Classifying Digits

Haoran Sun

March 8, 2021

Abstract

In this assignment,we will be recognizing digits from a set of images by building a Linear classifier(LDA). We will start by choosing two digits and identify them at each time. We will also try to identify three digits using LDA and compare its performance with two-digit case. Finally, we will apply SVM(support vector machines) and decision trees and see how they perform compared to LDA.

1 Introduction and Overview

In this task, we seek to develop a classifier that can identify digits from images using Linear Discriminant Analysis(LDA). We are given 60,000 images of digits in 28*28 pixels, with each image labeled by its number. We will reduce their dimensions by using PCA and use them as the training data for our classifier. Then, we will test our classifier on another set of 10,000 images of digits and see how it performs. All the data are from MNIST data set. We will start by distinguish a pair of digits, and then try to build a linear classifier for 3 digits.

For comparison, we will also use SVM(support vector machines) and tree classifiers, which were used prevalently before for classifying.

2 Theoretical Background

2.1 Linear Discriminant Analysis(LDA)

The purpose of the Linear Discriminant Analysis is to project the classes of data onto a basis, which maximizes the distance between data for different classes, and minimizes the distance between the data in same class. In another word, LDA separates the classes from each other on the new basis and each classes are tightened together. Figure 1 is the visualization of the goal of LDA. In (a), the data from two classes are blended together on the new projection, and there is no way to distinguish them. By contrast, the projection in (b) successfully maximized the distance between the two classes.

How do we find out what this best projection is?

2.1.1 LDA for 2 data sets

For finding out the ideal projection for 2 data sets, we first calculate the means for each groups for each feature(which is the principal components we will be using in this case). We denote them as μ_1, μ_2. then the **between-class scatter matrix** is defined as:

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{1}$$

S_B measures the variance between the means of each groups(and therefore between the groups). For measuring the variance within each group, the **within-class scatter matrix** is defined as

$$\mathbf{S}_w = \sum_{j=1}^{2} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{2}$$

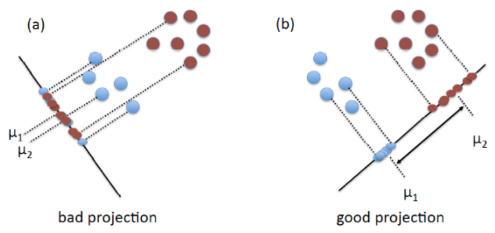


Figure 1: Visualization of LDA Projection

As discussed previously, we want to maximize the variance between classes and minimize the variance within each class. Mathematically, it is given by:

$$\mathbf{w} = \text{argmax}\frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_w\mathbf{w}} \tag{3}$$

where the vector $\mathbf{w}$ maximizes this fraction. This simplifies to be a eigenvalue problem:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_w\mathbf{w} \tag{4}$$

After projecting data using this vector and separating classes, a threshold can be determined between classes for decision making.

2.1.2 LDA for multiple data sets

Similar to the case in 2 data sets, the between-class scatter matrix for multiple classes are defined as:

$$\mathbf{S}_B = \sum_{j=1}^{N}(\mu_j - \mu)(\mu_j - \mu)^T \tag{5}$$

where μ is the overall mean. The within-class scatter matrix for multiple classes are defined as:

$$\mathbf{S}_w = \sum_{j=1}^{N}\sum_{x}(x - \mu_j)(x - \mu_j)^T \tag{6}$$

N is the number of classes in both equations.

3 Algorithm Implementation and Development

3.1 PCA

Before starting to classify the digits, we will first project the data into its PCA space. The initial data is stored in the matrix of size 28*28*60000. We reshape it into a 782*60000 matrix, where each column represents the pixels of each different image. We then apply reduced SVD on this matrix(before doing this, we will subtract the mean row-wisely to demean our matrix).

The obtained U matrix contains the principal components of our data on each column. And the row of V is the coefficients of each image. The singular values on the diagonal of matrix S determines how important the corresponding principal component is. After obtaining S,U,V matrices of our demeaned data matrix, we will:

1. Plot the singular value spectrum and analyze how many modes are needed for a good data reconstruction

2. Plot three randomly-chosen principal components of each digits,colored by their digital labels in one 3D plot.

3. Project our data into its low-rank PCA space, which is given by **U(:,1:feature)'X**. **X** is the original data matrix.

