Chingpo Lin
Amath482
HW4

Abstract

There are 60000 pictures of handwritten number from 0-9 for me to explore by using machine learning. I will using LDA try to identify some of digits and SVM to identify all. I will see how well they will perform on distinguish these digits.

Introduction and Overview

I was given two kinds of dataset one is training dataset and another is testing dataset. For each of dataset, I was given a 28x28x60000 matrix containing 60000 handwritten pictures and a 60000x1 array with the label which identify which digit it belongs to. To train the model of these data, I will first project my data into PCA space, which helps me to use LDA and SVM to train the data and then test how well the model is with testing dataset.

Theoretical Background

SVD analysis is an important aspect of PCA, we divide the matrix into three parts whose product (we need to do the reverse on the last element) equals the original matrix. First, doing the SVD will give us three matrix: U, S, V, which satisfy:

$$A = U\sum V^* \quad (1)$$

With U is m x m complex unitary matrix,

$\sum$ is m x n diagonal matrix,

V is n x n complex unitary

And for the projection onto principle components, we transform it into:

$$X = USV', \text{ (2) which equals}$$

$$U'X = SV' \text{ (3)}$$

After doing projection, we will introduce LDA, which is Linear Discriminant Analysis. For using LDA in 2 dataset, we use two matrix:

1. Between-class scatter matrix:

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \text{ (4)}$$

Which measures the variance between groups

2. Within-class scatter matrix:

$$S_W = \sum_{j=1}^{2} \sum_{x} (x - \mu_j)(x - \mu_j)^T \text{ (5)}$$

Which measures the variance within each group

Then, with both matrix, we can find the vector w

$$w = argmax \frac{w^T S_B w}{w^T S_W w} \text{ (6)}$$

This w can actually maximize the quotient by using the Eigenvector that corresponding to the largest Eigenvalue, then we have:

$$S_B w = \lambda S_w w \text{ (7)}$$

For the LDA problem in more group, we just need to simple replace above S_B and S_w by:

$$S_B = \sum_{j=1}^{N} (\mu_j - \mu)(\mu_j - \mu)^T \text{ (8)}$$

$$S_W = \sum_{j=1}^{N} \sum_{x} (x - \mu_j)(x - \mu_j)^T \quad (9)$$

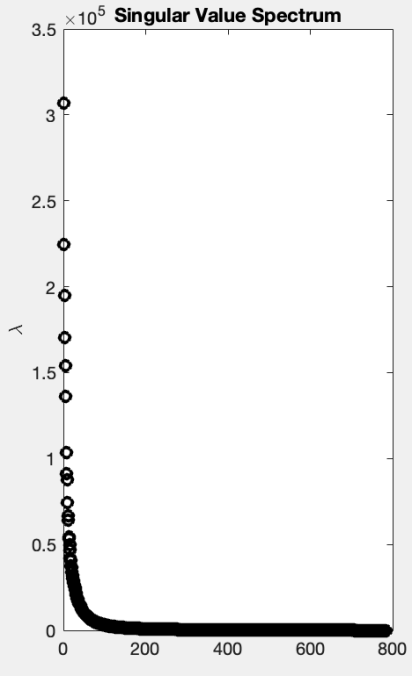
Algorithm Implementation and Development

First we read the image which is a 28x28x60000 matrix, and we reshape into 2D matrix which is 784x60000 with each column represent an image. Then we do a SVD analysis which will be used later. Then we see the change of lambda and find most of energy contained in first 100 ranks out of 784 by seeing the singular value spectrum. Then, we use the matrix U with three choosing column to see the 3D projection for each digit label. From here, we project our data into PCA space, and we want to build a classifier to train the individual digits. First I randomly pick two digits and get rid of unrelated column (not these two digits) by using find to pick the index with the label I choose. Then, I want to do the projection again by choosing more features. Here, I choose 10 as my feature. Then, I found the Sw and Sb by adapting the formula above, and use them to find our w vector. After that, I made a projection onto w, and set up the threshold to differentiate these two digits. From here, I have the training model, and can try to test how well it work for 2 digits. For three digits problem, I just need to replace the Sw and Sb by the multiple group formula and do the same thing. Next thing is to test how well the model perform in each two digit. I use for loop to test all possible combination, and use the same well to deal with test set matrix as we do in the train set. Using U to times the resulting image matrix, we get a test matrix, and then project

