# AMATH 582 Homework 4

Joseph David

March 10, 2021

**Abstract**

In this project we work with the MNIST dataset of hand-drawn digits. First, we perform SVD analysis to determine how many modes are necessary for good image reconstruction. Then, we train linear classifiers (LDAs), support vector machines (SVMs), and decision tree classifiers on the training images and compare their performances in classifying digits from the test images.

## 1 Introduction and Overview

We begin by downloading the MNIST dataset of hand-drawn digits as well as their labels, and creating matrices storing this data. We then perform a Singular Value Decompostion (SVD) in order to analyze the number of modes necessary for good image reconstruction, and plot a few modes in order to present a visual on the clustering that occurs among images of the same digit.

Next, we build linear classifiers (LDAs) using the training images and test it on the test images and compute how well the LDAs do at classifying all pairs of digits. We also compute how well LDAs when it has to classify three digits as opposed to just pairs of digits.

Finally, we use MATLAB functions to create support vector machines (SVMs) and decision tree classifiers and use them to classify pairs of digits. We then compare these results to those from LDA.

## 2 Theoretical Background

The first analytical technique we employ is SVD analysis. Given a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$, there exists orthonormal bases for $\mathbb{R}^m$ and $\mathbb{R}^n$ with respect to which $T$ is defined by a diagonal matrix. In terms of matrices, suppose that $A$ is the $m \times n$ matrix such that, for all $x \in \mathbb{R}^m$,

$$T(x) = Ax. \tag{1}$$

Then, the SVD theorem states that there exist unitary matrices $U$, $V$ and diagonal matrix $\Sigma$ such that

$$A = U\Sigma V^*, \tag{2}$$

where $V^*$ denotes the complex conjugate of $V$. The *singular value spectrum* of $A$ is defined as the diagonal of $\Sigma$, and these values tell us which components contain the most information in the system, and hence this type of analysis is called Principal Component Analysis (PCA). The usefulness of PCA is that it reveals which directions (components) are most important and hence reveals patterns. In Figure 1, we extracted three rows from the matrix $V$ and only took rows that corresponded to digits 1,2, or 3. We then plotted each row as a point in 3D and color coded it based on which digit. We immediately see that the same digits tend to be have similar values in these three columns and hence cluster.

## 3 Algorithm Implementation and Development

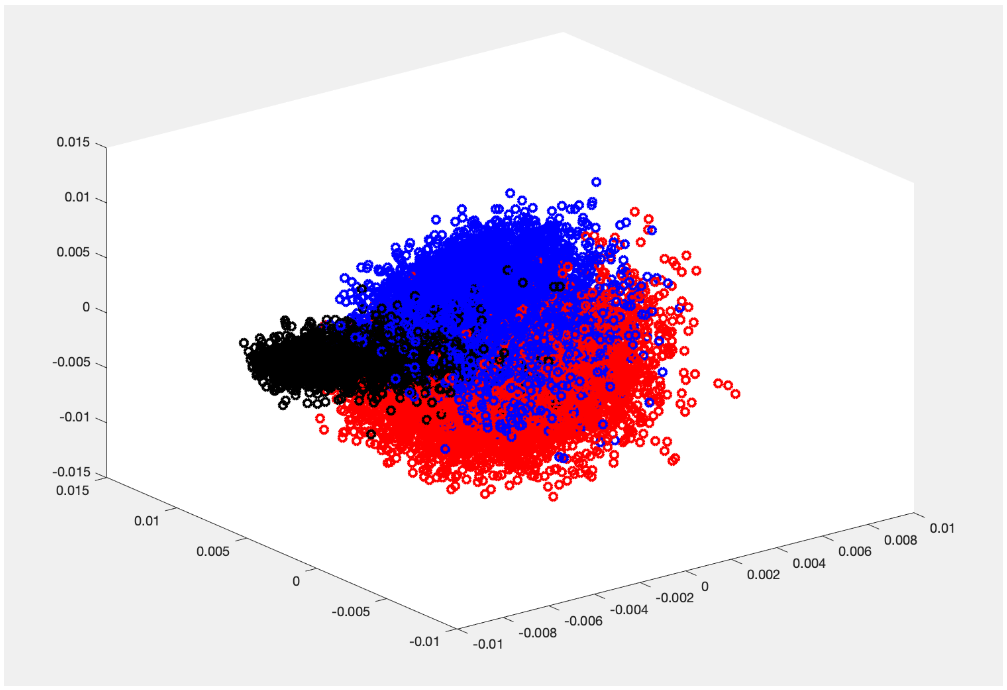

Figure 1: Here we plotted three columns from rows of $V$ that corresponded to digits of 1,2, and 3. We see significant clustering, and this will allow us to classify new images of digits.