3.2 Build LDA Classifier

After Project our data into its low-rank PCA space, we will extract all the data corresponding to its digit separately and put them into their corresponding digit data matrix. By doing this, we can choose the two digits we decide to classify later. The algorithm for building classifier and training data is detailed in Algorithm 1.

Algorithm 1: LDA

```
    Choose two digits we decide to classify on and use their data matrix var_1,var_2 as our data sets
    Calculate the best projection line w by applying the theory in section 2.1.1
    Project the two data sets by doing w' * var_1 and w' * var_2
    Make consistency that the first data set is always below the threshold
    Set up the threshold, which will be the midpoint with same number of data of each group on each side
```

The process for building the classifier for 3 digits are similar to Algorithm 1. However, when we have more than two digits, we cannot make consistency of the order of the data sets. Therefore, the code need to be adjusted accordingly with different three data sets chosen. For three digits, there will be two thresholds to separate them.

3.3 Test LDA Classifier

After we determined the best projection line of the data and the thresholds from training the classifier, we can use it to classify our test data. Similar to the training data, we reshape the test data, where each column of the test data matrix is a different test image. Then:

1. Subtract the mean from the test data set(the mean here is the mean of training data set)

2. Apply SVD of the demeaned test data set and project it into its low-rank principal components

3. Put the data of each digits into matrices separately so that we can choose which two digits we decide to test on.

4. Project the two test data sets onto the best projection line `w` obtained from training LDA.

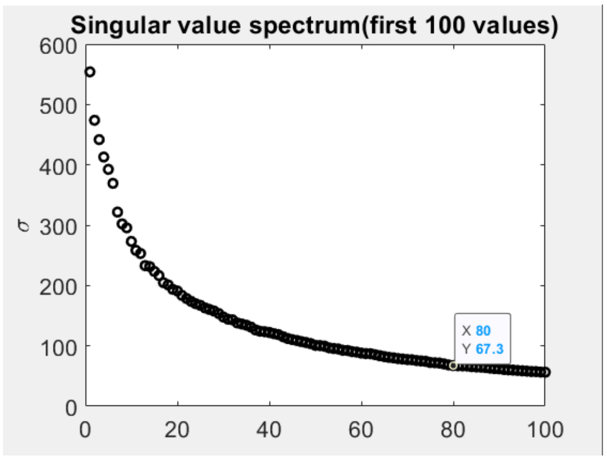
5. Make consistency that the first test data set is always below the threshold(as we did in Algorithm 1.

In the end, we can see the accuracy of the classifier by looking at how many data in the first data set(which is supposed to be below threshold) is beyond the threshold, and opposite for the second set.

4 Computational Results

4.1 PCA

The singular value spectrum for first 100 singular values, and the cumulative energy plot with respect to the first 200 singular value are shown in Figure 2 and 3, respectively. It is obvious that a very heavy-tail distribution is shown in Figure 2. The 80th singular values still has a value over 60.It means the first few singular values do not dominate the system, and lots of information is in the later modes. From Figure 3, we can tell about 100 modes are necessary to reconstruct 90% of the total energy of the original data.



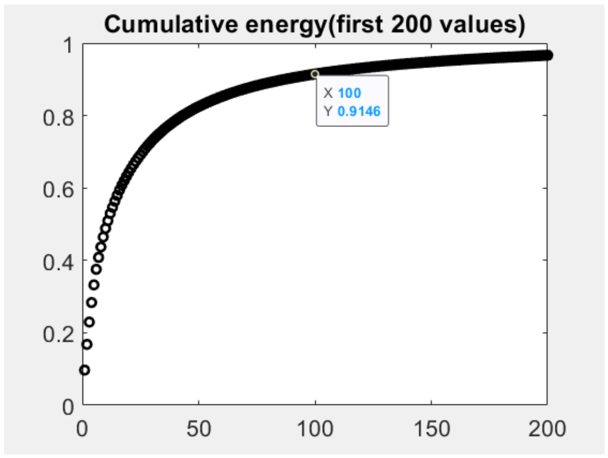


Figure 2: The first 100 singular values(Left)

Figure 3: Cumulative energy for first 200 singular values (Right)

However, it turns out we do not need to restore this much of the energy to get our classifier to work, which means we can only use less than 100 modes. Figure 4 shows the approximation of low ranks of one randomly chosen digit from our data set. It is noticeable that when the image has a rank of 20, its characteristics has already been distinguishable and we can tell this digit is 9. Therefore, we will use 20 as our feature.

