Amath482 Homework4

Kexin Jiao

Abstract:

The homework mainly focuses on the practice of Linear discriminant analysis(LDA), Support vector machines(SVM), and decision trees to solve the actual problem: building classifier based on the training data and applying on the test data to see the accuracy.

Introduction and overview:

The goal of the assignment is to build classifier based on the training data, apply on the test data to see the accuracy, and compare the performance between LDA, SVM, and decision trees. To be specific, we are asked to do the following:
-Do an SVD analysis of the digit images. Reshape each image into a column vector and each column of your data matrix is a different image.
-What does the singular value spectrum look like and how many modes are necessary for good image reconstruction? (i.e. rank r of the digit space?)
-What is the interpretation of the U, Σ, and V matrices?
-On a 3D plot, project onto three selected V-modes (columns) colored by their digit label.
-Pick two and three digits. Build linear classifiers to reasonable identify them.
-Which two digits in the data set appear to be the most difficult and easy to separate? Quantify the accuracy of the separation with LDA on the test data.
-How well do SVM and decision tree classifiers separate between all digits?
-Compare the performance between LDA, SVM and decision trees on the hardest and easiest pair of digits to separate (from above).

Theoretical background:

The Singular Value Decomposition
$A = U\Sigma V^*$ gives a decomposition of the matrix A through the unitary matrices U and V and a diagonal matrix Σ. The U and V matrix are rotation matrices while the Σ is the stretching matrix. The values σ on the diagonal of Σ are singular values; the vectors u(columns of U) are left singular vectors; the vectors v(columns of V) are right singular vectors. When handling non-square matrices of arbitrary size, m*n matrix(m>=n), m-n columns that are orthogonal are added to matrix U to make the decomposition work. m-n rows of zeros is added to the matrix Σ as well. However, the Matlab has a built-in function [U,S,V] = svd(A,'econ') to do the singular value decomposition and produces an economy-size decomposition of m*n matrix A.
Computing the SVD

$$A^*A = (U\Sigma V^*) * (U\Sigma V^*)$$
$$= V\Sigma U^*U\Sigma V^* = V\Sigma^2 V^*$$

Multiplying on the right by V gives $A^*AV = V\Sigma^2$ so that the columns of V are the eigenvectors of A^*A with eigenvalues given by the square of the singular values. Multiplying on the right by U gives $A^*AU = U\Sigma^2$ to solve U.

Linear Discriminant Analysis

The goal of LDA is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. When implementing LDA for 2 data sets, the between-class scatter matrix, a measure of the variance between groups(between the means μ_1 and μ_2), is defined by

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

While the within-class scatter matrix, a measure of the variance within each group, is

$$S_w = \sum_{j=1}^{2} \sum_{x} (x - \mu_j)(x - \mu_j)^T$$

A vector w is then

$$w = \operatorname{argmax} \frac{W^T S_B W}{W^T S_W W}$$

The vector w is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem

$$S_B w = \lambda S_w w$$

In matlab, we can just use eig command. When implementing LDS for more groups, the scatter matrix is

$$S_B = \sum_{j=1}^{N} (\mu_j - \mu)(\mu_j - \mu)^T$$

Where μ is the overall mean and μ_j is the mean of each of the N >= 3 groups.

The within-class scatter matrix is

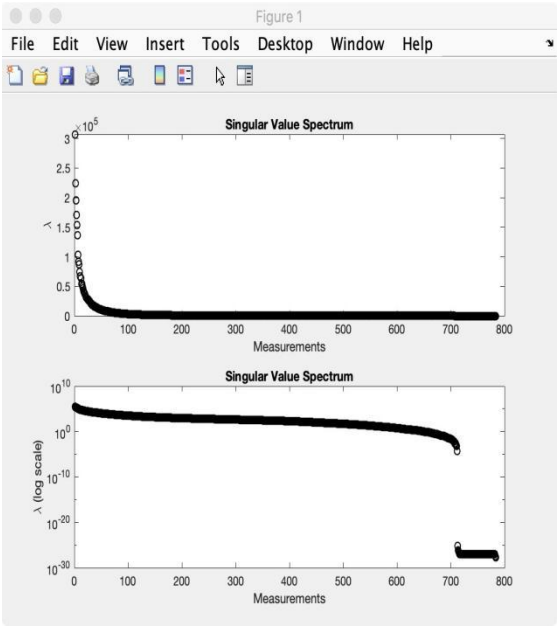
$$S_w = \sum_{j=1}^{N} \sum_{x} (x - \mu_j)(x - \mu_j)^T$$

