AMATH482 Homework 4

Vandy Zhang
March 8, 2021

Abstract

This assignment is to build classifiers for images based on the training data provided and compare the predictions to the test data to check its performance in the context of machine learning.

1 Introduction and Overview

For this homework, I was provided with a training data consisted of 60000 images, each with $28 * 28$ pixels and a data of labels of the image pixels. Each pixel has a number ranging from 0 to 9 representing an indicator of color. For the first part, I was asked to apply the singular value decomposition (SVD) to the digital images and look for the important principal components, which are important for image reconstruction. After that, I should project the data onto three randomly-selected modes on a 3D basis.

For the second part, I was asked to apply the linear discriminant analysis to separate two or three digits in sequence. To find out which pair of digits are the most and least difficult to separate, I was also supposed to build a nested for loop to pair the digits of the pixels up. To check the performance of the classifier I built, I loaded the test data which has 10000 image and calculated the rate of success by finding the errors between the digits I got and the labels provided. Finally, I built SVM (support vector machines) and decision tree classifiers on the digits and compared their performances with the linear classifier.

2 Theoretical Background

2.1 The Singular Value Decomposition

While multiplying a vector by a matrix, we have $\mathbf{Av} = \sigma\mathbf{u}$, in which σ represents a stretching vector and $\mathbf{u}$ represents a rotation vector. In other words, when we trying to change a vector, we change its direction by applying a rotation vector to it and change its length by applying a stretching vector to it. In matrix form, we have $\mathbf{AV} = \mathbf{U\Sigma}$, where $\mathbf{V}$ is a

unitary matrix, $\mathbf{U}$ represents a rotation matrix, and $\mathbf{\Sigma}$ represents a stretching matrix with σ on its diagonals representing singular values. For any unitary matrices, we have $\mathbf{V^{-1}} = \mathbf{V^*}$, where $\mathbf{V^*}$ means the transpose of V. Therefore, we can isolate A to get:

$$\mathbf{A} = \mathbf{U\Sigma V^*},$$

thus, giving us a decomposition of the matrix A through unitary matrices $\mathbf{U}$ and $\mathbf{V}$ and a diagonal matrix $\mathbf{\Sigma}$. [1] The principal components are the squares of the singular values.

In MATLAB, there is a built-in svd(X, 'econ') function that we can use to apply the reduced singular value decomposition, which is a more efficient way that has a lower order.

2.2 The Linear Discriminant Analysis

The goal of the linear discriminant analysis is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. While implementing the Linear Discriminant Analysis for 2 datasets, we try to find the correct subspace to project our data on so that we can better separate our groups. We define the between-class scatter matrix [2]

$$\mathrm{S_B} = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

where μ_1 and μ_2 are the means of each group. While the within-class scatter matrix is defined as

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

After finding the variance, we try to find the vector w which is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem that maximizes the quotient below:

$$\mathrm{w} = \mathrm{argmax} \frac{w^T S_B w}{w^T S_w w}$$

$$\mathrm{S_B} w = \lambda S_w w$$

While implementing the Linear Discriminant Analysis for 3 or more datasets, we have the variance between classes defined as below

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

We have the within-class variance defined as

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

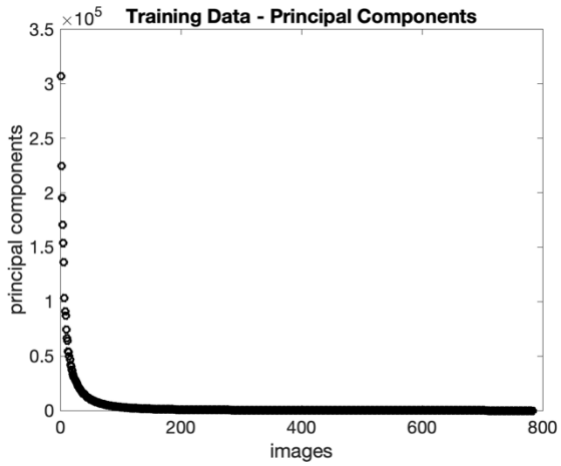
3 Algorithm Implementation and Development