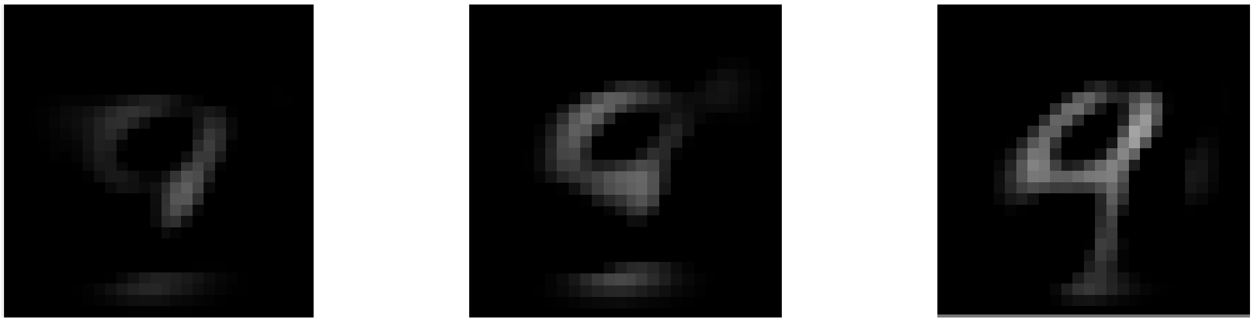


Figure 4: Rank 3, Rank 10 and Rank 20 approximation of one image

The 3D plot of the projection onto three selected V-modes of each digit are shown in Figure 5. Although the projection of each digits overlap by some amount, it is clear the projections between digits are not completely mixed up.

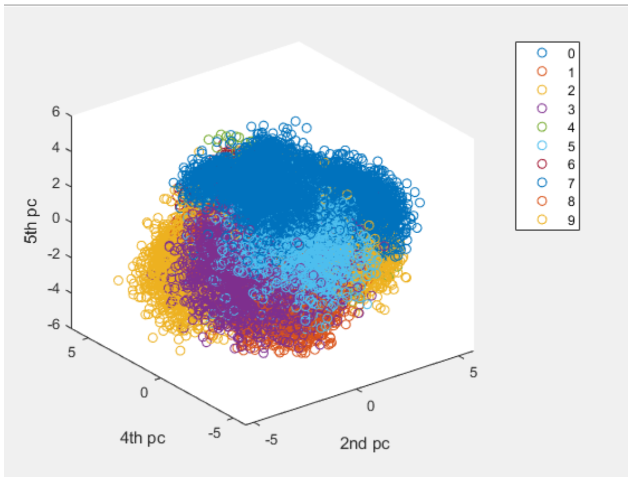
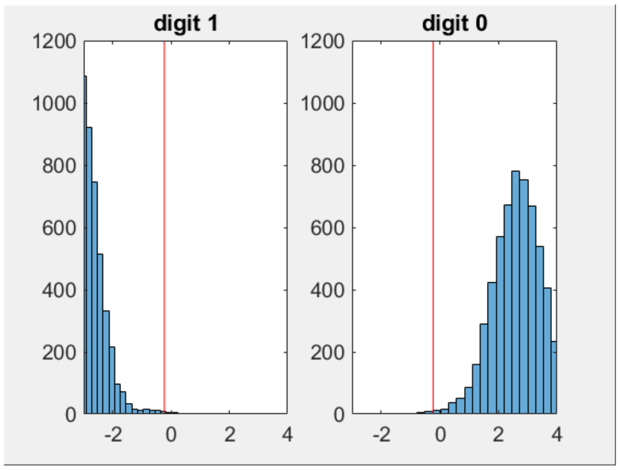