**Algorithm 1:** Linear Classifier (LDA)

```
Import the 3D points of two sets of training digits
Take the average over each set to obtain two points avgdigit1 and avgdigit2
Given a point, compute which the closest average point is and return that digit.
```

## 4 Computational Results

After computing the singular value spectrum on the matrix of training images, we found the singular value spectrum (see Figure 3) and in addition plotted the image reconstruction of two images using various number of modes. We see that most of the energy is contained in the first 100 modes, and from the image reconstruction, it does not take many modes for the human eye to determine what the digits are.

Next, we built linear classifiers for all digits and used that to test how well they could distinguish between pairs of digits from the test digits. In Figure 4 we plotted where the test digits for digits 1 and 2 landed, and whether they were correctly labeled by the LDA. From the figure, it appears that 1 and 2 were well clustered, and so the LDA performed well classifying them. In Figure 5, it appears that the digit 0 overall was the easiest to classify, especially when distinguishing it from 1 and 9. On the other hand pairs such as (3,7), (3,4), and (1,9) had very poor performance.

Then we used MATLAB SVM functions and tested its performance in classifying digits, see Figure 6. We see that this performed much worse than LDA, but interestingly the same pairs of digits that were did well on LDA also did well in SVM, and similarly for the bad pairs. For instance, SVM did well classifying against 0 but also did poorly with (3,7) and (3,4).

## 5 Summary and Conclusions

In conclusion, we saw that PCA provides a very effective platform from which to classify clustered data. Surprisingly, LDA performed best. However, what was most interesting to me was that both LDA and SVM

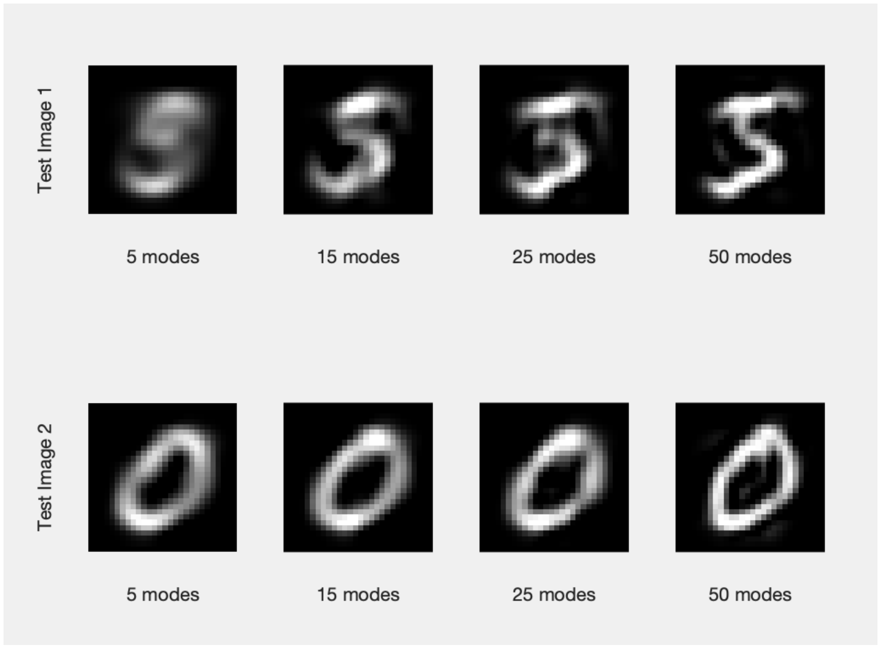

Figure 2: It does not take many modes in these two cases to tell which digit it is.

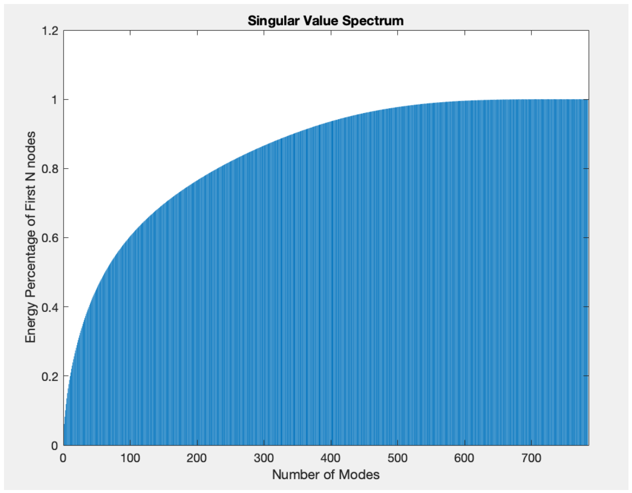

Figure 3: Singular value spectrum. We see, for instance, that 60 percent of the energy is contained in the first 100 modes.

