CS303 Assignment Report

December 30, 2021

11912625 Hanqi Su

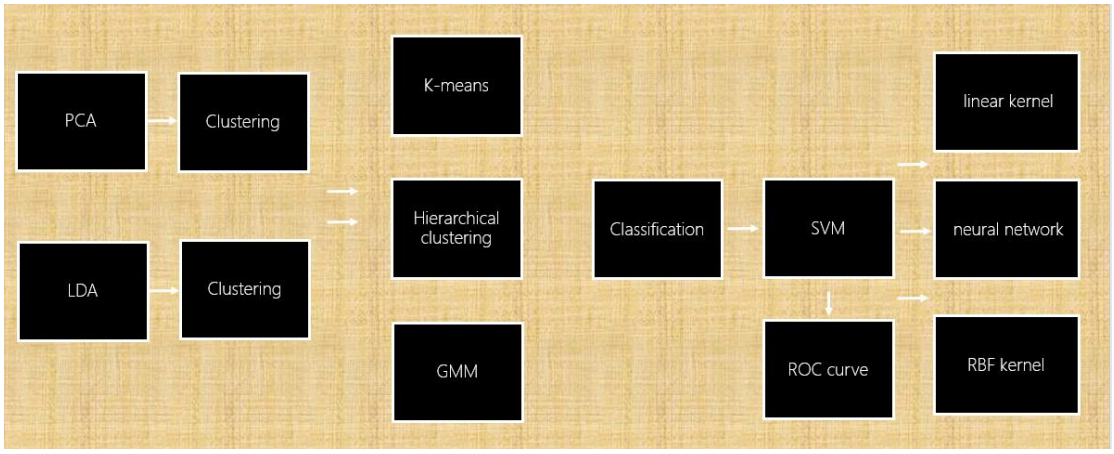
Requirements:

Code for producing all results for each task can be shown in Project2_MainCode.m file.
LDA_method.m and PCA_method.m implement the function of PCA and LDA.
In these three files, most of the code is commented so people can understand it.

1 Introduction

In this assignment, I implemented two methods: PCA and LDA to achieve the dimensionality reduction by using the first two principal components. And then I visualize data from MNIST and compared the clustering results after dimensionality reduction. I also discuss whether the resulting clusters match well the actual ground truth partition of classes. For the classification, it is a two-class classification problem. I use SVM with a RBF kernel, SVM with a linear kernel, and a neural network classifier with one hidden layer to classify the dataset in a 5-fold cross validation setting. Compare and discuss the results using the obtained validation accuracy. Create and plot the ROC curves of the results using the three different classifiers, and compare their performance using the area under the ROC curve (AUC). Last but not least, I pick one parameter called KernelScale to tune in this dataset and get some good results.

Overall framework is shown as below.



2 Data Processing

I use MATLAAB to load the data from mnist-1-5-8.mat. It contains two variables: images and labels. For each image, the size of image is 28x28 pixels. And in the dataset it convert the image matrix into a vector (a 784-dim feature vector for each image in

our dataset). So I use image_data to contain 600 images with each imgae represented by 784x1 matrix and the dimension of image_data is 784x600. And label contains each the number in the image it has. And here are three classes: number 1 , 5 , 8.

```
% load data
load mnist-1-5-8.mat
class = [1,5,8];
image_data = images;
label = labels;
```

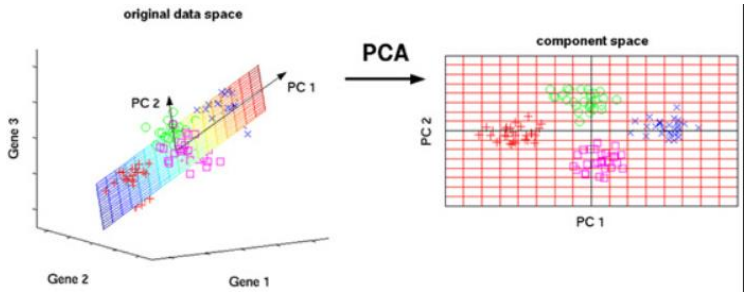
3 Question 1

3.1 Dimensionality Reduction Using PCA

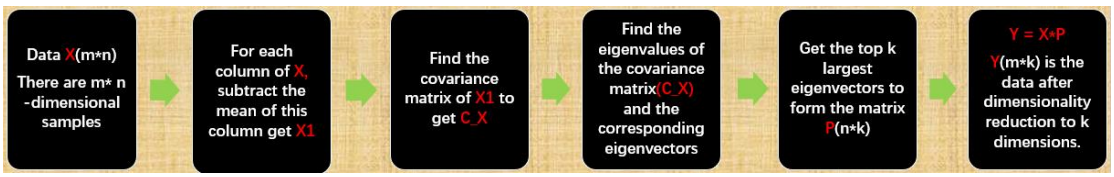
PCA (Principal Component Analysis) is a common approach to data analysis, often used to reduce the dimensionality of high-dimensional data, and can be used to extract the main feature components of the data.

The main idea of PCA is to map the N-dimensional features to the K-dimension, which is a new orthogonal feature, also known as the principal component, which is reconstructed on the basis of the original N-dimensional features.

The work of PCA is to find a set of mutually orthogonal coordinate axes sequentially from the original space. The selection of the new coordinate axes is closely related to the data itself. Among them, the first new coordinate axis is selected in the direction of the largest variance in the original data, the second new coordinate axis is selected in the plane orthogonal to the first coordinate axis to make the largest variance, and the third axis is selected in the plane orthogonal to the first and second axes to make the largest variance. And so on, you get n of these axes. Retaining only the dimension features containing most of the variance, while ignoring the feature dimensions containing almost zero variance, so as to achieve dimensionality reduction of data features.



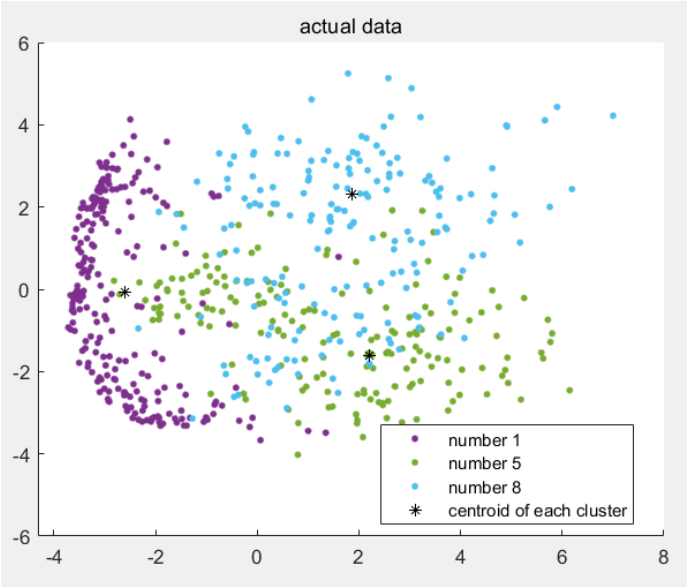
So how to solve PCA? We need to achieve from n-dimensional to k-dimensional operation. The following is the operation to implement.