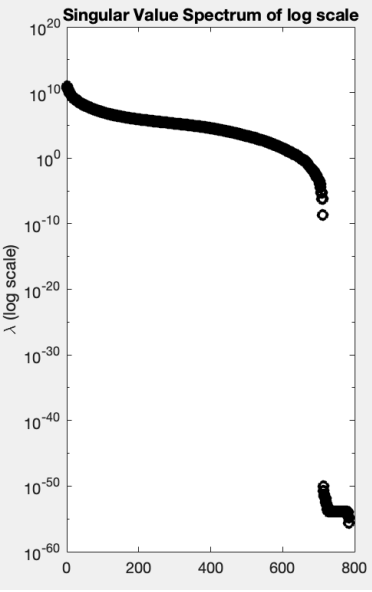
onto w, and the resulting vector will have Respond Vector and test label, and test label will give us the testing result so the sum of their difference vector is the error number, and divided by number of data we have is the error rate. Then, we will use SVM to see how well this can do in all 10 digits.

Computational Result

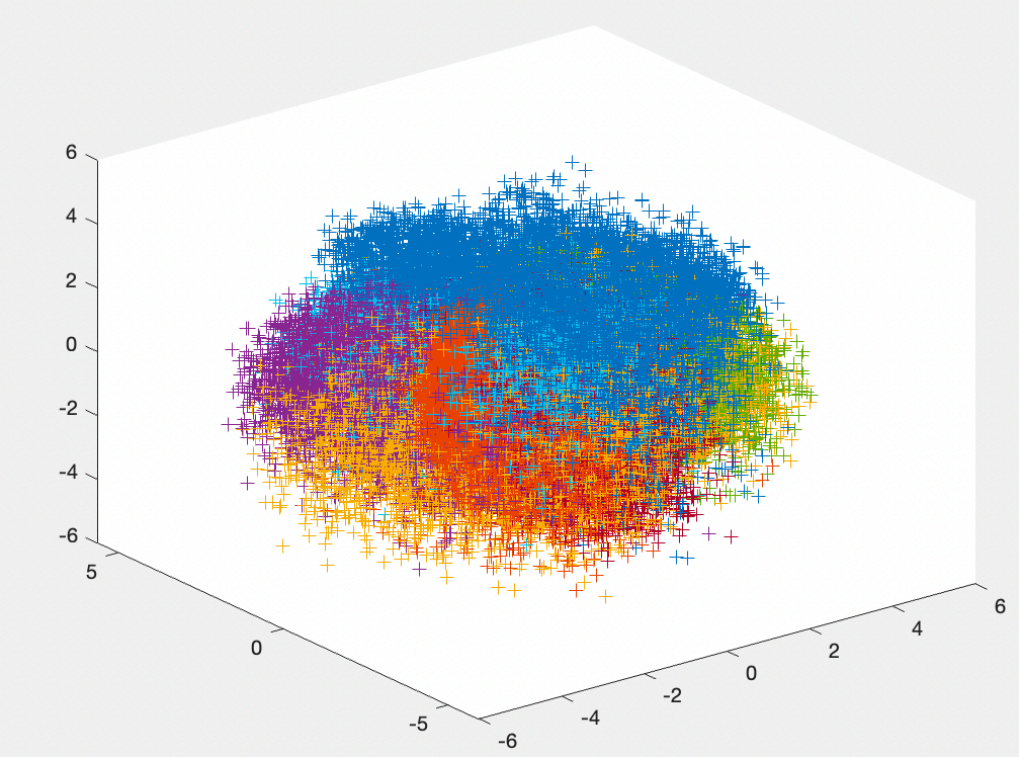
Graphs below are using 10 as feature number:



figure(1): singular value specturm



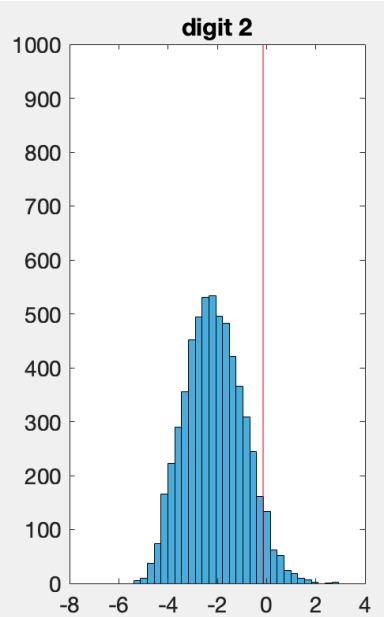
figure(2): singular value spectrum in log scale.