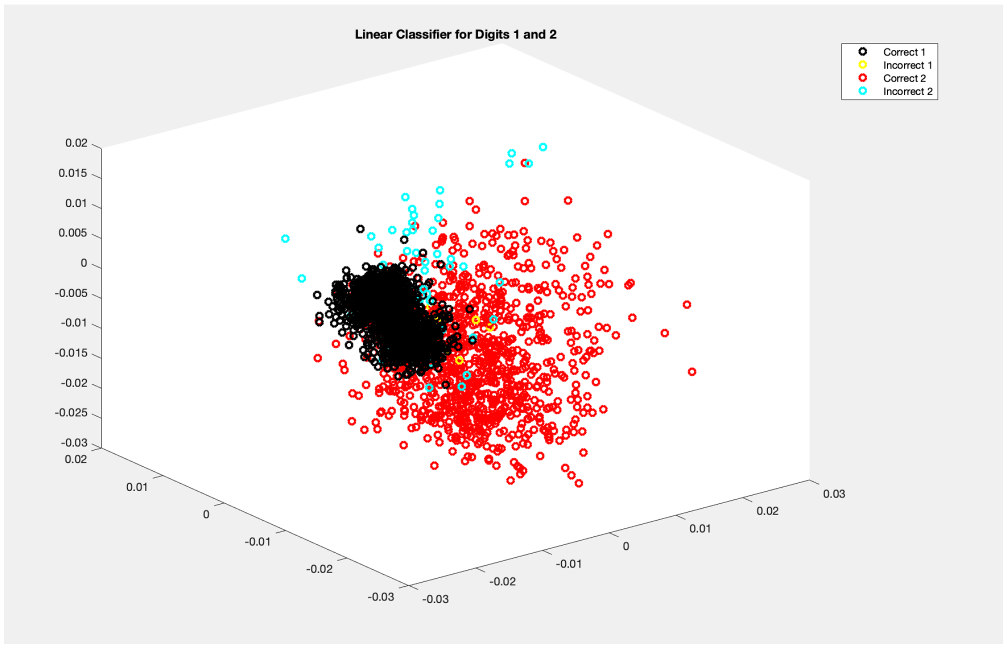

Figure 4: This shows how well the linear classifier performed on the test digits using digits 1 and 2. Black and red indicate the the classifier was correct, whereas yellow and cyan are false 1's or 2's, respectively.

had similar best and worst pairs of digits. This implies that in some cases, the clustering may be too little so that no algorithm can do a good job identifying it.

## Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.

## Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

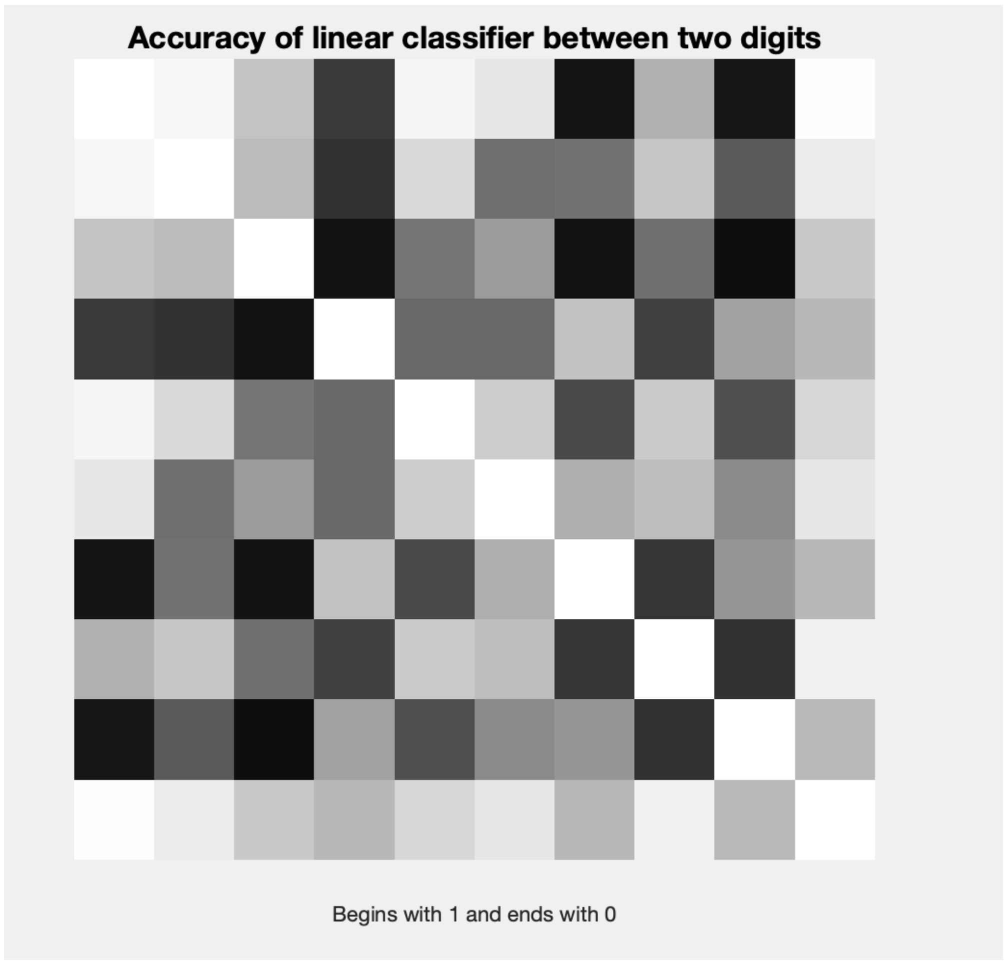

Figure 5: This table indicates how well the LDA performed on pairs of digits. So, the plot is symmetric, with white indicating 100 percent accuracy and black indicating 0 percent accuracy.