The following code and figure compute and visualize the data after using PCA

method.

```
function new_low_dim_matrix = PCA_method(origin_high_dim_matrix,k)
% achieve PCA
 % output information about image with first k leading vectors
 % dim(origin_high_dim_matrix)=600x784
     mean_X = mean(origin_high_dim_matrix); % 600x1
     high_dim_matrix = origin_high_dim_matrix - repmat(mean_X, size(origin_high_dim_matrix,1), 1);
     covariance_X = cov(high_dim_matrix);
     [V, D] = eigs(covariance_X); % number of V = number of image characteristic = 784
     new_low_dim_matrix = high_dim_matrix*V(:,1:k);% get the projection on the first k leading vectors

end
```



3.2 Clustering Using K-means

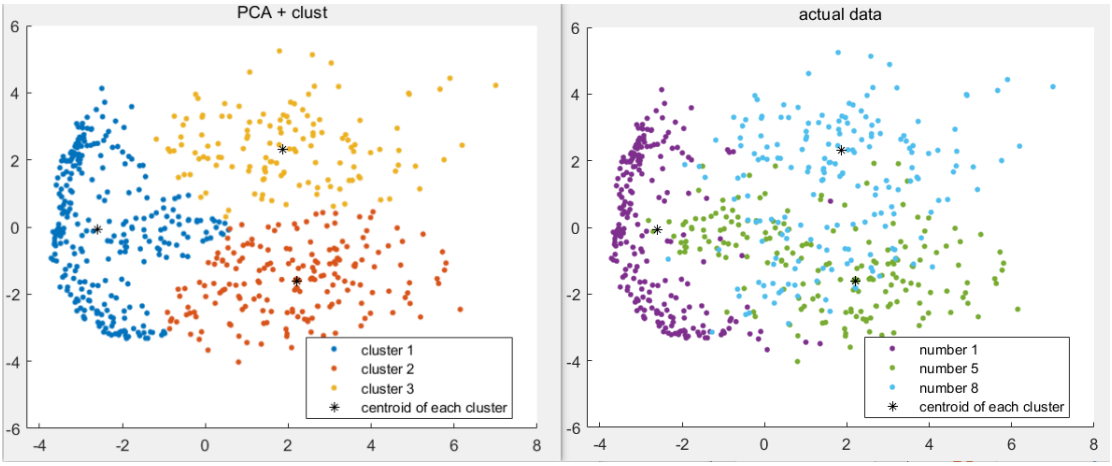
K-means algorithm is a clustering algorithm based on partition. It takes K as the parameter to divide N data objects into K clusters, so that the similarity within the cluster is high and the similarity between clusters is low.

K-means algorithm Process:

Firstly, k data objects are randomly selected, and each data object represents a cluster center, that is, k initial centers are selected. For each remaining object, it is assigned to the cluster corresponding to its most similar cluster center according to its similarity (distance) to each cluster center. The average value of all objects in each cluster is then recalculated as the new cluster center. The above process is repeated until the criterion function converges, that is, the cluster center does not change significantly. The mean square error is usually used as a criterion function, which minimizes the sum of squares of the distances from each point to the center of the nearest cluster.

So the following code computes the clustering result using K-means. The comparation of clustering result and ground truth clusters are shown in pair. I draw the data which has done in PCA process and K-means method. And the actual image is drawn for comparison.

```
% 1 using k-means
[idx,C] = kmeans(PCA_component,3,'Distance','cityblock');
% % using cityblock
% [idx,C] = kmeans(PCA_component,3);
% generate color
color = lines(6);
figure(1)
hold on
% gscatter:
% creates a scatter plot of x and y, grouped by g. The inputs x and y are
% specifies the marker color clr for each group
%gscatter(PCA_component(:,1),PCA_component(:,2),idx,'bgm')
gscatter(PCA_component(:,1),PCA_component(:,2),idx,color(1:3,:))
plot(C(:,1),C(:,2),'k*')
title("PCA + clust")
legend('cluster 1','cluster 2','cluster 3','centroid of each cluster')
hold off

figure(2)
hold on
%gscatter(PCA_component(:,1),PCA_component(:,2),label,'gmb')
gscatter(PCA_component(:,1),PCA_component(:,2),label,color(4:6,:))
plot(C(:,1),C(:,2),'k*')
title("actual data")
legend('number 1','number 5','number 8','centroid of each cluster')
hold off
```



On the left is an image of the classification using PCA, and on the right is the actual distribution image. We can see that using PCA and K-means clustering does not separate out certain points very well.

For the reason why it cannot perform well, from the picture on the right, we can see that there is a wide range of crossover between the real data points of the numbers 5 and 8, which is the awkward situation of you in me, me in you. It is difficult to achieve a good partition of data using PCA as an unsupervised dimensionality reduction method. Since the goal of principal component analysis (PCA) is to find the direction that best explain the variations of data. Moreover, K-means algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within-cluster sum-of-squares. Thus it cannot cluster the embedded data to meet the ground truth clusters.

3.3 Clustering Using Hierarchical Clustering

Hierarchical Clustering is a kind of Clustering algorithm, which creates a

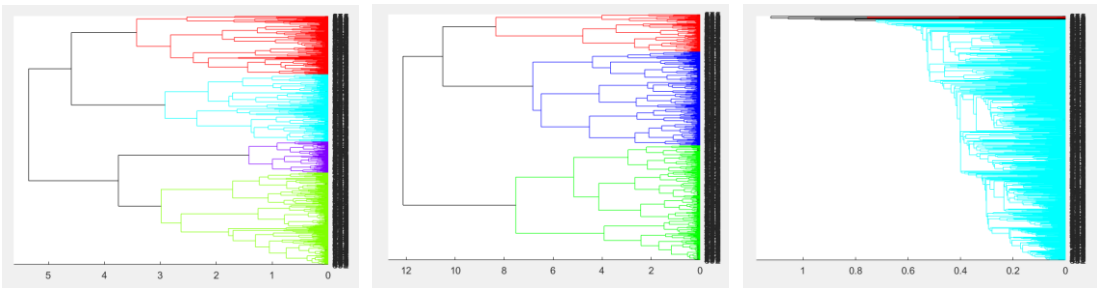
hierarchical nested clustering tree by calculating the similarity among data points of different categories. In a cluster tree, the original data points of different categories are the lowest layer of the tree, and the top layer of the tree is the root node of a cluster. One of the advantages of hierarchical clustering algorithm compared with partition clustering algorithm is that the clustering of data sets can be displayed at different scales (hierarchies).