For the first part, I used the singular value decomposition to apply the principal component analysis. After loading the data of images and labels, I first reshaped the images data, which was initially a 3D matrix $28 * 28 * 60000$, to be a 2D matrix $784 * 60000$, having each image in one column. Then before doing the singular value decomposition, I also have to demean my data. I first changed the data type from unit8 to double using im2double, then demean the data by finding the mean of each row and making copies using the repmat function, and finally I had my data with values around zero in hand. Then I find the singular values by applying the economic svd function. Since the principal components are the squares of the singular values, I squared the values and got my components, which made my important components more identifiable. Then I built the 3D singular value spectrum of the original data and the data that had been projected to the important component basis using the surf function. Finally, I got my data projected to the first three components and displayed the projections of each digit of pixel in a 3D plot.

For the second part, I applied the linear discriminant analysis. I first picked two digits from 0 to 9 and find the indices of the two digits from the data of labels using the find function. Then I found the vectors of the two digits and calculated the S_B and S_w, which are the between class variance and the within class variance respectively. By finding the eigenvalues and eigenvectors of S_B and S_w using the eig function, I was able to find the subspace w I was supposed to project on. After that, I tried to differentiate their means, sort the two data in an ascending order, set the value of the smaller digit to be 0 and the larger one to be 1, and find the threshold of the separation by going through each digit that was not in the range it was supposed to be. Similarly, I created a nested for loop for the process above to extract the model from each pair of digits in the training data. To test the accuracy of separation with LDA on the test data, I loaded the data of test images and labels, projected the images data onto the w subspace found from the training process of the corresponding pair of digits, and then calculate the rate of success by comparing to the labels provided. For the LDA of three digits, I followed a similar process and changed the values of separation according to the theorem stated in the theoretical background.

4 Computational Results

4.1 Singular Value Decomposition



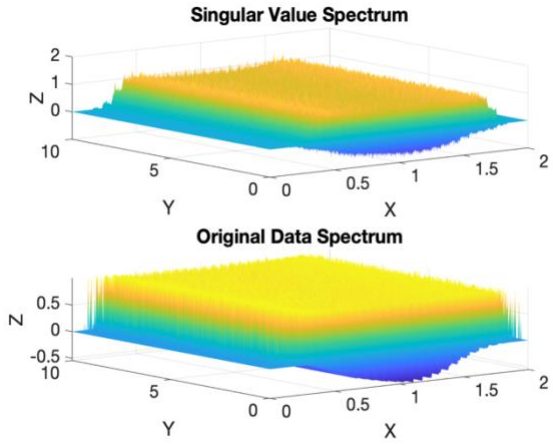


Figure 1 (left): plot of the distribution of the principal components of the training data
Figure 2 (right): (top) plot of the singular value spectrum based on the important components (the first 87 principal components); (bottom) the spectrum of the original training data