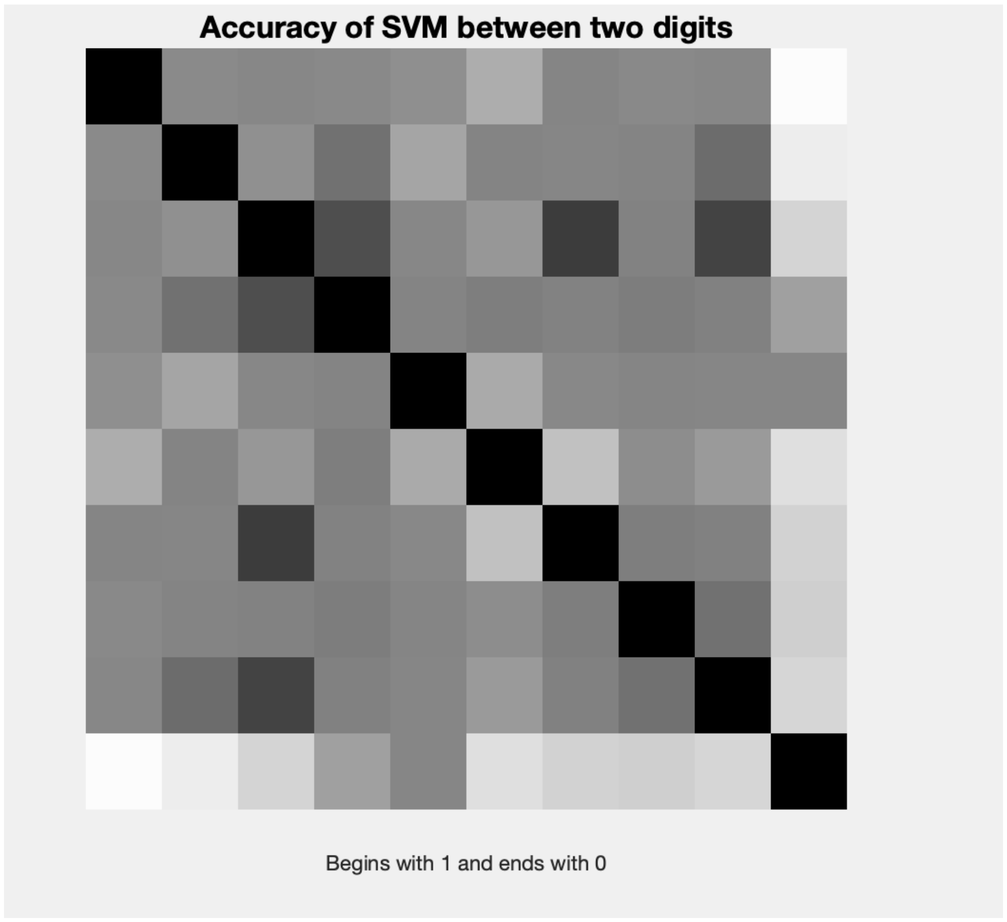

Figure 6: This table indicates how well SVM performed on pairs of digits. So, the plot is symmetric, with white indicating 100 percent accuracy and black indicating 0 percent accuracy.

```
clear all; close all; clc

myFolder1 = '/Users/joseph/Documents/MATLAB/t10k-images.idx3-ubyte';
myFolder2 = '/Users/joseph/Documents/MATLAB/t10k-labels.idx1-ubyte';
myFolder3 = '/Users/joseph/Documents/MATLAB/train-images.idx3-ubyte';
myFolder4 = '/Users/joseph/Documents/MATLAB/train-labels.idx1-ubyte';

TestImages = loadMNISTImages(myFolder1);
TrainImages = loadMNISTImages(myFolder3);
TestLabels = loadMNISTLabels(myFolder2);
TrainLabels = loadMNISTLabels(myFolder4);

[U,S,V] = svd(TrainImages,'econ');
[U2,S2,V2] = svd(TestImages,'econ');

S_sum = zeros(1,784); %stores cumulative energy of first j singular values
sum = sum(S,'all');
for j = 1 : 784
    if j == 1
        S_sum(j) = S(j);
    else S_sum(j) = S_sum(j-1) + S(j,j);
    end
end

figure(1)
bar(S_sum/sum)
title('Singular Value Spectrum')
xlabel('Number of Modes'); ylabel('Energy Percentage of First N nodes')

TI_col1 = zeros(28,28);
TI_col2 = zeros(28,28);
N=50;
figure(2)
for j = 1 : N
    TI_col1 = TI_col1 + dot(TrainImages(:,1),U(:,j)) * reshape(U(:,j),28,28);
    TI_col2 = TI_col2 + dot(TrainImages(:,2),U(:,j)) * reshape(U(:,j),28,28);
    if j == 5
        subplot(2,4,1), imshow(TI_col1)
        xlabel('5 modes'); ylabel('Test Image 1')
        subplot(2,4,5), imshow(TI_col2)
        xlabel('5 modes'); ylabel('Test Image 2')
    end
    if j == 15
        subplot(2,4,2), imshow(TI_col1)
        xlabel('15 modes')
        subplot(2,4,6), imshow(TI_col2)
        xlabel('15 modes')
    end
    if j == 25
        subplot(2,4,3), imshow(TI_col1)
        xlabel('25 modes')
        subplot(2,4,7), imshow(TI_col2)
        xlabel('25 modes')
    end
    if j == 50
        subplot(2,4,4), imshow(TI_col1)
        xlabel('50 modes')
        subplot(2,4,8), imshow(TI_col2)
        xlabel('50 modes')
    end
end
```

```
ones=[]; twos=[]; threes=[]; fours=[]; fives=[]; sixes=[]; sevens=[];
eights=[]; nines=[]; zers=[];

for j = 1 : length(TrainLabels(1,:))
   if TrainLabels(1,j) == 1
       ones(end+1) = j;
   elseif TrainLabels(2,j) == 1
       twos(end+1) = j;
   elseif TrainLabels(3,j) == 1
       threes(end+1) = j;
   elseif TrainLabels(4,j) == 1
       fours(end+1) = j;
   elseif TrainLabels(5,j) == 1
       fives(end+1) = j;
   elseif TrainLabels(6,j) == 1
       sixes(end+1) = j;
   elseif TrainLabels(7,j) == 1
       sevens(end+1) = j;
   elseif TrainLabels(8,j) == 1
       eights(end+1) = j;
   elseif TrainLabels(9,j) == 1
       nines(end+1) = j;
   else
       zers(end+1) = j;
   end
end

figure(3)
plot3(V(ones,2),V(ones,3),V(ones,5),'ko','Linewidth',[2])
hold on
plot3(V(twos,2),V(twos,3),V(twos,5),'ro','Linewidth',[2])
hold on
plot3(V(threes,2),V(threes,3),V(threes,5),'bo','Linewidth',[2])

ones2=[]; twos2=[]; threes2=[]; fours2=[]; fives2=[]; sixes2=[]; sevens2=[];
eights2=[]; nines2=[]; zers2=[];

for j = 1 : length(TestLabels(1,:))
   if TestLabels(1,j) == 1
       ones2(end+1) = j;
   elseif TestLabels(2,j) == 1
       twos2(end+1) = j;
   elseif TestLabels(3,j) == 1
       threes2(end+1) = j;
   elseif TestLabels(4,j) == 1
       fours2(end+1) = j;
   elseif TestLabels(5,j) == 1
       fives2(end+1) = j;
   elseif TestLabels(6,j) == 1
       sixes2(end+1) = j;
   elseif TestLabels(7,j) == 1
       sevens2(end+1) = j;
   elseif TestLabels(8,j) == 1
       eights2(end+1) = j;
   elseif TestLabels(9,j) == 1
       nines2(end+1) = j;
   else
       zers2(end+1) = j;
   end
end

one_3d = [V(ones,2)'; V(ones,3)'; V(ones,5)'];
two_3d = [V(twos,2)'; V(twos,3)'; V(twos,5)'];
three_3d = [V(threes,2)'; V(threes,3)'; V(threes,5)'];