Figure 5: 3d plot for 2nd, 4th, 5th PC

4.2 Identifying digits using LDA

4.2.1 Two digits

After building the classifier and applying it to distinguish all the possible pairs of digits, The pair(0,1) appears to be the most easy to separate, with the accuracy of 99.67% digit 1 being recognized and 99.63% digit 0 being recognized correctly in training set. In the test set 99.91% digit 1 and 99.69% digit 0 being recognized. Those stats are astonishingly satisfying. The most difficult pair to separate is (5,3). 92.46% digit 5 and 93.30% digit 3 are recognized in training set, and 92.83% digit 5 and 96.34% digit 3 are recognized in test set. Although it is not as accurate as for digit 0 and 1, the accuracy is still decent. The histogram of the data values of both pairs in training setsare plotted in Figure 5 and Figure 6. The red line is the threshold for each pair. It is also clear visually, that digit 1 and 0 are separated distantly, and digit 5 and 3 are closer compared to the first pair.



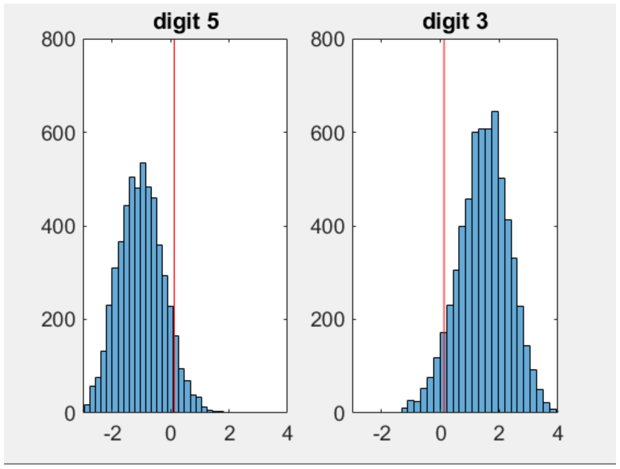


Figure 6: Histograms of digit 1 and 0 (Left)

Figure 7: Histograms of digit 5 and 3 (Right)

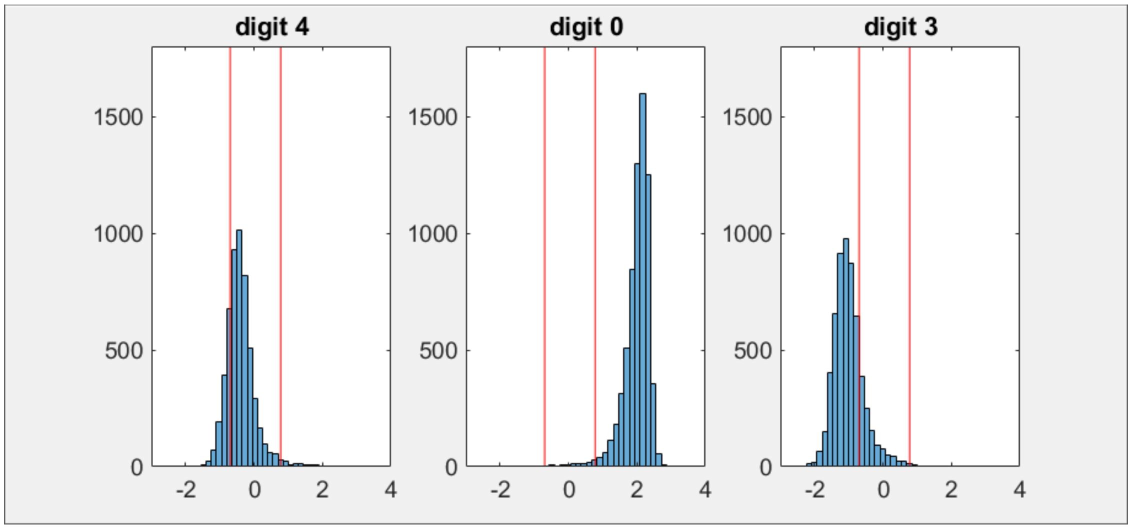


Figure 8: Histograms of digit 4, 0 and 3

4.2.2 Three digits

As mentioned in section 3.2, we cannot make consistency of the order of the data sets if we have more than two sets, which means we need to change the code every time we test different groups of digits data to fit the situation. In this section, we will only look at group (4,0,3), which is randomly chosen. It turns out 77.38% of digit 4, 98.44% of digit 0 and 78.48% of digit 3 in training sets are accurately recognized. Similar results are obtained for test sets. Although the accuracy for identifying 0 is high, the rest two digits does not obtain a high accuracy. The histogram is shown in Figure 7. The red vertical lines are the two thresholds.