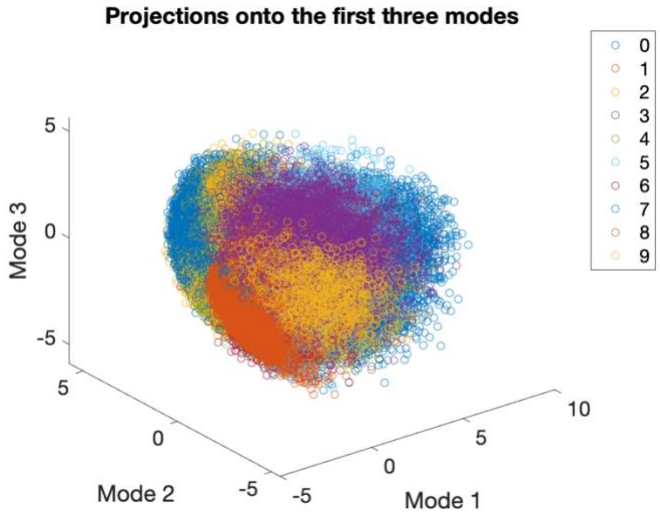


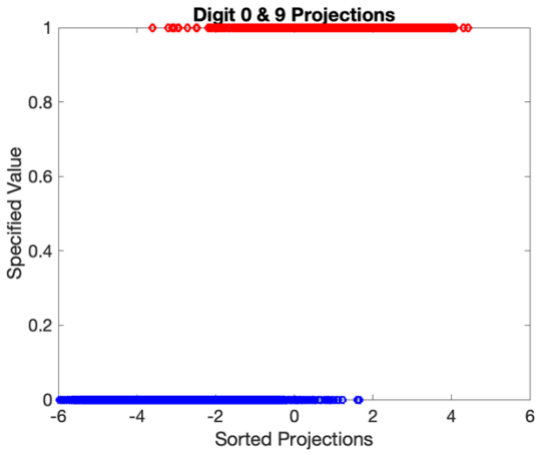
Figure 3: a plot of the projections of the training data onto the first three principal components (modes)

As seen in the figure 1, we know that the first a few principal components are important. By setting up a for loop to find the components that captured 90% of the energy, I found that the first 87 components are important. Therefore, the rank r of our training data is 87, and 87 modes are necessary for good image reconstruction.

Interpretations of $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$:

- $\mathbf{U}$ is a rotation matrix $784 * 784$ that captures the features of each digital image.
- $\mathbf{\Sigma}$ is a diagonal matrix $784 * 784$ that is a measure of analyzing how important every column of $\mathbf{U}$ is.
- $\mathbf{V}$ is a rotation matrix $784 * 60000$ that are the linear coefficients of the original sample that could help to reconstruct the data.

4.2 Linear Discriminant Analysis



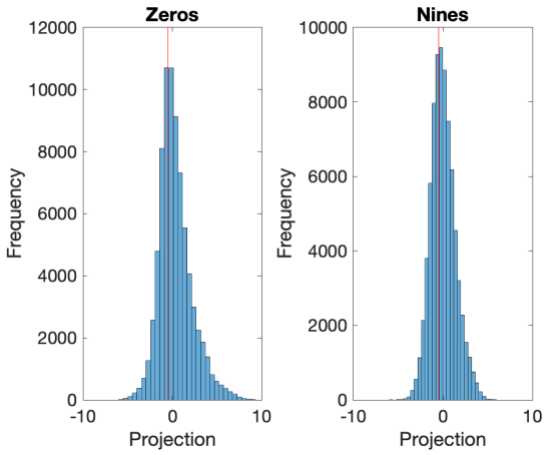


Figure 4 (left): A plot of the sorted projections of the digit 0 and digit 9 with digit 0 specified to be 0 and digit 9 specified to be 1
Figure 5 (right): A plot of the histogram of the projections of digit 0 and digit 9. The red line drawn shows the threshold of separation.

As seen in the above two plots, I looped each pair of digits following the same process. I got the pair that was the most difficult to separate to be digit 4 and digit 9 with a success rate of 56.91%, and the pair that was the least difficult to separate to be the digit 1 and digit 4 with a success rate of 99.76%. The pair of digit 0 and digit 9, which I used as an example, had a success rate of 98.49%.

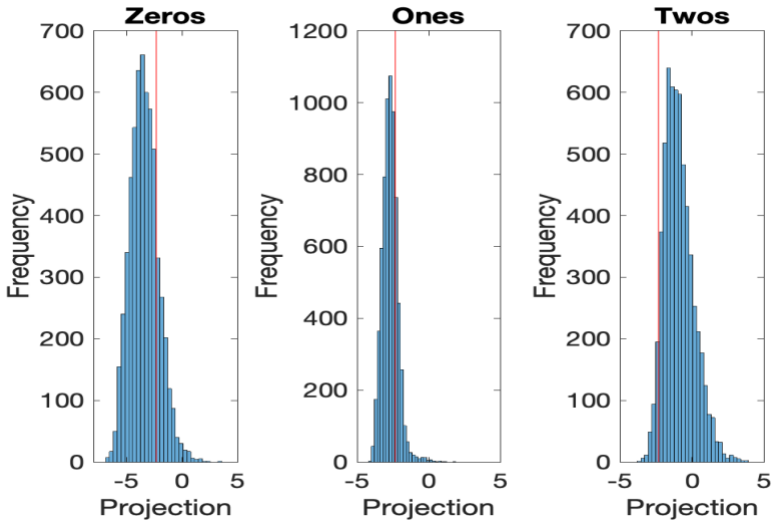


Figure 6: A plot of the histogram of the distribution of the projections after the LDA was applied to the first three digits 0, 1, and 2.