There are two ways to create clustering tree: bottom-up merging and top-down merging. The hierarchical clustering algorithm can be classified as Agglomerative or Divisive.

The two approaches are not superior to each other, but in practical application, depending on the characteristics of the data and the number of "classes" you want, consider whether top-down or bottom-up is faster. The Linkage methods are the shortest, longest, middle, class average, etc. (class average is usually the most common and most useful method, on the one hand, because of its monotonicity, on the other hand, because of its space expansion/concentration degree moderate). To make up for the deficiency of decomposition and merging, hierarchical merging is often combined with other clustering methods, such as cyclic location.

The following code and figure are using hierarchical clustering to solve the problem. And it shows that linkage choose 'average' fit well than 'single' and 'complete' in some degree.

```
% 2 using hierarchical clustering
%Compute three clusters of the PCA process data using single-linkage
Z = linkage(PCA_component,'average','euclidean'); %create the linkage tree using average-link
c = cluster(Z,'maxclust',3);
% See how the cluster assignments correspond to the three number: 1 5 8
crosstab(c, label)
% Create a dendrogram plot of Z, and visualize it.
figure(3)
% dendrogram(Z,0)
% cutoff = median([Z(end-2,3) Z(end-1,3)]);
%dendrogram(Z,'ColorThreshold',cutoff)
dendrogram(Z,0,'Orientation','left','ColorThreshold','default')
```



average complete single

Why does the linkage select Average work better? Here we analyze three kinds of connection ways theoretically. Single Linkage: The Linkage is defined as the distance between the nearest two combination data points. This approach is susceptible to extreme values. Two very similar combined data points may be combined because one of the extreme data points is close to each other. Complete Linkage: The Complete Linkage is calculated the opposite of a Single Linkage, using the distance between the

two most distant data points as the distance between the two combined data points. The problem with a Complete Linkage is also the opposite of a Single Linkage, where two dissimilar combination data points may not be combined due to the distance between the extreme values. Average Linkage: The Average Linkage is calculated by calculating the distance between each of the two combined data points and all the other data points. The mean of all distances is taken as the distance between the two combined data points. This method is more computationally intensive, but the result is more reasonable than the first two methods. So linkage choose 'average' fit well than 'single' and 'complete' in some degree.

3.4 Cluster Using GMM

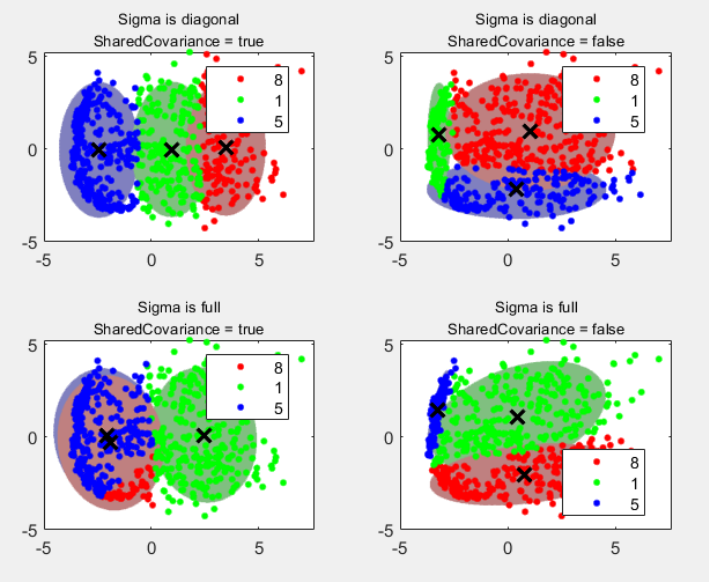
Gaussian Mixture Model, commonly referred to as GMM, is a widely used clustering algorithm in the industry, which uses Gaussian distribution as a parameter Model and is trained with Expectation Maximization (EM) algorithm.

Gaussian mixture model is a simple extension of gaussian model, and GMM uses the combination of multiple Gaussian distributions to describe the data distribution. Firstly, the distribution probability is the sum of K Gaussian distributions. Each Gaussian distribution has its own mean and variance parameters, as well as corresponding weight parameters. The weight value must be positive, and the sum of ownership weight must be equal to 1 to ensure that the value given by the formula is a reasonable probability density value. In other words, if we combine the input space corresponding to this formula, the result will be equal to 1. Intuitively speaking, the EM training process of the model is like this: We judge whether a model fits well by observing the proximity between the probability value of the sample and the probability value of the model. We then adjust the model so that the new model better fits the probability values of the sample. We iterate this process many times until the two probability values are very close, and we stop updating and finish the model training. We will implement this process with an algorithm, using the model generated data to determine the likelihood value, that is, the model to calculate the expected value of the data. Maximize the expected value by updating the parameters μ and σ. This process can be iterated over until the generated parameters change very little between the two iterations. This process is similar to the algorithm training process of K-means (K-means constantly updates the class center to maximize the result), except that in the Gaussian model here, we need to update two parameters simultaneously: the mean and standard deviation of the distribution.