avg1 = linear_classifier(one_3d);
avg2 = linear_classifier(two_3d);
avg3 = linear_classifier(three_3d);

correct_ones=zeros(3,1);
incorrect_ones=zeros(3,1);
correct_twos=zeros(3,1);
incorrect_twos=zeros(3,1);

for j = 1 : length(ones2)
    x = [V2(ones2(j),2);V2(ones2(j),3);V2(ones2(j),5)];
    if norm(x-avg1)<=norm(x-avg2)
        correct_ones(:,end+1) = x;
    else
        incorrect_ones(:,end+1) = x;
    end
end

for j = 1 : length(twos2)
    x = [V2(twos2(j),2);V2(twos2(j),3);V2(twos2(j),5)];
    if norm(x-avg2)<=norm(x-avg1)
        correct_twos(:,end+1) = x;
    else
        incorrect_twos(:,end+1) = x;
    end
end

figure(4)
plot3(correct_ones(1,:),correct_ones(2,:),correct_ones(3,:),'ko','Linewidth',[2])
hold on
plot3(incorrect_ones(1,:),incorrect_ones(2,:),incorrect_ones(3,:),'yo','Linewidth',[2])
hold on
plot3(correct_twos(1,:),correct_twos(2,:),correct_twos(3,:),'ro','Linewidth',[2])
hold on
plot3(incorrect_twos(1,:),incorrect_twos(2,:),incorrect_twos(3,:),'co','Linewidth',[2])
title('Linear Classifier for Digits 1 and 2')
legend('Correct 1', 'Incorrect 1', 'Correct 2', 'Incorrect 2')
xlabel('')

correct_ones=zeros(3,1);
incorrect_ones_two=zeros(3,1);
incorrect_ones_three=zeros(3,1);
correct_twos=zeros(3,1);
incorrect_twos_one=zeros(3,1);
incorrect_twos_three=zeros(3,1);
correct_threes=zeros(3,1);
incorrect_threes_one=zeros(3,1);
incorrect_threes_two=zeros(3,1);
```

```
for j = 1 : length(ones2)
    x = [V2(ones2(j),2);V2(ones2(j),3);V2(ones2(j),5)];
    m = min( [norm(x-avg1) norm(x-avg2) norm(x-avg3)] );
    if norm(x-avg1) == m
        correct_ones(:,end+1) = x;
    elseif norm(x-avg2) == m
        incorrect_ones_two(:,end+1) = x;
    else
        incorrect_ones_three(:,end+1)=x;
    end
end
for j = 1 : length(twos2)
    x = [V2(twos2(j),2);V2(twos2(j),3);V2(twos2(j),5)];
    m = min( [norm(x-avg1) norm(x-avg2) norm(x-avg3)] );
    if norm(x-avg2) == m
        correct_twos(:,end+1) = x;
    elseif norm(x-avg1) == m
        incorrect_twos_one(:,end+1) = x;
    else
        incorrect_twos_three(:,end+1)=x;
    end
end
for j = 1 : length(threes2)
    x = [V2(ones2(j),2);V2(ones2(j),3);V2(ones2(j),5)];
    m = min( [norm(x-avg1) norm(x-avg2) norm(x-avg3)] );
    if norm(x-avg3) == m
        correct_threes(:,end+1) = x;
    elseif norm(x-avg1) == m
        incorrect_threes_one(:,end+1) = x;
    else
        incorrect_threes_two(:,end+1)=x;
    end
end

figure(5)
plot3(correct_ones(1,:),correct_ones(2,:),correct_ones(3,:),'ko','Linewidth',[2])
hold on
plot3(correct_twos(1,:),correct_twos(2,:),correct_twos(3,:),'ro','Linewidth',[2])
hold on
plot3(correct_threes(1,:),correct_threes(2,:),correct_threes(3,:),'bo','Linewidth',[2])
hold on
plot3(incorrect_ones_two(1,:),incorrect_ones_two(2,:),incorrect_ones_two(3,:),'k*','Linewidth',[2])
hold on
plot3(incorrect_ones_three(1,:),incorrect_ones_three(2,:),incorrect_ones_three(3,:),'k+','Linewidth',[2
hold on
plot3(incorrect_twos_one(1,:),incorrect_twos_one(2,:),incorrect_twos_one(3,:),'r^','Linewidth',[2])
hold on
plot3(incorrect_twos_three(1,:),incorrect_twos_three(2,:),incorrect_twos_three(3,:),'r+','Linewidth',[2
hold on
plot3(incorrect_threes_one(1,:),incorrect_threes_one(2,:),incorrect_threes_one(3,:),'b^','Linewidth',[2
hold on
plot3(incorrect_threes_two(1,:),incorrect_threes_two(2,:),incorrect_threes_two(3,:),'b*','Linewidth',[2

ones_twos_threes = [ones2 twos2 threes2];
correct_decision=0;
```

```
for j = 1 : length(ones_twos_threes)
    x = [V2(ones_twos_threes(j),2);V2(ones_twos_threes(j),3);V2(ones_twos_threes(j),5)];
    if min([norm(x-avg1) norm(x-avg2) norm(x-avg3)]) == norm(x-avg1)
        if j <= length(ones2)
            correct_decision = correct_decision+1;
        end
        figure(6)
        plot3(V2(ones_twos_threes(j),2),V2(ones_twos_threes(j),3),V2(ones_twos_threes(j),5),'ko','Linew