4.3 SVM, Decision Tree, & LDA

The speed of SVM extracting a model from the training data was much lower than the others, about 15 minutes. Regarding the success rates of the prediction models, SVM had a 94.46% success rate, which was much higher than decision tree, and we can't tell the difference between the rate given by LDA and SVM. Therefore, after comparison, LDA would be the best method doing the machine learning for this data because it was fast and relatively accurate.

5 Summary and Conclusions

By applying the singular value decomposition (the principal component analysis) and the linear discriminant analysis, I was able to analyze the training data provided, got a subspace that I was supposed to project my data onto, and used it as a model to predict the performance the future data. Comparing the LDA results with the test data, I got the basic idea of how well the model performed in the context of supervised machine learning.

References

[1] University of Washington. AMATH482 Lecture 11 Notes. URL: https://canvas.uw.edu/courses/1433289/files/71735394?wrap=1
[2] University of Washington. AMATH482 Lecture 19 Notes. URL: https://canvas.uw.edu/courses/1433289/files/72661443?wrap=1
[3] Jose Nathan Kutz. Data-driven modeling & scientific computation: methods for complex systems & bigdata. Oxford University Press, 2013.

Appendix A. MATLAB Functions

mean(A, dim)	computes the mean of elements along the dimension dim
reshape(A, sz)	reshape A to have a size of sz
find()	returns the indices of the elements satisfying the condition specified in find
[U, S, V] = svd(A, 'econ')	performs the singular value decomposition and produces an economy-size decomposition of matrix A
repmat(A, n)	repeat n copies of the array
surf(Z)	creates a surface plot
[V, D] = eig(A)	returns diagonal matrix D of eigenvalues and matrix V of eigenvectors
histogram(X, nbins)	plots a histogram with a number of bins specified in nbins

Appendix B. Extra Graph

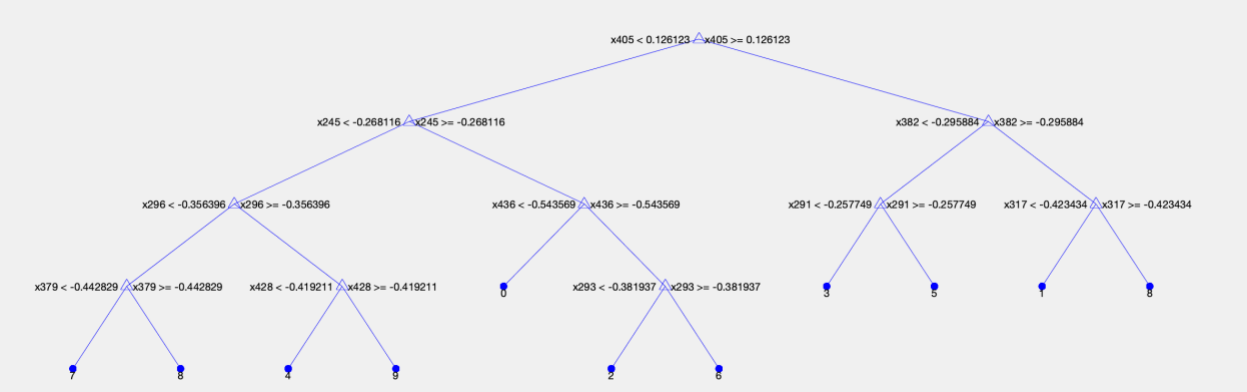


Figure 7: The plot produced by the decision tree method. As seen in the figure, digit 0 and digit 2 were the least difficult to separate with a success rate of 54.36%. Digit 1 and digit 7 were the most difficult to separate with a success rate of 12.61%.