4.3 SVM and Tree Classifiers

Now we will see how SVM and Tree Classifier performs on these sets of data. We will first use them to separate all 10 digits. The error of SVM is 0.0578, and the error of Tree Classifiers are 0.1645. In both classifiers, we used 20 features, which is the same as is LDA classifier. Then we tried to apply them for only recognizing the easiest and hardest pairs of digits. For the most difficult pair (5,3), the error of SVM is 0.0012, and the error of Tree Classifiers is 0.0053. Both performs better than LDA does on this hardest pair. While for the easiest pair (1,0), SVM has an error of 6.3166e-04, and the tree classifiers has an error of 0.0017. Still, both does great than SVM.

5 Summary and Conclusions

Using LDA, we acquired very high accuracy for identifying 2 digits, with over 90 % successful rate even for the hardest pair of digits. However, using LDA for recognizing 3 digits does not work very accurately(yet it still has an accuracy over 70%). We compare the results for 2 digits in LDA with the results obtained from SVM and Tree Classifiers. The later two did better than LDA with a higher accuracy. However, the accuracy of LDA is still very satisfying. While for identifying 3 digits, there might be potential methods and algorithms to improve the performance of LDA, which we could be looking at.

Appendix A MATLAB Functions

- `sort` sorts in ascending order.
- `M = max(X)` returns the largest element of vector `X`.

- `find` Find indices of nonzero elements.
- `[U,S,V] = svd(X)` produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that X = U*S*V'.
- `diag(V)` puts V on the main diagonal
- `histogram(X)` plots a histogram of X.
- `[V,D] = eig(A,B)` produces a diagonal matrix D of generalized eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = B*V*D.
- `fitcecoc` Fit a multiclass model for Support Vector Machine or other classifiers.
- `fitctree` Fit a classification decision tree.