(Reference: Amath482 lecture notes from Professor Jason Bramburger)

Algorithm implementation and development:

To set up the data, I first loaded the data by minist_parce function, converted the data from unit8 into double type, and reshaped the data from 28*28*60000 into 784*60000. Then I demeaned the data by subtracting the mean of each row with the use of repmat command and applied singular value decomposition to get U,S,V matrices with svd method. I squared the singular values so that the difference can be more stark on the plot. I also set up my threshold at 90% energy to get the number of modes that are necessary for a good image reconstruction. I plotted the singular values in both normal-scale and log-scale. Each pixel(0-9) represents a color. I projected my data on the first three components(columns) and plotted in 3D, using find and plot3 commands. Once I projected my data into PCA space, I built classifiers to identify digits in the training set. To be specific, I made feature at 12 to take for demonstration and projected onto principal components. Picking two digits, I first found them by the find command and calculated the between-class and within-class scatter matrices with for loops. I also found w using eig command with two arguments. I multiplied w' to project onto w. Then I set a threshold for the classifier to have a standard for which one is above or below the threshold. I sorted the elements of the data in a ascending order. To deal with the overlap, I picked a threshold at the midpoint to make the number of errors for both digits approximately the same. Finally, I plotted histograms of the values and draw a vertical line for the decision-making threshold. For three digits, I did the same procedure except for the calculation of the scatter matrices: the μ I used is the overall mean and the μ_j is mean of each N>=3 groups. To find the two digits in the data set that are most difficult and easy to separate, I did the same procedure(as above) to make a model based on the given training data and tested the performance on the testing data. Specifically, I calculated the error and success rate in all combinations of two digits and found the maximum, minimum, and their two digits. In the end, I made the SVM classifier with the use of fitcecoc and predict commands, while I made decision tree classifier with fitctree, kfoldLoss, min, immse, and predict commands to compare the performance among LDA, SVM, and decision trees.

Computational results:



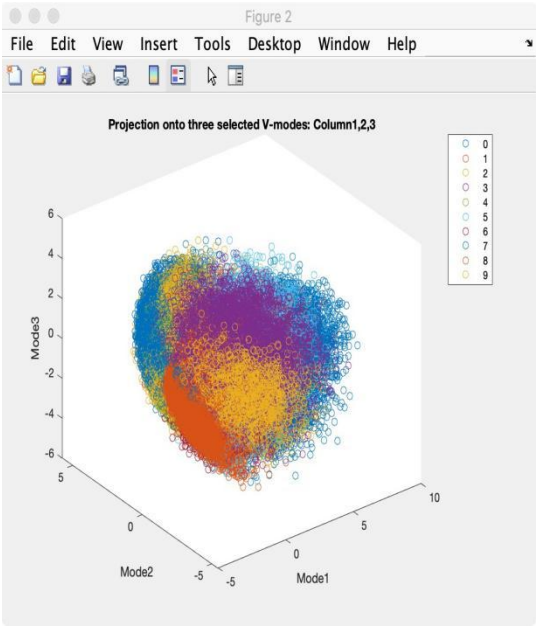
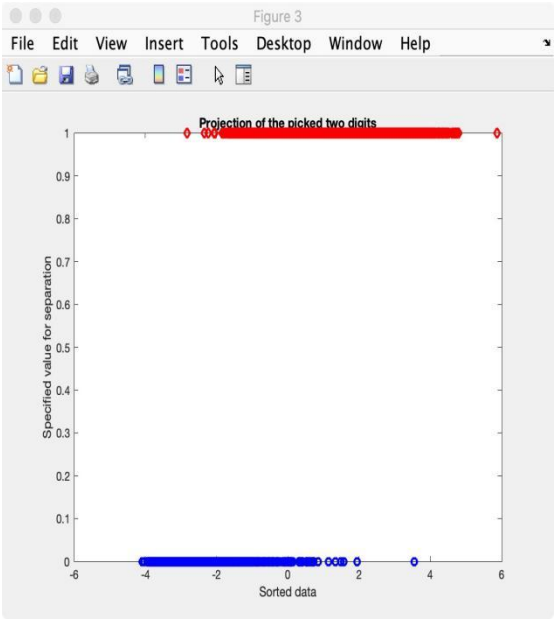