Appendix C. MATLAB Code

```
%% Set up data
close all; clc; clear;

[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-
ubyte');
[imagesT, labelsT] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-
ubyte');

[n, ~, length] = size(images);
for k = 1:length
    digimages(:,k) = reshape(images(:,:,k), n*n, 1);
end

digimages = im2double(digimages);
digimages = digimages - repmat(mean(digimages, 2), 1, length);
[U, S, V] = svd(digimages, 'econ');
lambda = diag(S).^2;
k = 1;
while sum(lambda(1:k))/sum(lambda) < 0.9
    k = k + 1;
end
figure(1)
plot(lambda, 'ko', 'Linewidth', 2)
xlabel('images');ylabel('principal components');
title('Training Data - Principal Components');
set(gca,'Fontsize',16)
print('HW4PCA.png', '-dpng');


%% Building singular value spectrum
ff = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
x = linspace(0,2,784);
```

```
t = linspace(0,10,60000);
[T, X] = meshgrid(t,x);

figure(2)
subplot(2, 1, 1)
surf(X,T,ff)
shading interp
title('Singular Value Spectrum');
xlabel('X');ylabel('Y');zlabel('Z');
set(gca,'Fontsize',16);
subplot(2, 1, 2)
surf(X, T, digimages)
shading interp
title('Original Data Spectrum');
xlabel('X');ylabel('Y');zlabel('Z');
set(gca,'Fontsize',16);
print('HW4Spectrum.png', '-dpng');

% Make projections onto three selected V-modes (columns)
projection = U(:, [1, 2, 3])' * digimages;
projection_zero = projection(:, find(labels == 0));
projection_one = projection(:, find(labels == 1));
projection_two = projection(:, find(labels == 2));
projection_three = projection(:, find(labels == 3));
projection_four = projection(:, find(labels == 4));
projection_five = projection(:, find(labels == 5));
projection_six = projection(:, find(labels == 6));
projection_seven = projection(:, find(labels == 7));
projection_eight = projection(:, find(labels == 8));
projection_nine = projection(:, find(labels == 9));
figure(3)
plot3(projection_zero(1, :), projection_zero(2, :), projection_zero(3, :),
'o'); hold on
plot3(projection_one(1, :), projection_one(2, :), projection_one(3, :), 'o');
hold on
plot3(projection_two(1, :), projection_two(2, :), projection_two(3, :), 'o');
hold on
plot3(projection_three(1, :), projection_three(2, :), projection_three(3, :),
'o'); hold on
plot3(projection_four(1, :), projection_four(2, :), projection_four(3, :),
'o'); hold on
plot3(projection_five(1, :), projection_five(2, :), projection_five(3, :),
'o'); hold on
plot3(projection_six(1, :), projection_six(2, :), projection_six(3, :), 'o');
hold on
plot3(projection_seven(1, :), projection_seven(2, :), projection_seven(3, :),
'o'); hold on
plot3(projection_eight(1, :), projection_eight(2, :), projection_eight(3, :),
'o'); hold on
plot3(projection_nine(1, :), projection_nine(2, :), projection_nine(3, :),
'o'); hold on
legend('0','1','2','3','4','5','6','7','8','9')
xlabel('Mode 1'); ylabel('Mode 2'); zlabel('Mode 3');
title('Projections onto the first three modes');
set(gca,'Fontsize',16);
print('HW4Projections.png', '-dpng');
```

```
%% LDA for two digits
feature = 13;
images = S * V';
zerosidc = find(labels == 0);
ninesidc = find(labels == 9);
zeros = images(1:feature, zerosidc);
nines = images(1:feature, ninesidc);

numzeros = size(zerosidc', 2);
numnines = size(ninesidc', 2);

mzeros = mean(zeros,2);
mnines = mean(nines,2);
Sw = 0; % within class variances
for k = 1:numzeros
    Sw = Sw + (zeros(:,k) - mzeros)*(zeros(:,k) - mzeros)';
end
for k = 1:numnines
    Sw =  Sw + (nines(:,k) - mnines)*(nines(:,k) - mnines)';
end
Sb = (mzeros-mnines)*(mzeros-mnines)'; % between class