The following code and figure are using GMM to solve the problem. It just follow the official guide from MATLAB. And it shows that when sigma is diagonal and sharecovarience equals to false, the fitting result is the best.

```
% 3 using GMM
X = PCA_component;
[n,p] = size(X);
figure(4)
plot(X(:,1),X(:,2),'.','MarkerSize',15);
title('original data');

rng(3);
k = 3; % Number of GMM components
options = statset('MaxIter',1000);

Sigma = {'diagonal','full'}; % Options for covariance matrix type
nSigma = numel(Sigma);

SharedCovariance = {true,false};
% Indicator for identical or nonidentical covariance matrices
SCtext = {'true','false'};
nSC = numel(SharedCovariance);

d = 500; % Grid length
x1 = linspace(min(X(:,1))-2, max(X(:,1))+2, d);
x2 = linspace(min(X(:,2))-2, max(X(:,2))+2, d);
[x1grid,x2grid] = meshgrid(x1,x2);
X0 = [x1grid(:) x2grid(:)];
```

```
figure(5)
threshold = sqrt(chi2inv(0.99,2));
count = 1;
for i = 1:nSigma
    for j = 1:nSC
        gmfit = fitgmdist(X,k,'CovarianceType',Sigma{i}, ...
            'SharedCovariance',SharedCovariance{j},'Options',options); % Fitted GMM
        clusterX = cluster(gmfit,X); % Cluster index
        mahalDist = mahal(gmfit,X0); % Distance from each grid point to each GMM component
        % Draw ellipsoids over each GMM component and show clustering result.
        subplot(2,2,count);
        h1 = gscatter(X(:,1),X(:,2),clusterX);
        hold on
            for m = 1:k
                idx = mahalDist(:,m)<=threshold;
                Color = h1(m).Color*0.75 - 0.5*(h1(m).Color - 1);
                h2 = plot(X0(idx,1),X0(idx,2),'.','Color',Color,'MarkerSize',1);
                uistack(h2,'bottom');
            end
        plot(gmfit.mu(:,1),gmfit.mu(:,2),'kx','LineWidth',2,'MarkerSize',10)
        title(sprintf('Sigma is %s\nSharedCovariance = %s',Sigma{i},SCtext{j}),'FontSize',8)
        legend(h1,{'8','1','5'})
        hold off
        count = count + 1;
    end
end
```



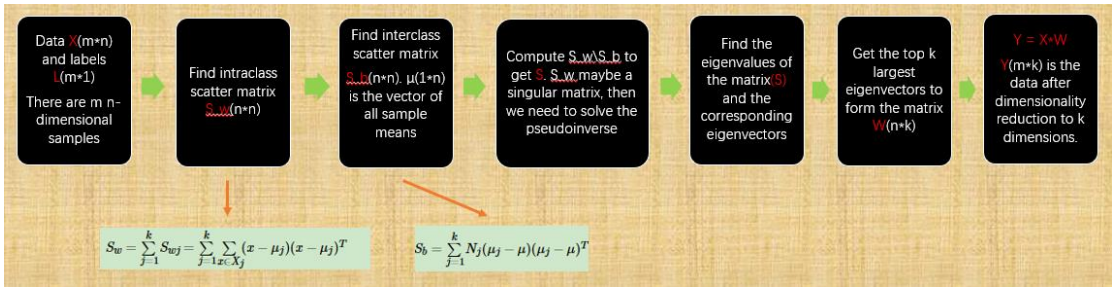
4 Question 2

4.1 Dimensionality Reduction Using LDA

Linear Discriminant Analysis (LDA) is a classical method for dimension reduction of supervised data.

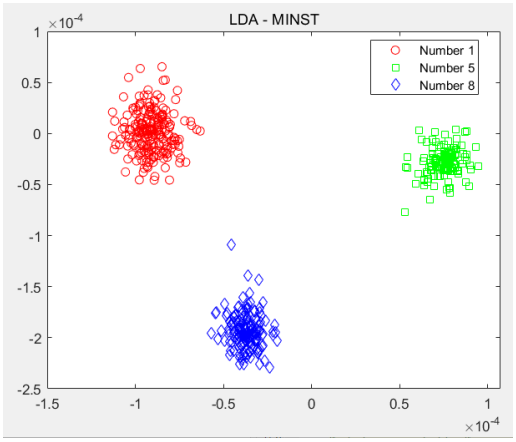
The main idea of LDA is to project the data in a high-dimensional space to a lower-dimensional space, and ensure that the intra-class variance of each category is small and the inter-class mean difference is large after the projection, which means that after the high-dimensional data in the same category is projected to a lower-dimensional space, the same categories are grouped together, but different categories are far apart.This is different from PCA, which is an unsupervised dimensionality reduction technique that does not take into account the output of the sample categories.

So how to solve LDA? We need to achieve from n-dimensional to k-dimensional operation. The following is the operation to implement.



The following code and figure compute and visualize the data after using LDA method.

```
function w = LDA_method(original_matrix, labels, class, k)
% achieve LDA
    a = labels;
    N = length(class);  % the number of the classes
    [~, col] = size(original_matrix);  % col = image of characteristic = 784
    means_C = zeros(N, col);
    class_M = cell(N, 1); % Put different numbers into different groups

    % seperate different value("1","5","8") in different class, and compute mean
    for i = 1:N
        class_M{i, 1} = original_matrix(a == class(i), :);
        means_C(i, :) = mean(original_matrix(a == class(i), :));
    end

    mean_X = mean(original_matrix); %  Average intensity of each location information point
    % generate empty S_w and S_b
    S_w = zeros(col);
    S_b = zeros(col);
    % compute S_w
    for i = 1:N
        % (class_M{i,1} - mean_X) is Mw -> within-class scatter
        sigema = (class_M{i, 1} - mean_X)' * (class_M{i, 1} - mean_X);
        S_w = S_w + sigema;
    end
    % compute S_b
    for i = 1:N
        % (means_C(i,:) - mean_X) is Mb
        % between class scatter
        S_b = S_b + size(class_M{i,1},1) * (means_C(i,:) - mean_X)' * (means_C(i,:) - mean_X);
    end

    % computing the LDA projection vector w
    W = pinv(S_w) * S_b; % optimal transformation by solving a generalized eigenvalue problem
    [v, ~] = eigs(W); % we need to get the eigenvector
    w = real(v(:, 1:k));
end
```

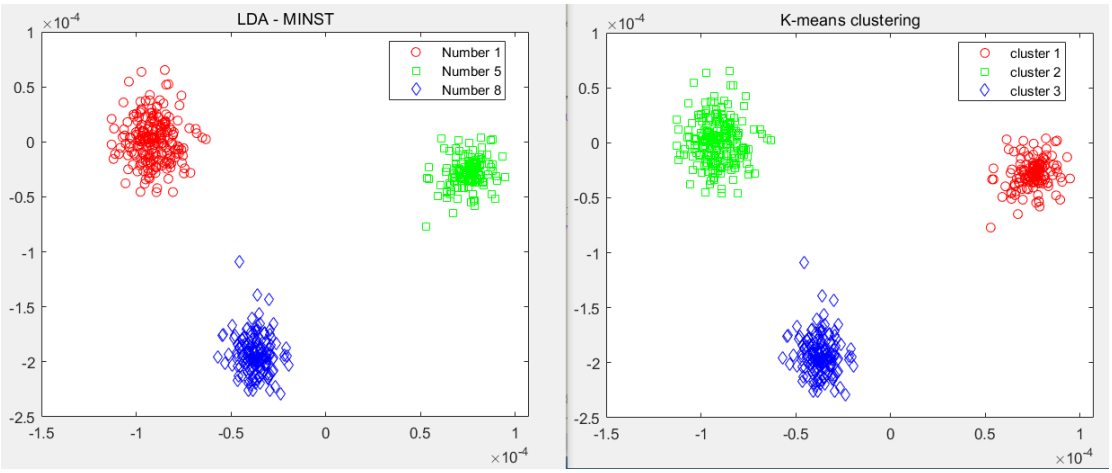


4.2 Clustering Using K-means

K-means algorithm is a clustering algorithm based on partition. It takes K as the parameter to divide N data objects into K clusters, so that the similarity within the cluster is high and the similarity between clusters is low.