Figure 1(left): This figure displays the plot of the singular values(1) and the plot of its log-scale(2).

Figure 2(right): This plot displays the projection onto the first three columns colored by their digital labels.



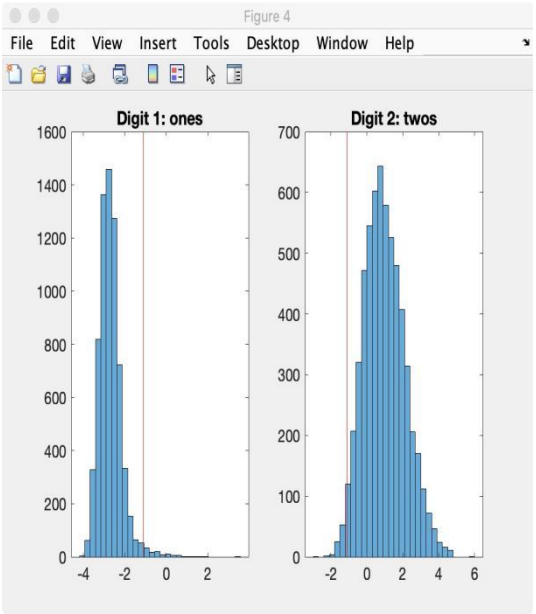


Figure 3(left): This plot displays the separation of the two digits(1,2) I picked, specifying the value at 0 and 1.

Figure 4(right): This plot displays the histogram of the projection of the two digits(1,2) I picked, each with a vertical line as threshold of separation, using LDA.

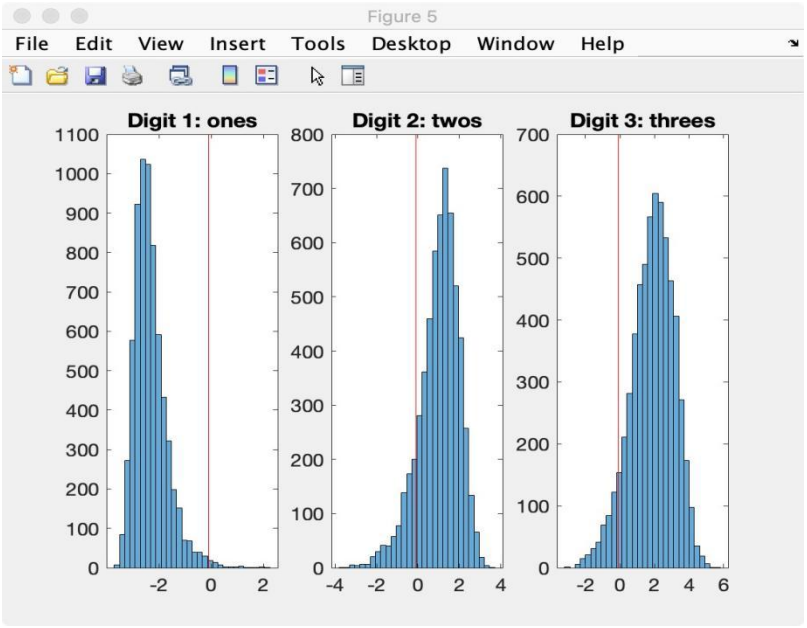


Figure 5: This plot displays the histogram of the projection of the three digits(1,2,3) I picked, each with a vertical line as threshold of separation, using LDA.