Appendix B MATLAB Code

```
%% clear
clear;close all; clc

%% load data
[images, labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte'); %Traning data
[images2,labels2] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte'); %Test data

%% Reshape & Reduced SVD
data =im2double(reshape(images,784,60000));
data2 = im2double(reshape(images2,784,10000));
[m,n]=size(data); % compute data size
mn=mean(data,2); % compute mean for each row
data=data-repmat(mn,1,n); % subtract mean
data2 = data2 - repmat(mn,1,10000);
[U,S,V] = svd(data,'econ'); % reduced SVD
%% Singular value spectrum
plot(diag(S),'ko','Linewidth',2);
set(gca,'Fontsize',16,'Xlim',[0 100]);
title('Singular value spectrum(first 100 values)');
ylabel('\sigma')
%% energy
% plot(cumsum(diag(S).^2)/sum(diag(S).^2),'ko','Linewidth',2);
% set(gca,'Fontsize',16,'Xlim',[0 200]);
% title('Cumulative energy(first 200 values)');
%% low-rank approximation
% k = 20;
% data_approx = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'; %'// rank-15 approximation
% image_approx = reshape(data_approx(:,5),28,28);
% imshow(image_approx);

%% 3d plot of three selected principal components
Y_3mode = U(:,[2,4,5])'*data;

for k = 0:9
    data_3mode = Y_3mode(:,find(labels == k));
    plot3(data_3mode(1,:),data_3mode(2,:),data_3mode(3,:),'o');hold on
end
xlabel('2nd pc'); ylabel('4th pc'); zlabel('5th pc')
legend('0','1','2','3','4','5','6','7','8','9');

%% Create low-dimension matrix for each digit in training/test set in PC
%Traning set
feature = 20;
Y = U(:,1:feature)'*data;
data_0 = Y(:,find(labels == 0));
data_1 = Y(:,find(labels == 1));
data_2 = Y(:,find(labels == 2));
data_3 = Y(:,find(labels == 3));
data_4 = Y(:,find(labels == 4));
data_5 = Y(:,find(labels == 5));
data_6 = Y(:,find(labels == 6));
data_7 = Y(:,find(labels == 7));
```

```
data_8 = Y(:,find(labels == 8));
data_9 = Y(:,find(labels == 9));
data_cell= {data_0,data_1,data_2,data_3,data_4,data_5,data_6,data_7,data_8,data_9};

%Test set
Y2 = U(:,1:feature)'*data2;
data2_0 = Y2(:,find(labels2 == 0));
data2_1 = Y2(:,find(labels2 == 1));
data2_2 = Y2(:,find(labels2 == 2));
data2_3 = Y2(:,find(labels2 == 3));
data2_4 = Y2(:,find(labels2 == 4));
data2_5 = Y2(:,find(labels2 == 5));
data2_6 = Y2(:,find(labels2 == 6));
data2_7 = Y2(:,find(labels2 == 7));
data2_8 = Y2(:,find(labels2 == 8));
data2_9 = Y2(:,find(labels2 == 9));
data2_cell= {data2_0,data2_1,data2_2,data2_3,data2_4,data2_5,data2_6,data2_7,data2_8,data2_9};

%% the two digits we choose in traning set
digit_1 = 1;
digit_2 = 0;
%the data of the chosen 2 digits in training set.
var_1 = data_cell{digit_1+1};
var_2 = data_cell{digit_2+1};

var2_1 = data2_cell{digit_1+1};
var2_2 = data2_cell{digit_2+1};


%% Calculate scatter matrices
m1 = mean(var_1,2);
m2 = mean(var_2,2);

Sw = 0; % within class variances
for k = 1:size(var_1,2);
    Sw = Sw + (var_1(:,k) - m1)*(var_1(:,k) - m1)';
end
for k = 1:size(var_2,2);
   Sw =  Sw + (var_2(:,k) - m2)*(var_2(:,k) - m2)';
end

Sb = (m1-m2)*(m1-m2)'; % between class
%% Find the best projection line
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);

%% Project onto w
v_var1 = w'*var_1;
v_var2 = w'*var_2;

%for test data
v2_var1 = w'*var2_1;
```

```
v2_var2 = w'*var2_2;


%% Make var2 above the threshold

if mean(v_var1) > mean(v_var2)
    w = -w;
    v_var1 = -v_var1;
    v_var2 = -v_var2;
    v2_var1 = -v2_var1;
    v2_var2 = -v2_var2;

end

%% Find the threshold value

sort1 = sort(v_var1);
sort2 = sort(v_var2);

t1 = length(sort1);
t2 = 1;
while sort1(t1) > sort2(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold = (sort1(t1) + sort2(t2))/2;

%% Plot histogram of results
hold off
subplot(1,2,1)
histogram(sort1,30); hold on, plot([threshold threshold], [0 800],'r')
set(gca,'Xlim',[-3 4],'Ylim',[0 800],'Fontsize',14)
title('digit 5')
subplot(1,2,2)
histogram(sort2,30); hold on, plot([threshold threshold], [0 800],'r')
set(gca,'Xlim',[-3 4],'Ylim',[0 800],'Fontsize',14)
title('digit 3')
%% Test accuracy between 2 digits in training data.
ResVec_1 = (v_var1 < threshold);
success_1 = sum(ResVec_1);
err_1 = length(v_var1) - success_1;
sucRate_1 = success_1 / length(v_var1)

ResVec_2 = (v_var2 > threshold);
success_2 = sum(ResVec_2);
err_2 = length(v_var2) - success_2;
sucRate_2 = success_2 / length(v_var2)

% Accuracy between 2 digits in test set

ResVec2_1 = (v2_var1 < threshold);
success2_1 = sum(ResVec2_1);
err2_1 = length(v2_var1) - success2_1;
sucRate2_1 = success2_1 / length(v2_var1)
```

```
ResVec2_2 = (v2_var2 > threshold);
success2_2 = sum(ResVec2_2);
err2_2 = length(v2_var2) - success2_2;
sucRate2_2 = success2_2 / length(v2_var2);

%% SVM
data_00 = var_1./max(var_1(:));
data_11 = var_2./max(var_2(:));
ones1 = ones(length(sort1),1);
ones2 = ones(length(sort2),1);
label0 = digit_1.*ones1;
label1 = digit_2.*ones2;
xtrain = [data_00 data_11];
xtrain = xtrain.';
xlabel = [label0;label1];
Mdl = fitcecoc(xtrain,xlabel);
error = resubLoss(Mdl)

%% SVM test
xtest = Y2./max(Y2(:));
xtest = xtest';
test_labels = predict(Mdl,xtest);
CVSVMModel = crossval(Mdl);
%%
classLoss = kfoldLoss(CVSVMModel)
%% fitctree
MdlDefault = fitctree(xtrain,xlabel);

%% fitctree error
result = predict(MdlDefault, xtest);
```