So the following code computes the clustering result using K-means. The comparation of clustering result and ground truth clusters are shown in pair. I draw the data which has done in LDA process and K-means method.

```
%% LDA
LDA_w = LDA_method(image_data', label, class, 2);
% computing the projection score
score = image_data'*LDA_w;
figure(7)
gscatter(score(:,1), score(:,2), label, 'rgb','osd')
legend('Number 1','Number 5','Number 8')
title("LDA - MINST")
figure(8)
[idx, C] = kmeans(score, 3);
gscatter(score(:,1),score(:,2),idx,'rgb','osd')
legend('cluster 1','cluster 2','cluster 3')
title("K-means clustering")
```



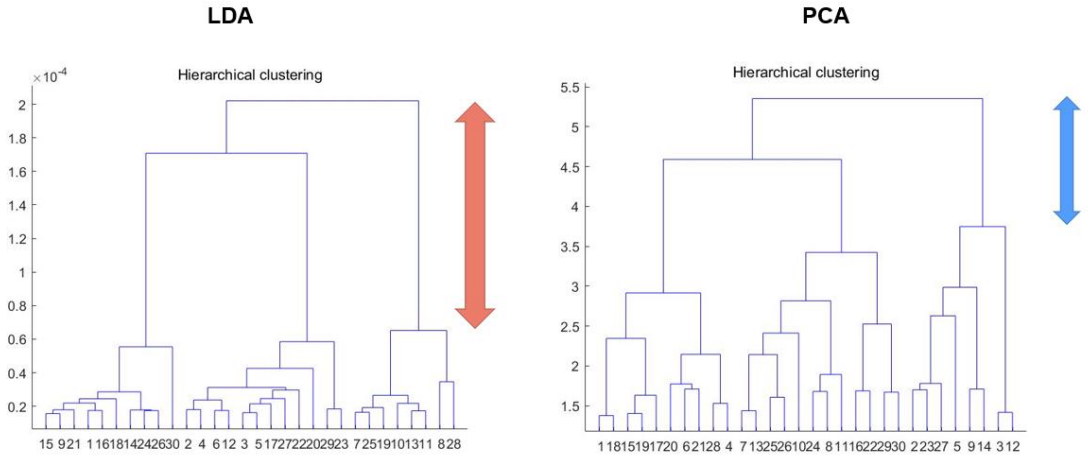
From the image weight, we can find that after data dimensionality reduction through LDA, the similarity inside the cluster is high, but the similarity between the clusters is low. Each of the three types of numbers has an aggregation area, which can achieve the clustering effect well. Using K-means for clustering, the results are still pretty good. The treatment result after LDA and K-means is much better than that of PCA.

4.3 Clustering Using Hierarchical Clustering

Hierarchical Clustering is a kind of Clustering algorithm, which creates a hierarchical nested clustering tree by calculating the similarity among data points of different categories. In a cluster tree, the original data points of different categories are the lowest layer of the tree, and the top layer of the tree is the root node of a cluster. One of the advantages of hierarchical clustering algorithm compared with partition

clustering algorithm is that the clustering of data sets can be displayed at different scales (hierarchies).

The following figure are using hierarchical clustering to solve the problem. And it shows that linkage choose 'average' fit well than 'single' and 'complete' in some degree. The processing results of LDA+ hierarchical clustering and PCA+ hierarchical clustering were analyzed and compared. We will find that the processing results of LDA are better than those of PCA. And we know, as you can see from the top of the graph, this is at the beginning of classification, that the longer the vertical axis is, the more different the classes are at the beginning of classification. The shorter the vertical axis, the smaller the difference between the categories. This classification is undoubtedly the best. Then back to our image, the distance between classes is reflected by the length of the vertical axis. We can obviously find that the distance between classes is very large at the beginning of LDA classification, while the distance within classes is very small at the end of LDA classification. In contrast, PCA images show that the distance between classes is not very large at the beginning of classification, and the distance within classes is not very small at the end of classification. This shows that LDA is better for processing.



4.4 Comparison between LDA and PCA

From the perspective of the solving process of PCA and LDA dimensionality reduction methods, they are indeed very similar, but the corresponding principles are different. First of all, PCA selects the direction with the largest variance of projected data. Since it is unsupervised, the larger the variance of PCA hypothesis is, the more information is contained. Using principal components to represent original data can remove redundant dimensions and achieve dimensionality reduction. LDA, on the other hand, selects the direction with small intra-class variance and large inter-class variance after projection. In order to find the discriminant dimensions in the data, different categories can be distinguished as far as possible after the original data is projected in these directions.

PCA pros and cons

Advantage: 1. The amount of information only needs to be measured by variance and is not affected by factors outside the data set. 2. Each principal component is

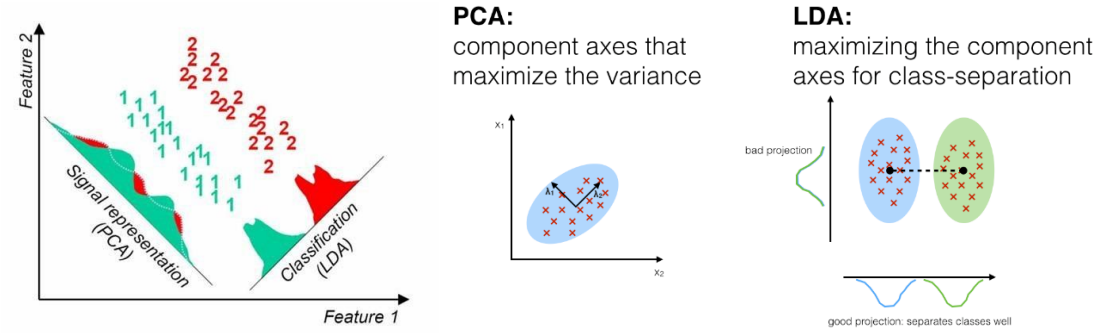
orthogonal, which can eliminate the factors influencing each other of the original data components. 3. The calculation method is simple, the main operation is eigenvalue decomposition, easy to implement.