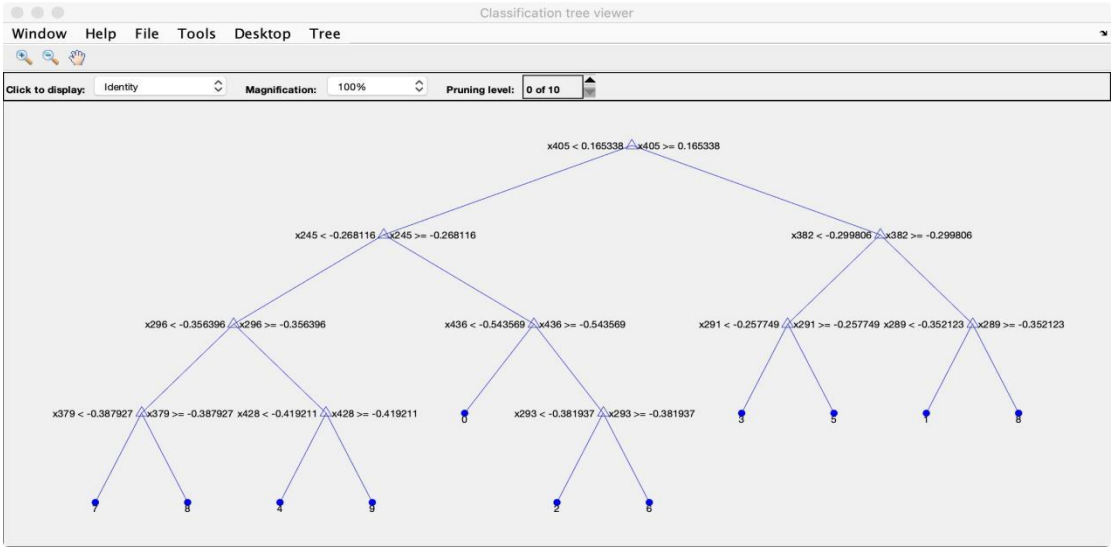


Figure 6: The classification tree viewer with decision trees with overall 41.94% success rate

-According to my calculation, 87 modes(rank 87) are necessary for good image reconstruction, with a threshold setting at 90%.
-After SVD analysis, the matrix U is a 784*784 rotation matrix with orthogonal columns, capturing the modes of each image. The matrix Σ is a 784*784 diagonal stretching matrix with non-negative singular values on the diagonal, telling the importance of each column of U. The matrix V is a 60000*784 orthonormal matrix, representing the linear coefficients of the data.
-The two digits in the data set appear to be most difficult to separate are digits 4 and 7, and the accuracy of the separation with LDA on the test data is 83.17%. The two digits in the data set appear to be easiest to separate are digits 6 and 9, and the accuracy of the separation with LDA on the test data is

97.54%.
-The performance of SVM classifier to separate digits is great, with success rate arriving at 94.46%, while the performance of decision tree classifier is normal, only with 41.94% success rate. To conclude, the LDA and SVM are better at separation than the decision trees. However, the SVM takes much more time than the other two.

Summary and conclusions:

After setting up the data with the use of singular value decomposition(SVD), I built classifier based on the training data and applied on the test data to see the accuracy with the use of Linear discriminant analysis(LDA) and Support vector machines(SVM). Namely, I gave the computer some experience to help it learn the difference. After I made the model, the computer can tell the difference itself.