Code for LDA for two digits

```
% LDA for three digits
%% clear
clear;close all; clc

%% load data
[images, labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte'); %Traning data
[images2,labels2] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte'); %Test data


%% Reshape & Reduced SVD
data =im2double(reshape(images,784,60000));
data2 = im2double(reshape(images2,784,10000));
[m,n]=size(data); % compute data size
mn=mean(data,2); % compute mean for each row
data=data-repmat(mn,1,n); % subtract mean
data2 = data2 - repmat(mn,1,10000);
[U,S,V] = svd(data,'econ'); % reduced SVD
%% Singular value spectrum
plot(diag(S),'ko','Linewidth',2);
set(gca,'Fontsize',16,'Xlim',[0 100]);
title('Singular value spectrum(first 100 values)');
```

```
ylabel('\sigma')
%% energy
plot(cumsum(diag(S).^2)/sum(diag(S).^2),'ko','Linewidth',2);
set(gca,'Fontsize',16,'Xlim',[0 200]);
title('Cumulative energy(first 200 values)');
%% low-rank approximation
k = 15;
data_approx = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'; %'// rank-15 approximation
aaa = reshape(data_approx(:,5),28,28);
imshow(aaa);

%% 3d plot of three selected principal components
Y_3mode = U(:,[2,4,5])'*data;

for k = 0:9
    data_3mode = Y_3mode(:,find(labels == k));
    plot3(data_3mode(1,:),data_3mode(2,:),data_3mode(3,:),'o');hold on
end
xlabel('2nd pc'); ylabel('4th pc'); zlabel('5th pc')
legend('0','1','2','3','4','5','6','7','8','9');

%% Create low-dimension matrix for each digit in training/test set in PC
%Traning set
feature = 100;
Y = U(:,1:feature)'*data;
data_0 = Y(:,find(labels == 0));
data_1 = Y(:,find(labels == 1));
data_2 = Y(:,find(labels == 2));
data_3 = Y(:,find(labels == 3));
data_4 = Y(:,find(labels == 4));
data_5 = Y(:,find(labels == 5));
data_6 = Y(:,find(labels == 6));
data_7 = Y(:,find(labels == 7));
data_8 = Y(:,find(labels == 8));
data_9 = Y(:,find(labels == 9));
data_cell= {data_1,data_2,data_3,data_4,data_5,data_6,data_7,data_8,data_9};

%Test set
Y2 = U(:,1:feature)'*data2;
data2_0 = Y2(:,find(labels2 == 0));
data2_1 = Y2(:,find(labels2 == 1));
data2_2 = Y2(:,find(labels2 == 2));
data2_3 = Y2(:,find(labels2 == 3));
data2_4 = Y2(:,find(labels2 == 4));
data2_5 = Y2(:,find(labels2 == 5));
data2_6 = Y2(:,find(labels2 == 6));
data2_7 = Y2(:,find(labels2 == 7));
data2_8 = Y2(:,find(labels2 == 8));
data2_9 = Y2(:,find(labels2 == 9));
data2_cell= {data2_1,data2_2,data2_3,data2_4,data2_5,data2_6,data2_7,data2_8,data2_9};

%% the three digits we choose in traning set
digit_1 = 4;
digit_2 = 0;
```

```
digit_3 = 3;
%the data of the chosen 2 digits in training set.
var_1 = data_cell{digit_1+1};
var_2 = data_cell{digit_2+1};
var_3 = data_cell{digit_3+1};

var2_1 = data2_cell{digit_1+1};
var2_2 = data2_cell{digit_2+1};
var2_3 = data2_cell{digit_3+1};



%% Calculate scatter matrices