Disadvantage: 1. The meaning of each characteristic dimension of the principal component is fuzzy to a certain extent, which is not as explanatory as the original sample features. 2. Non-principal components with small variance may also contain important information about sample differences, because dimension reduction discarding may have an impact on subsequent data processing.

LDA pros and cons

Advantage: 1. Prior knowledge of categories can be used. 2. Compared with the ambiguity of PCA, supervised dimension reduction, which measures differences by labels and categories, has a clearer purpose and better reflects the differences between samples.

Disadvantage: 1. LDA is not suitable for dimensionality reduction of non-Gaussian distribution samples. 2. Dimension reduction of LDA can be reduced to k-1 dimension at most. 3. The dimension reduction effect of LDA is poor when the sample classification information depends on variance rather than mean. 4. LDA may overfit data.



5 Question 3

5.1 Establish Two-class Classification Problem

Since the question requires now consider separating the images of digit '5' from the rest (the images of '1' and '8').Note this is now a two-class classification problem. We need to change the value of label. The relative code is as follows:

```
new_label = zeros(size(label));
for i = 1:max(size(new_label))
    if(label(i) == 1)
        new_label(i) = 0;
    end
    if(label(i) == 8)
        new_label(i) = 0;
    end
    if(label(i) == 5)
        new_label(i) = 1;
    end
end
```

Then use SVM with a RBF kernel, SVM with a linear kernel, and a neural network classifier with one hidden layer to classify the dataset in a 5-fold cross validation setting.

Some important codes are shown as below:

Get 5-fold cross validation setting. I used ‘crossvalind’ function to achieve this task. The core function of this function is to generate indices for training and test sets.

As we can see, ‘cvIndices = crossvalind(cvMethod,N,M)’ returns the indices ‘cvIndices’ after applying ‘cvMethod’ on N observations using M as the selection parameter. For ‘cvMethod’ I choose K-fold, since the method uses K-fold cross-validation to generate indices. This method uses M-1 folds for training and the last fold for evaluation. The method repeats this process M times, leaving one different fold for evaluation each time. ‘N’ is the total number of observations or grouping information, specified as a positive integer, vector of positive integers, logical vector, or cell array of character vectors. M is the fold parameter, most commonly known as K in the K-fold cross-validation. M must be a positive integer. The default value is 5.

```
% get 5-fold cross validation setting
k = 5;
% Create indices for the 5-fold cross-validation.
cvIndices = crossvalind('Kfold',new_label,k);
acc_RBF = 0;
acc_linear = 0;
acc_neuralnetwork = 0;
AUC_RBF = 0;
AUC_linear = 0;
AUC_neuralnetwork = 0;
```

5.2 build SVM with an RBF kernel

In order to build SVM with an RBF kernel, I choose to use ‘fitcsvm’ to build the model, then use ‘predict’ function to do the prediction where the input are model, test image and the output are label and score that function predict. Furthermore, use ‘perfcurve’ to get the X and Y of ROC curve and the value of AUC. Finally, compute the average value of accuracy and AUC.

For ‘fitcsvm’, it trains or cross-validates a support vector machine (SVM) model for one-class and two-class (binary) classification on a low-dimensional or moderate-dimensional predictor data set. fitcsvm supports mapping the predictor data using kernel functions, and supports sequential minimal optimization (SMO), iterative single data algorithm (ISDA), or L1 soft-margin minimization via quadratic programming for objective-function minimization.

For ‘predict’, [LABEL, SCORE] = predict(MODEL,X) returns predicted class labels and scores for SVM model MODEL and predictors X. X must be a table if SVM was originally trained on a table, or a numeric matrix if SVM was originally trained on a matrix. If X is a table, it must contain all the predictors used for training this model. If X is a matrix, it must have P columns, where P is the number of predictors used for training this model. Classification labels LABEL have the same type as Y used for training. Scores SCORE are an N-by-K numeric matrix for N observations and K classes. The predicted label is assigned to the class with the largest score.

For ‘perfcurve’, [X,Y,T,AUC] = perfcurve(labels,scores,posclass). It returns the X and Y coordinates of an ROC curve for a vector of classifier predictions, scores, given true class labels, labels, and the positive class label ‘posclass’. I can visualize the

performance curve using plot(X,Y). 'T' returns an array of thresholds on classifier scores for the computed values of X and Y. AUC returns the area under the curve for the computed values of X and Y.

```
% SVM with a RBF kernel
Model_RBF = fitcsvm(train_image, train_label,'KernelFunction','rbf','KernelScale','auto');
[label_RBF, scores_RBF] = predict(Model_RBF, test_image);
[X_RBF, Y_RBF, T_rbf, AUC_rbf] = perfcurve(test_label, scores_RBF(:, 2), 1);
AUC_RBF = AUC_RBF + AUC_rbf;
acc_rbf = sum(label_RBF == test_label)/length(test_label);
acc_RBF = acc_RBF + acc_rbf;
figure(8+i)
hold on
plot(X_RBF, Y_RBF,'r')
```

5.3 build SVM with a Linear kernel

In order to build SVM with an RBF kernel, I choose to use 'fitcsvm' to build the model, then use 'predict' function to do the prediction where the input are model, test image and the output are label and score that function predict. Furthermore, use 'perfcurve' to get the X and Y of ROC curve and the value of AUC. Finally, compute the average value of accuracy and AUC.

For 'fitcsvm', it trains or cross-validates a support vector machine (SVM) model for one-class and two-class (binary) classification on a low-dimensional or moderate-dimensional predictor data set. fitcsvm supports mapping the predictor data using kernel functions, and supports sequential minimal optimization (SMO), iterative single data algorithm (ISDA), or L1 soft-margin minimization via quadratic programming for objective-function minimization.

For 'predict', [LABEL, SCORE] = predict(MODEL,X) returns predicted class labels and scores for SVM model MODEL and predictors X. X must be a table if SVM was originally trained on a table, or a numeric matrix if SVM was originally trained on a matrix. If X is a table, it must contain all the predictors used for training this model. If X is a matrix, it must have P columns, where P is the number of predictors used for training this model. Classification labels LABEL have the same type as Y used for training. Scores SCORE are an N-by-K numeric matrix for N observations and K classes. The predicted label is assigned to the class with the largest score.