MATLAB function used and brief implementation explanation:

-size: return the length of the dimension
-find: return indices and values of nonzero elements
-im2double: convert image to double precision
-reshape: reshape matrix A into a sz1by...by szN array
-repmat: repeat copies of array
-[U,S,V] = svd(A,'econ'): perform a singular value decomposition and produce an economy-size decomposition
-diag: get diagonal elements of matrix
-plot: plot multiple XY pairs using the same axes
-plot3: 3D point/line plot
-semilogy: plot data with logarithmic scale for the y axis
-subplot(m,n,p): divide the current figure into m*n grid and create axes in the position p
-histogram: create histogram plot
-eig: returns a column vector containing the generalized eigenvalues of square matrices A and B.
-sort: sort the elements in ascending order
-set: set graphics object properties
-legend: add legend to axes
-fitcecoc: Fit multiclass models for support vector machines or other classifiers
-predit: predict responses of linear regression model
-fitctree: Fit binary decision tree for multiclass classification
-kfoldLoss: Cross-validation loss of partitioned regression model
-immse: calculate the mean-squared error
(Reference: MATLAB search helps)

MATLAB codes:

```
clear all; close all; clc

[images_train, labels_train] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
[images_test, labels_test] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');

images_train = im2double(images_train);
size_train = size(images_train);
images_train = reshape(images_train,size_train(1)*size_train(2),size_train(3));
images_train = images_train-repmat(mean(images_train,2),1,size_train(3));
[U,S,V] = svd(images_train,'econ');

sig = diag(S);
lambda = sig.^2;
threshold = 0.9;
energy = 0;
r = 0;
while energy <= threshold
    r = r+1;
    energy = energy+lambda(r)/sum(lambda);
end
figure(1)
subplot(2,1,1)
plot(lambda,'ko','Linewidth',1)
ylabel('\lambda');
xlabel('Measurements');
title("Singular Value Spectrum");
subplot(2,1,2)
semilogy(lambda,'ko','Linewidth',1)
ylabel('\lambda (log scale)');
xlabel('Measurements');
title("Singular Value Spectrum");

figure(2)
projection = U(:,[1,2,3])'*images_train;
projection0 = projection(:,find(labels_train == 0));
projection1 = projection(:,find(labels_train == 1));
projection2 = projection(:,find(labels_train == 2));
projection3 = projection(:,find(labels_train == 3));
projection4 = projection(:,find(labels_train == 4));
projection5 = projection(:,find(labels_train == 5));
projection6 = projection(:,find(labels_train == 6));
projection7 = projection(:,find(labels_train == 7));
```

```
projection8 = projection(:,find(labels_train == 8));
projection9 = projection(:,find(labels_train == 9));
plot3(projection0(1,:),projection0(2,:),projection0(3,:),'o');hold on
plot3(projection1(1,:),projection1(2,:),projection1(3,:),'o');hold on
plot3(projection2(1,:),projection2(2,:),projection2(3,:),'o');hold on
plot3(projection3(1,:),projection3(2,:),projection3(3,:),'o');hold on
plot3(projection4(1,:),projection4(2,:),projection4(3,:),'o');hold on
plot3(projection5(1,:),projection5(2,:),projection5(3,:),'o');hold on
plot3(projection6(1,:),projection6(2,:),projection6(3,:),'o');hold on
plot3(projection7(1,:),projection7(2,:),projection7(3,:),'o');hold on
plot3(projection8(1,:),projection8(2,:),projection8(3,:),'o');hold on
plot3(projection9(1,:),projection9(2,:),projection9(3,:),'o');