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

vzeros = w' * zeros;
vnines = w' * nines;

if mean(vzeros)>mean(vnines)
    w = -w;
    vzeros = -vzeros;
    vnines = -vnines;
end

figure(4)
plot(vzeros,0,'ob','Linewidth',2)
hold on
plot(vnines,ones(numnines),'dr','Linewidth',2)
xlabel('Sorted Projections'); ylabel('Specified Value');
title('Digit 0 & 9 Projections');
set(gca,'Fontsize',16);
print('HW4TwoDigitProjections.png', '-dpng');

sortzeros = sort(zeros);
sortnines = sort(nines);

t1 = numzeros;
t2 = 1;
while sortzeros(t1) > sortnines(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
```

```
threshold = (sortzeros(t1) + sortnines(t2))/2;

figure(5)
subplot(1,2,1)
histogram(sortzeros,30); hold on
plot([threshold threshold], [0 12000], 'r')
xlabel('Projection'); ylabel('Frequency');
title('Zeros')
set(gca,'Xlim',[-10 10],'Ylim',[0 12000],'Fontsize',16)
subplot(1,2,2)
histogram(sortnines,30); hold on
plot([threshold threshold], [0 10000],'r')
xlabel('Projection'); ylabel('Frequency');
title('Nines')
set(gca,'Xlim',[-10 10],'Ylim',[0 10000],'Fontsize',16)
print('HW4TwoDigitHistograms.png', '-dpng');

[nT, ~, lengthT] = size(imagesT);
for k = 1:lengthT
    digimagesT(:,k) = reshape(imagesT(:,:,k), nT*nT, 1);
end

TestNum = size(digimagesT,2);
digimagesT = im2double(digimagesT);
TestMat = U(:,1:feature)'*digimagesT; % PCA projection
pval = w'*TestMat;

zerosidcT = find(labelsT == 0);
ninesidcT = find(labelsT == 9);

ResVec = (pval(ninesidcT) > threshold);
errNines = abs(sum(ResVec) - size(ninesidcT',2));

ResVec = (pval(zerosidcT) < threshold);
errZeros = abs(sum(ResVec) - size(zerosidcT',2));

sucRate = 1 - (errNines + errZeros)/(size(ninesidcT',2) +
size(zerosidcT',2));

% Pairs that are the most & least difficult to separate
min = 1;
max = 0;
index = 0;
for k = 0:8
    for j = (k+1):9
        index = index + 1;
        labelidc1 = find(labels == k);
        labelidc2 = find(labels == j);
        num1 = images(1:feature, labelidc1);
        num2 = images(1:feature, labelidc2);
        [threshold, w, sortNum1, sortNum2] = train(labelidc1, labelidc2,
num1, num2);
        TestNum = size(digimagesT,2);
        digimagesT = im2double(digimagesT);
        TestMat = U(:,1:feature)'*digimagesT; % PCA projection
```

```
        pval = w'*TestMat;
        labelsidcT1 = find(labelsT == k);
        labelsidcT2 = find(labelsT == j);
        sucrate = test(labelsidcT1, labelsidcT2, threshold, pval);
        rate(1,index) = sucrate;
        if sucrate < min
            min = sucrate;
            minNum1 = k;
            minNum2 = j;
        end
        if sucrate > max
            max = sucrate;
            maxNum1 = k;
            maxNum2 = j;
        end
    end
end

%% LDA for three digits
feature = 13;
images = S * V';
zerosidc = find(labels == 0);
onesidc = find(labels == 1);
twosidc = find(labels == 2);

zeros = images(1:feature, zerosidc);
ones = images(1:feature, onesidc);
twos = images(1:feature, twosidc);

numzeros = size(zerosidc', 2);
numones = size(onesidc', 2);
numtwos = size(twosidc', 2);

mzeros = mean(zeros,2);
mones = mean(ones,2);
mtwos = mean(twos,2);
mn = (mzeros + mones + mtwos)/3;

Sw = 0; % within class variances
for k = 1:numzeros
    Sw = Sw + (zeros(:,k) - mzeros)*(zeros(:,k) - mzeros)';
end
for k = 1:numones
    Sw =  Sw + (ones(:,k) - mones)*(ones(:,k) - mones)';
end
for k = 1:numtwos
    Sw =  Sw + (twos(:,k) - mtwos)*(twos(:,k) - mtwos)';
end
Sb = (mzeros-mn)*(mzeros-mn)' + (mones-mn)*(mones-mn)' + (mtwos-mn)*(mtwos-
mn)'; % between class

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