For 'perfcurve', [X,Y,T,AUC] = perfcurve(labels,scores,posclass). It returns the X and Y coordinates of an ROC curve for a vector of classifier predictions, scores, given true class labels, labels, and the positive class label 'posclass'. I can visualize the performance curve using plot(X,Y). 'T' returns an array of thresholds on classifier scores for the computed values of X and Y. AUC returns the area under the curve for the computed values of X and Y.

```
% SVM with a linear kernel
model_linear = fitcsvm(train_image, train_label,'KernelFunction','linear','KernelScale','auto');
[label_linear, scores_linear] = predict(model_linear, test_image);
[X_linear, Y_linear, T_linear, AUC_linear] = perfcurve(test_label, scores_linear(:, 2), 1);
AUC_linear = AUC_linear + AUC_linear;
acc_Linear = sum(label_linear == test_label)/length(test_label);
acc_linear = acc_linear + acc_Linear;
plot(X_linear, Y_linear,'g')
```

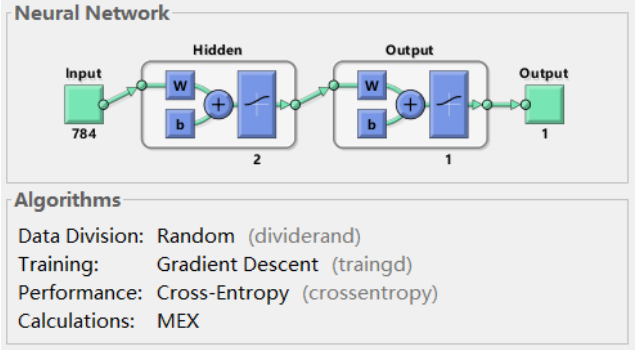
5.4 build neural network classifier with one hidden layer

When I build neural network classifier with one hidden layer, there are some basic operations. From the problem definition it is clear that the MLP should have two input nodes and one output node. I will also create a hidden layer containing two nodes. I can create an MLP in MATLAB with the 'feedforwardnet' command. To create such a network, called net. Then configure the net. Now I have a neural network which has been configured according to the defined problem. It is time to train it. After I trained the net, I can get score and label of net when using the test image for testing. Also use the 'perfcurve' to calculate value of accuracy and AUC and plot the ROC curve. The following is the code I have written.

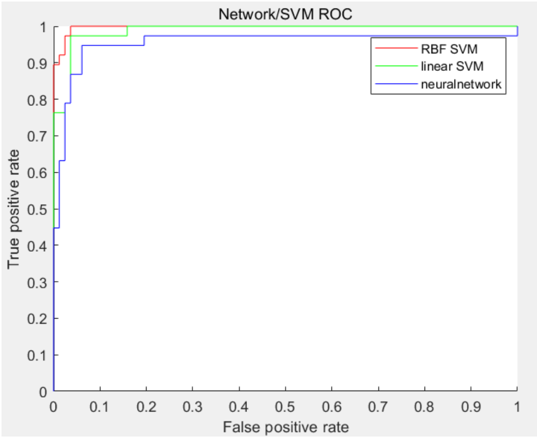
```
% neural network classifier with one hidden layer
net = feedforwardnet(2, 'traingd');
%net.divideParam.trainRatio = 1; % training set [%]
%net.divideParam.valRatio = 0; % validation set [%]
%net.divideParam.testRatio = 0; % test set [%]
% Configure the net
net.inputs{1}.processFcns = {}; % modify the process function for inputs
net.outputs{2}.processFcns = {}; % modify the process function for outputs
net.layers{1}.transferFcn = 'logsig'; % the transfer function for the first layer
net.layers{2}.transferFcn = 'softmax'; % the transfer function for the second layer
net.performFcn = 'crossentropy'; % the loss function
net.trainParam.lr = 0.1; % learning rate.
net.trainParam.epochs = 3000;

net = train(net, train_image', train_label');
score_net = net(test_image');
label_net = (score_net>0.5)';
[X_neuralnetwork,Y_neuralnetwork,T_net,AUC_net] = perfcurve(test_label,score_net,1);
AUC_neuralnetwork = AUC_neuralnetwork + AUC_net;
acc_Neuralnetwork = sum(label_net == test_label)/length(test_label);
acc_neuralnetwork = acc_neuralnetwork + acc_Neuralnetwork;
plot(X_neuralnetwork,Y_neuralnetwork,'b')
xlabel('False positive rate')
ylabel('True positive rate')
title("Network/SVM ROC")
legend('RBF SVM','linear SVM','neuralnetwork')
hold off
```

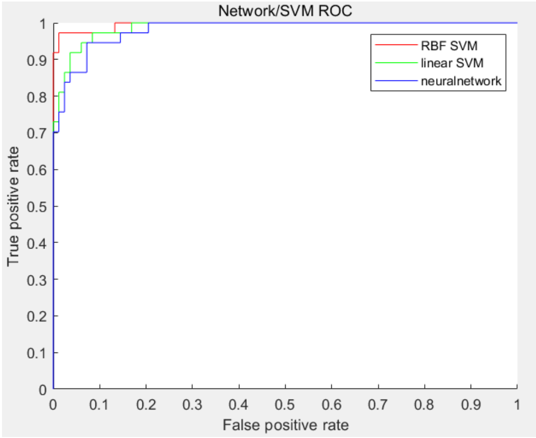
5.5 result of acc, AUC and ROC curve

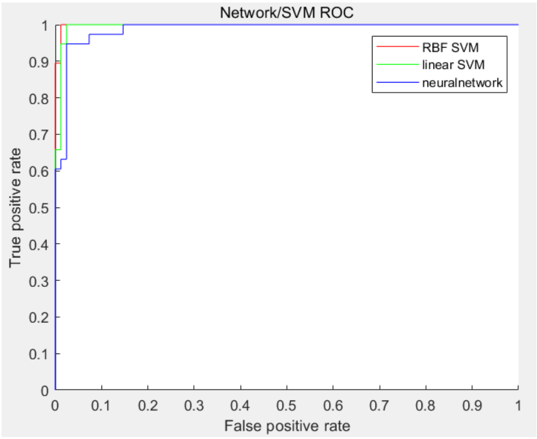
Compare and discuss the results using the obtained validation accuracy. Create and plot the ROC curves of the results using the three different classifiers, and compare their performance using the area under the ROC curve (AUC).