legend('0','1','2','3','4','5','6','7','8','9');
xlabel('Mode1');
ylabel('Mode2');
zlabel('Mode3');
title('Projection onto three selected V-modes: Column1,2,3');

% LDA-2 digits
feature = 12;
images = S*V';
i1 = find(labels_train == 1);
i2 = find(labels_train == 2);
dig1 = images(1:feature,i1);
dig2 = images(1:feature,i2);
num1 = length(i1);
num2 = length(i2);
% num1 = size(i1',2);
% num2 = size(i2',2);
m1 = mean(dig1,2);
m2 = mean(dig2,2);
Sw = 0; % within class variances
for k = 1:num1
    Sw = Sw + (dig1(:,k) - m1)*(dig1(:,k) - m1)';
end
for k = 1:num2
   Sw =   Sw + (dig2(:,k) - m2)*(dig2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)'; % between class

[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda2, ind] = max(abs(diag(D)));
w2 = V2(:,ind);
w2 = w2/norm(w2,2);
```

```
vdig1 = w2'*dig1;
vdig2 = w2'*dig2;

if mean(vdig1) > mean(vdig2)
    w2 = -w2;
    vdig1 = -vdig1;
    vdig2 = -vdig2;
end

figure(3)
plot(vdig1,0,'ob','Linewidth',2)
hold on
plot(vdig2,1,'dr','Linewidth',2)
ylabel('Specified value for separation');
xlabel('Sorted data');
title('Projection of the picked two digits');

sortdig1 = sort(vdig1);
sortdig2 = sort(vdig2);

t1 = length(sortdig1);
t2 = 1;
while sortdig1(t1) > sortdig2(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold = (sortdig1(t1) + sortdig2(t2))/2;

figure(4)
subplot(1,2,1)
histogram(sortdig1,30); hold on, plot([threshold threshold],[0 1600],'r')
set(gca,'Ylim',[0 1600],'Fontsize',14);
title('Digit 1: ones');
subplot(1,2,2)
histogram(sortdig2,30); hold on, plot([threshold threshold],[0 700],'r')
set(gca,'Ylim',[0 700],'Fontsize',14)
title('Digit 2: twos')

% LDA-3 digits
i3 = find(labels_train == 1);
i4 = find(labels_train == 2);
i5 = find(labels_train == 3);
dig3 = images(1:feature,i3);
```

```
dig4 = images(1:feature,i4);
dig5 = images(1:feature,i5);
num3 = length(i3);
num4 = length(i4);
num5 = length(i5);

m3 = mean(dig3,2);
m4 = mean(dig4,2);
m5 = mean(dig5,2);
mj = [m3,m4,m5];
Sw = 0; % within class variances
for k = 1:num3
    Sw = Sw + (dig3(:,k) - m3)*(dig3(:,k) - m3)';
end
for k = 1:num4
   Sw =   Sw + (dig4(:,k) - m4)*(dig4(:,k) - m4)';
end
for k = 1:num5
   Sw =   Sw + (dig5(:,k) - m5)*(dig5(:,k) - m5)';
end

mn = (m3 + m4 + m5)/3;
Sb = (mj(:,1)-mn)*(mj(:,1)-mn)'+(mj(:,2)-mn)*(mj(:,2)-mn)'+(mj(:,3)-mn)*(mj(:,3)-mn)';

[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda3, ind] = max(abs(diag(D)));
w3 = V2(:,ind);
w3 = w3/norm(w3,2);

vdig3 = w3'*dig3;
vdig4 = w3'*dig4;
vdig5 = w3'*dig5;

if mean(vdig3) > mean(vdig4)
    w3 = -w3;
    vdig3 = -vdig3;
    vdig4 = -vdig4;
end

if mean(vdig4) > mean(vdig5)
    w3 = -w3;
    vdig4 = -vdig4;
    vdig5 = -vdig5;
end
```

```
sortdig3 = sort(vdig3);
sortdig4 = sort(vdig4);
sortdig5 = sort(vdig5);

t3 = length(sortdig3);
t4 = 1;
while sortdig3(t3) > sortdig4(t4)
    t3 = t3 - 1;
    t4 = t4 + 1;
end
threshold1 = (sortdig3(t3) + sortdig4(t4))/2;

t3 = length(sortdig3);
t5=1;
while sortdig3(t3) > sortdig5(t5)
    t3 = t3 - 1;
    t5 = t5 + 1;
end
threshold2 = (sortdig3(t3) + sortdig5(t5))/2;
t4 = length(sortdig4);
t5=1;
while sortdig4(t4) > sortdig5(t5)
    t4 = t4 - 1;
    t5 = t5 + 1;
end
threshold3 = (sortdig4(t4) + sortdig5(t5))/2;
thresholdall = (threshold1+threshold2+threshold3)/3;

figure(5)
subplot(1,3,1)
histogram(sortdig3,30); hold on, plot([thresholdall thresholdall],[0,1100],'r')