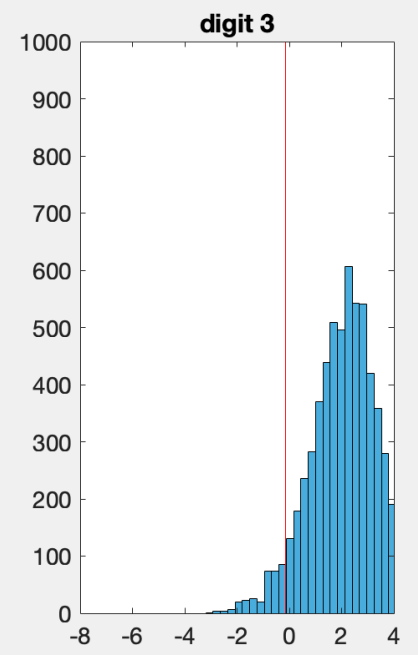
figure(3): 3d plot with selected 2,3,5 column with different color for each digit

The s in svd decomposition represents the energy proportion, and other work as the functionality as written in last part.

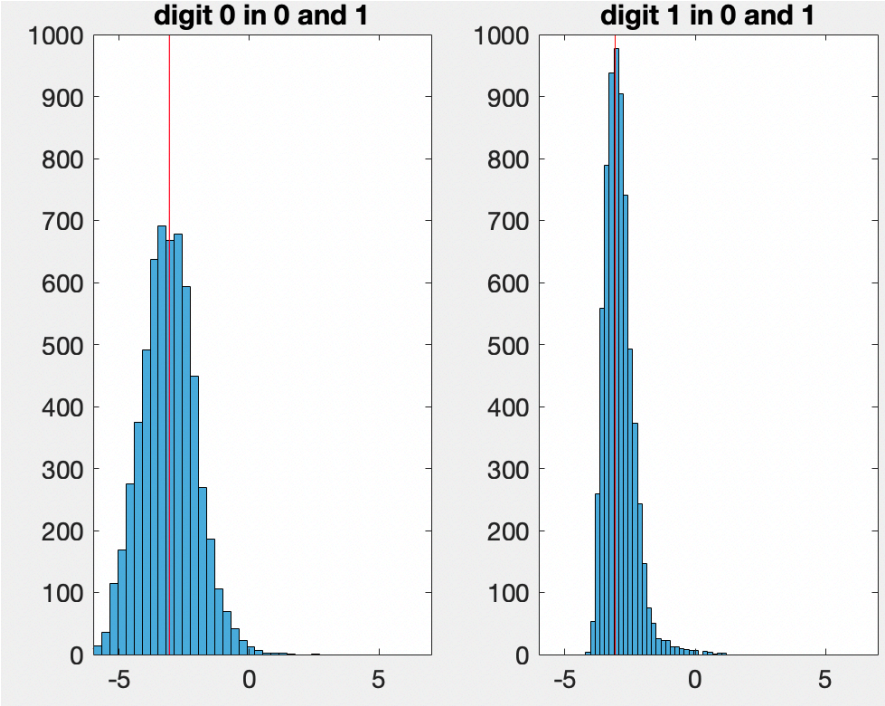
I try to make the histogram for LDA for 2 and 3 and LDA for 0, 1 and 2, which is:



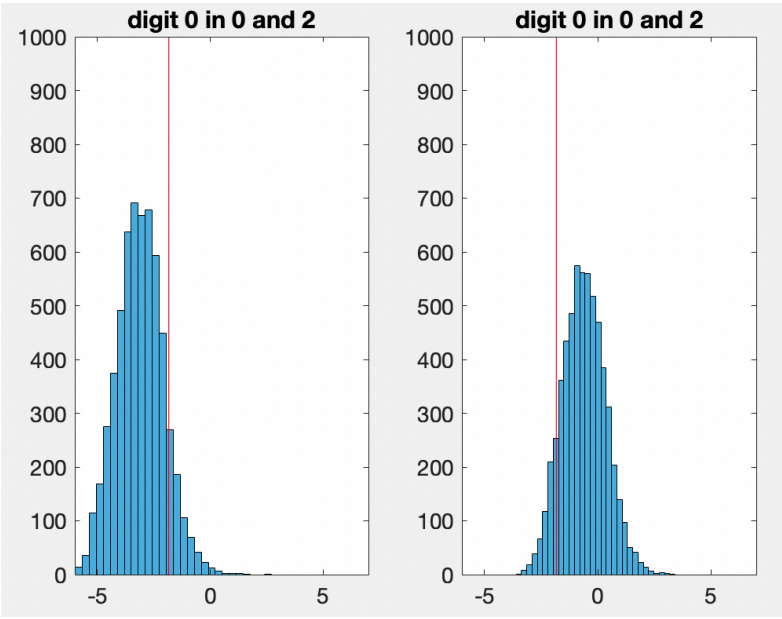
figure(4):histogram of digit 2 for LDA in digit 2 and 3



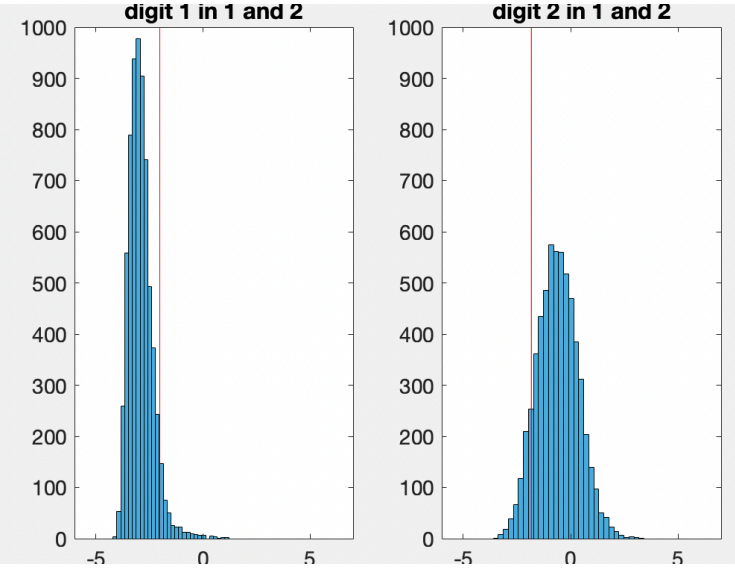
figure(5):histogram of digit 3 for LDA in digit 2 and 3



