accuracy=accuracy;
        best_svm=svmClassifier;
    end
end
disp(['average_accuracy  by svm:= ',num2str(sum_accuracy/10)]);
disp(['best_accuracy  by svm:= ',num2str(best_accuracy)]);
all_classify=fitcsvm(data_train,label_train,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');
all_result=predict(all_classify,data_test);
result=predict(best_svm,data_test);
```