```
vzeros = w' * zeros;
vones = w' * ones;
vtwos = w' * twos;

% setup threshold
if mean(vzeros) > mean(vones)
    w = -w;
    vzeros = -vzeros;
    vones = -vones;
end
if mean(vones) > mean(vtwos)
    w = -w;
    vones = -vones;
    vtwos = -vtwos;
end

szeros = sort(vzeros);
sones = sort(vones);
stwos = sort(vtwos);
t1 = numzeros;
t2 = 1;
while szeros(t1) > sones(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold1 = (szeros(t1) + sones(t2))/2;

t1 = numzeros;
t3 = 1;
while szeros(t1) > stwos(t3)
    t1 = t1 - 1;
    t3 = t3 + 1;
end
threshold2 = (szeros(t1) + stwos(t3))/2;

t2 = numones;
t3 = 1;
while sones(t2) > stwos(t3)
    t2 = t2 - 1;
    t3 = t3 + 1;
end
threshold3 = (sones(t2) + stwos(t3))/2;

threshold = (threshold1 + threshold2 + threshold3)/3;

figure(6)
subplot(1,3,1)
histogram(szeros,30); hold on
plot([threshold threshold], [0 700], 'r')
title('Zeros')
xlabel('Projection'); ylabel('Frequency');
set(gca,'Xlim',[-8 5],'Ylim',[0 700],'Fontsize',16)
subplot(1,3,2)
histogram(sones,30); hold on
```

```
plot([threshold threshold], [0 1200],'r')
title('Ones')
xlabel('Projection'); ylabel('Frequency');
set(gca,'Xlim',[-5 5],'Ylim',[0 1200],'Fontsize',16)
subplot(1,3,3)
histogram(stwos,30); hold on
plot([threshold threshold], [0 700],'r')
title('Twos')
xlabel('Projection'); ylabel('Frequency');
set(gca,'Xlim',[-5 5],'Ylim',[0 700],'Fontsize',16)
print('HW4ThreeDigitHistograms.png', '-dpng');

%% classification tree on fisheriris data

tree=fitctree(digimages',labels,'MaxNumSplits',10,'CrossVal','on');
view(tree.Trained{1},'Mode','graph');
classError = kfoldLoss(tree, 'Mode', 'individual');

%% SVM classifier with training data, labels and test set

success = 0;
Mdl = fitcecoc(digimages',labels);
test_labels = predict(Mdl,digimagesT');
for j = 1:length(test_labels)
    if (test_labels(j) - labelsT(j)) == 0
        success = success + 1;
    end
end

function sucrate = test(labelsidcT1, labelsidcT2, threshold, pval)
    ResVec = (pval(labelsidcT2) > threshold);
    errNum2 = abs(sum(ResVec) - size(labelsidcT2',2));

    ResVec = (pval(labelsidcT1) < threshold);
    errNum1 = abs(sum(ResVec) - size(labelsidcT1',2));

    sucrate = 1 - (errNum2 + errNum1)/(size(labelsidcT2',2) +
size(labelsidcT1',2));
end

function [threshold, w, sortNum1, sortNum2] = train(labelidc1, labelidc2,
num1, num2)
    nNum1 = size(labelidc1', 2);
    nNum2 = size(labelidc2', 2);

    mNum1 = mean(num1,2);
    mNum2 = mean(num2,2);
    Sw = 0; % within class variances
    for k = 1:nNum1
        Sw = Sw + (num1(:,k) - mNum1)*(num1(:,k) - mNum1)';
    end
    for k = 1:nNum2
        Sw =  Sw + (num2(:,k) - mNum2)*(num2(:,k) - mNum2)';
    end
```

```
    Sb = (mNum1-mNum2)*(mNum1-mNum2)'; % between class

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

    vNum1 = w' * num1;
    vNum2 = w' * num2;

    if mean(vNum1)>mean(vNum2)
        w = -w;
        vNum1 = -vNum1;
        vNum2 = -vNum2;
    end

    sortNum1 = sort(num1);
    sortNum2 = sort(num2);

    t1 = nNum1;
    t2 = 1;
    while sortNum1(t1) > sortNum2(t2)
        t1 = t1 - 1;
        t2 = t2 + 1;
    end
    threshold = (sortNum1(t1) + sortNum2(t2))/2;

end
```