figure(6):histogram of LDA in digit 0 and 1 out of 0,1, and 2



figure(7):histogram of LDA in digit 0 and 2 out of 0,1, and 2

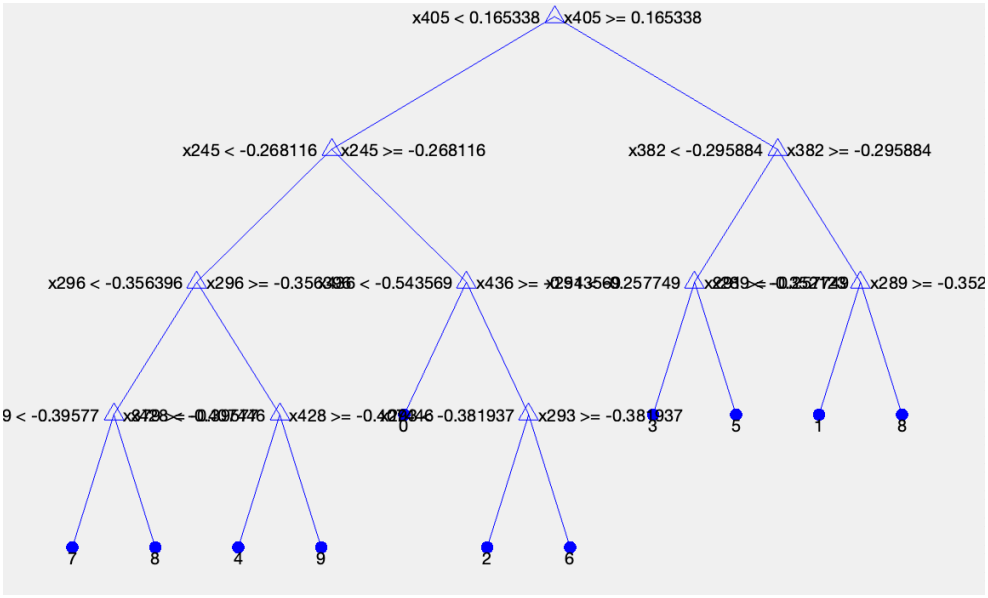


figure(8):histogram of LDA in digit 1 and 2 out of 0,1, and 2

By using a for loop in LDA to comparing each two number combination, I find that 0 and 1 has a highest success rate which is 0.9995, and 4 and 9 has a lowest which is 0.9141. Also, I use the train set on the model generated by train set and get 0.9892 which is greater than 0.9496 which generated by using test set on the model.

here is decision tree, and svm has 0.94 success rate

Then, I run the svm on the 10 digits, it just has a 0.5937 success rate overall.



figure(9): decision tree

When I use the same method to compare the svm and decision tree, svm range from 97 percent to 99.6 percent, which decision tree has range from 92 percent to 99.9 percent

Summary and Conclusions

From the data above, svm seems to have a stable performance although it has a worse performance on easiest case, but it work better overall. Also, LDA did a great job in two digits, but when we use 3 or more digits, the success rate seems to decline a lot.

Appendix A - function in code

Im2double: convert all value to double
mean: get the mean of given array or matrix
find: find the index of given condition
Sum: get the sum
zeros: create a vector or matrix with all 0 in given dimension
svd: get the svd matrix decomposition