m1 = mean(var_1,2);
m2 = mean(var_2,2);
m3 = mean(var_3,2);
m_all = (m1+m2+m3)./3;

Sw = 0; % within class variances
for k = 1:size(var_1,2);
    Sw = Sw + (var_1(:,k) - m1)*(var_1(:,k) - m1)';
end
for k = 1:size(var_2,2);
   Sw =  Sw + (var_2(:,k) - m2)*(var_2(:,k) - m2)';
end
for k = 1:size(var_3,2);
   Sw =  Sw + (var_3(:,k) - m3)*(var_3(:,k) - m3)';
end

Sb = (m1-m_all)*(m1-m_all)'+(m2-m_all)*(m2-m_all)'+(m3-m_all)*(m3-m_all)'; % between class
%% Find the best projection line
[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);

%% Project onto w
v_var1 = w'*var_1;
v_var2 = w'*var_2;
v_var3 = w'*var_3;

%for test data
v2_var1 = w'*var2_1;
v2_var2 = w'*var2_2;
v2_var3 = w'*var2_3;



%% compare the mean
a1 = mean(v_var1);
a2 = mean(v_var2);a3 = mean(v_var3);


%% Find the threshold value
```

```
sort1 = sort(v_var1);
sort2 = sort(v_var2);
sort3 = sort(v_var3);

t3 = length(sort3);
t1 = 1;
while sort3(t3) > sort1(t1)
    t3 = t3 - 1;
    t1 = t1 + 1;
end
threshold1 = (sort3(t3) + sort1(t1))/2;

t1 = length(sort1);
t2 = 1;
while sort1(t1) > sort2(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold2 = (sort2(t2) + sort1(t1))/2;
%% Plot histogram of results
hold off
subplot(1,3,1)
histogram(sort1,30); hold on, plot([threshold1 threshold1], [0 1800],'r')
plot([threshold2 threshold2], [0 1800],'r')
set(gca,'Xlim',[-3 4],'Ylim',[0,1800],'Fontsize',14)
title('digit 4')
subplot(1,3,2)
histogram(sort2,30); hold on,
plot([threshold1 threshold1], [0 1800],'r')
plot([threshold2 threshold2], [0 1800],'r')
set(gca,'Xlim',[-3 4],'Ylim',[0,1800],'Fontsize',14)
title('digit 0')
subplot(1,3,3)
histogram(sort3,30); hold on,
plot([threshold1 threshold1], [0 1800],'r')
plot([threshold2 threshold2], [0 1800],'r')
set(gca,'Xlim',[-3 4],'Ylim',[0,1800],'Fontsize',14)
title('digit 3')
%% Test accuracy between 3 digits in training data.
ResVec_1 = (v_var1 < threshold2 & v_var1 >threshold1);
success_1 = sum(ResVec_1);
err_1 = length(v_var1) - success_1;
sucRate_1 = success_1 / length(v_var1)

ResVec_2 = (v_var2 > threshold2);
success_2 = sum(ResVec_2);
err_2 = length(v_var2) - success_2;
sucRate_2 = success_2 / length(v_var2)

ResVec_3 = (v_var3 < threshold1);
success_3 = sum(ResVec_3);
err_3 = length(v_var3) - success_3;
sucRate_3 = success_3 / length(v_var3)
```

```
%% Accuracy between 3 digits in test set
ResVec2_1 = (v2_var1 < threshold2 &v2_var1 > threshold1);
success2_1 = sum(ResVec2_1);
err2_1 = length(v2_var1) - success2_1;
sucRate2_1 = success2_1 / length(v2_var1)

ResVec2_2 = (v2_var2 > threshold2);
success2_2 = sum(ResVec2_2);
err2_2 = length(v2_var2) - success2_2;
sucRate2_2 = success2_2 / length(v2_var2)

ResVec2_3 = (v2_var3 < threshold1);
success2_3 = sum(ResVec2_3);
err2_3 = length(v2_var3) - success2_3;
sucRate2_3 = success2_3 / length(v2_var3)
```

Code for LDA for Three digits