        hold on
    elseif min([norm(x-avg1) norm(x-avg2) norm(x-avg3)]) == norm(x-avg2)
        if length(ones2) < j <= length(twos2)
            correct_decision = correct_decision + 1;
        end
        figure(6)
        plot3(V2(ones_twos_threes(j),2),V2(ones_twos_threes(j),3),V2(ones_twos_threes(j),5),'ro','Linew
        hold on
    else
        if j > length(twos2)
            correct_decision = correct_decision + 1;
        end
        figure(6)
        plot3(V2(ones_twos_threes(j),2),V2(ones_twos_threes(j),3),V2(ones_twos_threes(j),5),'bo','Linew
        hold on
    end
end

string = ['The linear classifier got ',num2str(sum(correct_decision)/length(ones_twos_threes)),'% corre
disp(string)

four_3d = [V(fours,2)'; V(fours,3)'; V(fours,5)'];
five_3d = [V(fives,2)'; V(fives,3)'; V(fives,5)'];
six_3d = [V(sixes,2)'; V(sixes,3)'; V(sixes,5)'];
seven_3d = [V(sevens,2)'; V(sevens,3)'; V(sevens,5)'];
eight_3d = [V(eights,2)'; V(eights,3)'; V(eights,5)'];
nine_3d = [V(nines,2)'; V(nines,3)'; V(nines,5)'];
zero_3d = [V(zers,2)'; V(zers,3)'; V(zers,5)'];

%for linear classifier
avg4 = linear_classifier(four_3d);
avg5 = linear_classifier(five_3d);
avg6 = linear_classifier(six_3d);
avg7 = linear_classifier(seven_3d);
avg8 = linear_classifier(eight_3d);
avg9 = linear_classifier(nine_3d);
avg0 = linear_classifier(zero_3d);
```

```
lda_accuracy = zeros(10,10);
for i = 1 : 10
    for j = i : 10
        if i == 1
            dg = ones2; a1 = avg1;
        elseif i == 2
            dg = twos2; a1 = avg2;
        elseif i == 3
            dg = threes2; a1 = avg3;
        elseif i == 4
            dg = fours2; a1 = avg4;
        elseif i == 5
            dg = fives2; a1 = avg5;
        elseif i == 6
            dg = sixes2; a1 = avg6;
        elseif i == 7
            dg = sevens2; a1 = avg7;
        elseif i == 8
            dg = eights2; a1 = avg8;
        elseif i == 9
            dg = nines2; a1 = avg9;
        else
            dg = zers2; a1 = avg0;
        end

        if j == 1
            dg2 = ones2; a2 = avg1;
        elseif j == 2
            dg2 = twos2; a2 = avg2;
        elseif j == 3
            dg2 = threes2; a2 = avg3;
        elseif j == 4
            dg2 = fours2; a2 = avg4;
        elseif j == 5
            dg2 = fives2; a2 = avg5;
        elseif j == 6
            dg2 = sixes2; a2 = avg6;
        elseif j == 7
            dg2 = sevens2; a2 = avg7;
        elseif j == 8
            dg2 = eights2; a2 = avg8;
        elseif j == 9
            dg2 = nines2; a2 = avg9;
        else
            dg2 = zers2; a2 = avg0;
        end

        A = [V2(dg,2)';V2(dg,3)';V2(dg,5)'];
        B = [V2(dg2,2)';V2(dg2,3)';V2(dg2,5)'];

        lda_accuracy(i,j) = separate_two_digits(A,B,a1,a2);
        lda_accuracy(j,i) = lda_accuracy(i,j);
    end
end
```

```
figure(6)
imshow(lda_accuracy,[0 1], 'InitialMagnification',5000)
t = title('Accuracy of linear classifier between two digits')
xlabel('Begins with 1 and ends with 0')
t.FontSize = 16;

% % classification tree on fisheriris data
% load fisheriris;
% tree=fitctree(meas,species,'MaxNumSplits',3,'CrossVal','on');
% view(tree.Trained{1},'Mode','graph');
% classError = kfoldLoss(tree)

svm_accuracy = zeros(10,10);
for i = 1 : 10
    for j = i+1 : 10
        if i == 1
            train1 = one_3d; dg = ones2;
        elseif i == 2
            train1 = two_3d; dg = twos2;
        elseif i == 3
            train1 = three_3d; dg = threes2;
        elseif i == 4
            train1 = four_3d; dg = fours2;
        elseif i == 5
            train1 = five_3d; dg = fives2;
        elseif i == 6
            train1 = six_3d; dg = sixes2;
        elseif i == 7
            train1 = seven_3d; dg = sevens2;
        elseif i == 8
            train1 = eight_3d; dg = eights2;