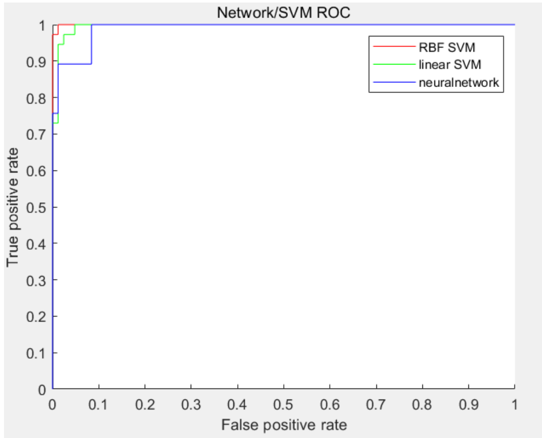


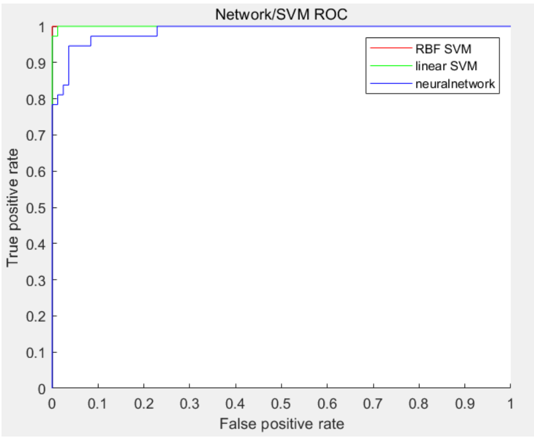
Here are the ROC curve and value of acc and AUC for three methods:












```
acc_RBF =

    0.9850


acc_linear =

    0.9667


acc_neuralnetwork =

    0.9383
```

```
AUC_RBF =

    0.9990


AUC_linear =

    0.9940


AUC_neuralnetwork =

    0.9899
```

The 5-fold ROC curves, average accuracy and average AUC are showed above.

It shows that RBF-SVM is the best effective way to classify our data. Since Kernel function used to compute the elements of the Gram matrix, specified as the comma-separated pair consisting of 'KernelFunction' and a kernel function name RBF is generally used for one-class learning. It may be the best method to compute the accuracy and AUC in these three methods. Of course, both linear-SVM classification methods and neural network classification methods also yielded good results. They are all over 90 percent.

5.6 Parameter Tuning From SVM

When I write the code, I find that for SVM with a linear kernel, the parameter 'KernelScale' is very important. If it is not reasonable, the result will be bad. So I change the parameter 'KernelScale' with different numbers and I can get a very nice accuracy.

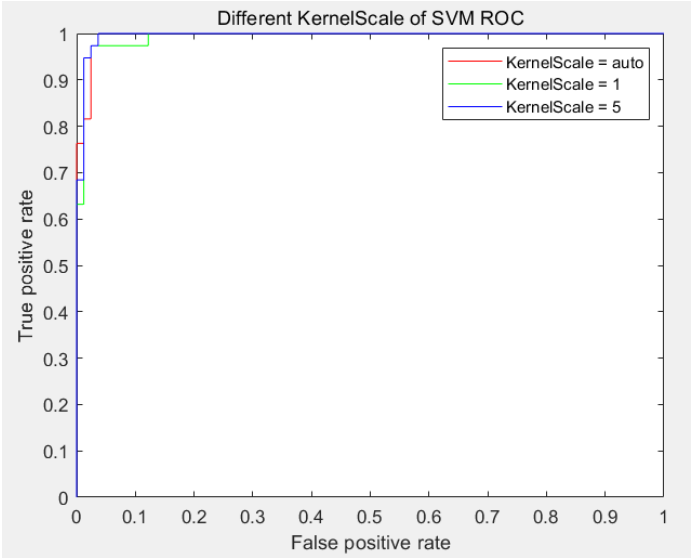
Kernel scale parameter, specified as the comma-separated pair consisting of 'KernelScale' and 'auto' or a positive scalar. The software divides all elements of the predictor matrix X by the value of KernelScale. Then, the software applies the appropriate kernel norm to compute the Gram matrix. If I specify 'auto', then the software selects an appropriate scale factor using a heuristic procedure. This heuristic procedure uses subsampling, so estimates can vary from one call to another. Therefore, to reproduce results, set a random number seed using rng before training.

At this time, SVM has many parameters that can be adjusted, I choose to adjust the 'KernelScale' parameter. I adjust the 'KernelScale' parameters to 'auto', 1 and 5, then find its accuracy and draw the ROC curve.

```
model_linear_auto = fitcsvm(train_image, train_label,'KernelFunction','linear', 'KernelScale','auto');
model_linear_1 = fitcsvm(train_image, train_label,'KernelFunction','linear','KernelScale',1);
model_linear_5 = fitcsvm(train_image, train_label,'KernelFunction','linear', 'KernelScale',5);
```

The following figure is the value of accuracy, AUC and ROC curve.

```
acc_lin_auto =
    0.9667

AUC_linear_auto =
    0.9945

acc_lin_1 =
    0.9583

AUC_linear_1 =
    0.9923

acc_lin_5 =
    0.9750

AUC_linear_5 =
    0.9952
```



For the accuracy of the model, when 'KernelScale' value is 5, it performs the best above three values. For the AUC of the model, when 'KernelScale' value is 5, it performs the best above three values. So when 'KernelScale' value is 5, it can perform the best estimation and prediction.

6 Conclusion

In this assignment, I mainly learn and achieve classification, dimensionality reduction, and clustering. Use MATLAB to program with existing tools for data mining and classification. Discuss machine learning methods and compare performances.

To sum up, PCA, LDA and other dimensionality reduction methods are currently

commonly used machine learning methods. This is a great way to address dimensional causes, visualize, and help interpret high-dimensional data sets and compress the data. Clustering methods such as K-means aim to determine the intrinsic grouping of a set of unlabeled data. This is a measure of allocation of process objects (or data records) into groups or categories (we call clustering) based on some similarity. Hierarchical Clustering is a kind of Clustering algorithm, which creates a hierarchical nested clustering tree by calculating the similarity among data points of different categories. Clustering method has been widely used in computer vision, such as skin color detection.

Support vector machine (SVM) is a set of supervised learning methods and outlier detection for classification and regression. It is valid in higher dimensional Spaces, but if the number of features, avoiding overfitting when selecting kernels, and regularizing terms are crucial. What's more, evaluate their performance is really important. Classification accuracy and area under ROC curve(AUC) and ROC curve are the three main evaluation criteria used.

7 Instructions

There are the instructions on how to run the code. It is very easy.

Just run the Project2_MainCode.m file. All the result for PCA, LDA, K-means, Hierarchical clustering, GMM, SVM, accuracy, AUC and ROC curve will show in the figure from figure 1 to figure 14.

For the method of PCA and LDA are implemented in the PCA_method.m and LDA_method.m file.