set(gca,'Ylim',[0 1100],'Fontsize',14)
title('Digit 1: ones')
subplot(1,3,2)
histogram(sortdig4,30); hold on, plot([thresholdall thresholdall],[0,800],'r')
set(gca,'Ylim',[0 800],'Fontsize',14)
title('Digit 2: twos')
subplot(1,3,3)
histogram(sortdig5,30); hold on, plot([thresholdall thresholdall],[0,700],'r')
set(gca,'Ylim',[0 700],'Fontsize',14)
title('Digit 3: threes')

% find most difficult and easy
```

```
images_test = im2double(images_test);
size_test = size(images_test);
images_test = reshape(images_test,size_test(1)*size_test(2),size_test(3));
images_test = images_test-repmat(mean(images_test,2),1,size_test(3));
feature = 12;
nummax = 0;
nummin = 1;
max1 = 0;
max2 = 0;
min1 = 0;
min2 = 0;
for i = 1:9
    for j = (i + 1):10
        index1 = find(labels_train == i-1);
        index2 = find(labels_train == j-1);
        tdig1 = images_train(1:feature,index1);
        tdig2 = images_train(1:feature,index2);
        tnum1 = size(index1',2);
        tnum2 = size(index2',2);
        tm1 = mean(tdig1,2);
        tm2 = mean(tdig2,2);
        tSw = 0; % within class variances
        for k = 1:tnum1
            tSw = tSw + (tdig1(:,k) - tm1)*(tdig1(:,k) - tm1)';
        end
        for k = 1:tnum2
            tSw =   tSw + (tdig2(:,k) - tm2)*(tdig2(:,k) - tm2)';
        end
        tSb = (tm1-tm2)*(tm1-tm2)'; % between class

        [tV2, tD] = eig(tSb,tSw); % linear disciminant analysis
        [tlambda, tind] = max(abs(diag(tD)));
        tw = tV2(:, tind);
        tw = tw/norm(tw,2);

        tvdig1 = tw'*tdig1;
        tvdig2 = tw'*tdig2;

        if mean(tvdig1) > mean(tvdig2)
            tw = -tw;
            tvdig1 = -tvdig1;
            tvdig2 = -tvdig2;
        end
```

```
        tsortdig1 = sort(tvdig1);
        tsortdig2 = sort(tvdig2);

        tt1 = length(tsortdig1);
        tt2 = 1;
        while tsortdig1(tt1) > tsortdig2(tt2)
            tt1 = tt1 - 1;
            tt2 = tt2 + 1;
        end
        tthreshold = (tsortdig1(tt1) + tsortdig2(tt2))/2;

        TestNum = size(images_test,2);
        TestMat = U(:, 1:feature)'*images_test; % PCA projection
        pval = tw'*TestMat;
        i1test = find(labels_test == i-1);
        i2test = find(labels_test == j-1);
        ResVec1 = (pval(i1test) < threshold);
        err1 = abs(sum(ResVec1) - size(i1test',2));
        ResVec2 = (pval(i2test) > threshold);
        err2 = abs(sum(ResVec2) - size(i2test',2));
        sucRate = 1 - (err1+err2)/TestNum;
        if sucRate > nummax
            nummax = sucRate;
            max1 = i-1;
            max2 = j-1;
        end
        if sucRate < nummin
            nummin = sucRate;
            min1 = i-1;
            min2 = j-1;
        end
    end
end

success_svm = 0;
Mdl = fitcecoc(images_train',labels_train);
test_labels = predict(Mdl,images_test');
for j = 1:length(test_labels)
    if (test_labels(j) - labels_test(j)) == 0
        success_svm = success_svm + 1;
    end
end
sucRate_svm = success_svm/length(test_labels);
```

```
tree = fitctree(images_train',labels_train, 'MaxNumSplits',10,'CrossVal','on');
view(tree.Trained{1},'Mode','graph');
classError = kfoldLoss(tree, 'mode', 'individual');
[~, k] = min(classError);
testpredict = predict(tree.Trained{k}, images_test');
errortree = immse(testpredict, labels_test);
diff = testpredict - labels_test;
success = find(diff == 0);
sucRate_tree = length(success) / length(labels_test);
```