Appendix B - Matlab code

```
%% Clean workspace
clear all; close all; clc

%% loading training set
[images1, labels1] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte');
%% loading tesing set
[images2, labels2] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
%% projection into PCA space
row_size = size(images1, 1) * size(images1, 2);
image = reshape(images1, row_size, size(images1, 3));
image = im2double(image);
meandata = mean(image, 2);
image = image - repmat(meandata,1,size(images1, 3));
row_size2 = size(images2, 1) * size(images2, 2);
images2 = reshape(images2, row_size2, size(images2, 3));
images2 = im2double(images2);
images2 = images2-repmat(meandata,1,size(images2, 3));
[U,S,V] = svd(image, 'econ');
sing_val = diag(S).^2;
lambda = sing_val.^2;
```

```
figure(1)
subplot(1,2,1)
plot(sing_val, 'ko','Linewidth',2)
ylabel('\lambda');
title("Singular Value Spectrum");
subplot(1,2,2)
semilogy(lambda,'ko','Linewidth',2)
ylabel('\lambda (log scale)');
title("Singular Value Spectrum of log scale");

figure(2)
p = U(:,[2,3,5])'*image;
for count = 1:10
   index=find(labels1==count - 1);
   pro=p(:,index);
   plot3(pro(1,:),pro(2,:),pro(3,:),'+');
   hold on
end


%% two digits train
feature = 10;
index1 = find(labels1 == 2)';
index2 = find(labels1 == 3)';
d1set = image(:,index1);
d2set = image(:,index2);
[U,vfirst,vsecond,threshold,w,sortn1,sortn2] = two_trainer(d1set,d2set,index1,index2,feature);
% plot
%plot(vfirst,zeros(size(vfirst,2)),'ob','Linewidth',2)
%hold on
%plot(vsecond,ones(size(vsecond,2)),'dr','Linewidth',2)
%ylim([0 1.2])

subplot(1,2,1)
histogram(sortn1,30); hold on, plot([threshold threshold], [0 1000],'r')
set(gca,'Xlim',[-8 4],'Ylim',[0 1000],'Fontsize',14)
title('digit 2')
subplot(1,2,2)
histogram(sortn2,30); hold on, plot([threshold threshold], [0 1000],'r')
set(gca,'Xlim',[-8 4],'Ylim',[0 1000],'Fontsize',14)
title('digit 3')
%% two digits train test
indext1 = find(labels1 == 2)';
indext2 = find(labels1 == 3)';
image3 = image(:,[indext1 indext2]);
TestNum = 60000;
TestMat = U'*image3;
pval = w'*TestMat;
ResVec = (pval > threshold);
testLabel = zeros(1,size(indext1,2)+size(indext2,2));
for i = 1:size(indext2,2)
   testLabel(size(indext1,2)+i) = 1;
end
err = abs(ResVec - testLabel);
```

```
errNum = sum(err);
sucRate = 1 - errNum/TestNum;
%% two digits test
indext1 = find(labels2 == 2)';
indext2 = find(labels2 == 3)';
image2 = images2(:,[indext1 indext2]);
TestNum = size(image2,2);
TestMat = U'*image2;
pval = w'*TestMat;
ResVec = (pval > threshold);
testLabel = zeros(1,size(indext1,2)+size(indext2,2));
for i = 1:size(indext2,2)
   testLabel(size(indext1,2)+i) = 1;
end
err = abs(ResVec - testLabel);
errNum = sum(err);
sucRate = 1 - errNum/TestNum;
%% finding most/least accurate
feature = 10;
highi1 = 0;
highi2 = 0;
lowi1 = 0;
lowi2 = 0;
highsr = 0;
lowsr = 1;
for i = 1:10
   for j = i+1:10
      index1 = find(labels1 == i-1)';
      index2 = find(labels1 == j-1)';
      d1set = image(:,index1);
      d2set = image(:,index2);
      [U,vfirst,vsecond,threshold,w,sortn1,sortn2] = two_trainer(d1set,d2set,index1,index2,feature);
      indext1 = find(labels2 == i-1)';
      indext2 = find(labels2 == j-1)';
      image2 = images2(:,[indext1 indext2]);
      TestNum = size(image2,2);
      TestMat = U'*image2;
      pval = w'*TestMat;
      ResVec = (pval > threshold);
      testLabel = zeros(1,size(indext1,2)+size(indext2,2));
      for index3 = 1:size(indext2,2)
         testLabel(size(indext1,2)+index3) = 1;
      end
      err = abs(ResVec - testLabel);
      errNum = sum(err);
      sucRate = 1 - errNum/TestNum;
      if sucRate > highsr
         highi1 = i-1;
         highi2 = j-1;
         highsr = sucRate;
      end
      if sucRate < lowsr
         lowi1 = i-1;
         lowi2 = j-1;
```

```