        elseif i == 9
            train1 = nine_3d; dg = nines2;
        else
            train1 = zero_3d; dg = zers2;
        end

        if j == 1
            train2 = one_3d; dg2 = ones2;
        elseif j == 2
            train2 = two_3d; dg2 = twos2;
        elseif j == 3
            train2 = three_3d; dg2 = threes2;
        elseif j == 4
            train2 = four_3d; dg2 = fours2;
        elseif j == 5
            train2 = five_3d; dg2 = fives2;
        elseif j == 6
            train2 = six_3d; dg2 = sixes2;
        elseif j == 7
            train2 = seven_3d; dg2 = sevens2;
        elseif j == 8
            train2 = eight_3d; dg2 = eights2;
        elseif j == 9
            train2 = nine_3d; dg2 = nines2;
        else
            train2 = zero_3d; dg2 = zers2;
        end
```

```
A = [V2(dg,2)';V2(dg,3)';V2(dg,5)'];
        B = [V2(dg2,2)';V2(dg2,3)';V2(dg2,5)'];

        svm_accuracy(i,j) = svm_classify(train1,train2,A,B);
        svm_accuracy(j,i) = svm_accuracy(i,j);
    end
end

figure(7)
imshow(svm_accuracy,[0 1], 'InitialMagnification',5000)
t = title('Accuracy of SVM between two digits');
xlabel('Begins with 1 and ends with 0')
t.FontSize = 16;

function x1 = linear_classifier(train_digit)
    x1 = [0;0;0];
    for j = 1 : length(train_digit(1,:))
        x1 = x1 + train_digit(:,j);
    end
    x1 = x1 / length(train_digit(1,:));
end

function r = separate_two_digits(digits1, digits2, avg1, avg2)
    r = 0;
    for j = 1 : length(digits1(1,:))
        if norm(digits1(:,j) - avg1) <= norm(digits1(:,j)-avg2)
            r = r+1;
        end
    end
    for j = 1 : length(digits2(1,:))
        if norm(digits2(:,j) - avg2) <= norm(digits2(:,j)-avg1)
            r = r+1;
        end
    end
    r = r / (length(digits1) + length(digits2));
end

function r = svm_classify(train_digits1, train_digits2, test_digits1, test_digits2)
    r = 0;
    xtrain = [train_digits1'; train_digits2'];
    label = [ones(length(train_digits1(1,:)),1) ; 2*ones(length(train_digits2(1,:)),1)];
    test = [test_digits1 test_digits2]';
    Mdl = fitcsvm(xtrain,label);
    test_labels = predict(Mdl,test);
    for j = 1 : length(test_digits1(1,:))
        if test_labels(j) == 1
            r = r+1;
        end
    end
    for j = length(test_digits1(1,:))+1 : length(test_digits1(1,:))+length(test_digits2(1,:))
        if test_labels(j) == 2
            r = r+1;
        end
    end
    r = r / (length(test_digits1(1,:)) + length(test_digits2(1,:)));
end
```

```
function images = loadMNISTImages(filename)
%loadMNISTImages returns a (28^2)x[number of MNIST images] matrix containing
%the raw MNIST images

fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename, '']);

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2051, ['Bad magic number in ', filename, '']);

numImages = fread(fp, 1, 'int32', 0, 'ieee-be');
numRows = fread(fp, 1, 'int32', 0, 'ieee-be');
numCols = fread(fp, 1, 'int32', 0, 'ieee-be');

images = fread(fp, inf, 'unsigned char');
images = reshape(images, numCols, numRows, numImages);
images = permute(images,[2 1 3]);

fclose(fp);

% Reshape to #pixels x #examples
images = reshape(images, size(images, 1) * size(images, 2), size(images, 3));
% Convert to double and rescale to [0,1]
images = double(images) / 255;

end

function labels = loadMNISTLabels(filename)

fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename, '']);

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2049, ['Bad magic number in ', filename, '']);

numItems = fread(fp, 1, 'int32', 0, 'ieee-be');
preLabels = fread(fp, inf, 'unsigned char');

labels = zeros([10 numItems]);
for i = 1 : length(preLabels)
    if preLabels(i) == 0
        labels(10,i) = 1;
    else
        labels(preLabels(i), i) = 1;
    end
end

fclose(fp);

end
```