            lowsr = sucRate;
        end
    end
end

%% svm
% classification tree on fisheriris data
load fisheriris;
tree = fitctree(image',labels1, 'MaxNumSplits',10,'CrossVal','on');
view(tree.Trained{1},'Mode','graph');
classError = kfoldLoss(tree, 'mode', 'individual');
[~, k] = min(classError);
testlabels = predict(tree.Trained{k}, images2');
err_tree = immse(testlabels, labels2);
correct = find((testlabels - labels2) == 0);
sRt = size(correct) / size(labels2);
%% three digit
i1 = find(labels1 == 0)';
i2 = find(labels1 == 1)';
i3 = find(labels1 == 2)';
ds1 = image(:,i1);
ds2= image(:,i2);
ds3 = image(:,i3);
[U,v1,v2,v3,t12,t13,t23,w,s1,s2,s3] = three_trainer(ds1,ds2,ds3,i1,i2,i3,20);

subplot(1,2,1)
histogram(s2,30); hold on, plot([t23 t23], [0 1000],'r')
set(gca,'Xlim',[-6 7],'Ylim',[0 1000],'Fontsize',14)
title('digit 1 in 1 and 2')
subplot(1,2,2)
histogram(s3,30); hold on, plot([t13 t13], [0 1000],'r')
set(gca,'Xlim',[-6 7],'Ylim',[0 1000],'Fontsize',14)
title('digit 2 in 1 and 2')
%% functions
function [U,vfirst,vsecond,threshold,w,sortn1,sortn2] = two_trainer(d1set,d2set,index1,index2,feature)
    [U,S,V] = svd([d1set d2set],'econ');
    U = U(:,1:feature);
    digit_proj = S*V';
    d1 = digit_proj(1:feature,1:size(index1,2));
    d2 = digit_proj(1:feature,size(index1,2)+1:size(index1,2)+size(index2,2));

    % scatter matric
    md1 = mean(d1,2);
    md2 = mean(d2,2);

    Sw = 0;
    for k = 1:size(index1,2)
        Sw = Sw+(d1(:,k)-md1)*(d1(:,k)-md1)';
    end
    for k = 1:size(index2,2)
        Sw = Sw+(d2(:,k)-md2)*(d2(:,k)-md2)';
    end
    Sb = (md1-md2)*(md1-md2)';
```

```
    % find w
    [V2, D] = eig(Sb, Sw);
    [~, ind] = max(abs(diag(D)));
    w = V2(:,ind);
    w = w/norm(w,2);

    % project onto w
    vfirst = w'*d1;
    vsecond = w'*d2;

    % setup threshold
    if mean(vfirst) > mean(vsecond)
        w = -w;
        vfirst = -vfirst;
        vsecond = -vsecond;
    end

    sortn1 = sort(vfirst);
    sortn2 = sort(vsecond);
    t1 = length(sortn1);
    t2=1;
    while sortn1(t1) > sortn2(t2)
        t1 = t1 - 1;
        t2 = t2 + 1;
    end
    threshold = (sortn1(t1) + sortn2(t2))/2;
end

function [U,v1,v2,v3,t12,t13,t23,w,s1,s2,s3] = three_trainer(ds1,ds2,ds3,i1,i2,i3,feature)
    [U,S,V] = svd([ds1 ds2 ds3],'econ');
    U = U(:,1:feature);
    digit_proj = S*V';
    d1 = digit_proj(1:feature,1:size(i1,2));
    d2 = digit_proj(1:feature,size(i1,2)+1:size(i1,2)+size(i2,2));
    d3 = digit_proj(1:feature,size(i1,2)+size(i2,2)+1:size(i1,2)+size(i2,2)+size(i3,2));

    % scatter matric
    md1 = mean(d1,2);
    md2 = mean(d2,2);
    md3 = mean(d3,2);
    mda = [md1 md2 md3];

    Sw = 0;
    for k = 1:size(i1,2)
        Sw = Sw+(d1(:,k)-md1)*(d1(:,k)-md1)';
    end
    for k = 1:size(i2,2)
        Sw = Sw+(d2(:,k)-md2)*(d2(:,k)-md2)';
    end
    for k = 1:size(i3,2)
        Sw = Sw+(d3(:,k)-md3)*(d3(:,k)-md3)';
    end
    Sb = (md1-md2)*(md1-md2)';
    meanall = (md1 + md2 + md3) / 3;
```

```
    Sb = Sb + (mda(:,1)-meanall)*(mda(:,1)-meanall)';
    Sb = Sb + (mda(:,2)-meanall)*(mda(:,2)-meanall)';
    Sb = Sb + (mda(:,3)-meanall)*(mda(:,3)-meanall)';

    % find w
    [V2, D] = eig(Sb, Sw);
    [~, ind] = max(abs(diag(D)));
    w = V2(:,ind);
    w = w/norm(w,2);

    % project onto w
    v1 = w'*d1;
    v2 = w'*d2;
    v3 = w'*d3;

    % setup threshold
    if mean(v1) > mean(v2)
        w = -w;
        v1 = -v1;
        v2 = -v2;
    end
    if mean(v2) > mean(v3)
        w = -w;
        v2 = -v2;
        v3 = -v3;
    end

    s1 = sort(v1);
    s2 = sort(v2);
    s3 = sort(v3);
    t1 = length(s1);
    t2=1;
    while s1(t1) > s2(t2)
        t1 = t1 - 1;
        t2 = t2 + 1;
    end
    t12 = (s1(t1) + s2(t2))/2;
    t1 = length(s1);
    t3=1;
    while s1(t1) > s3(t3)
        t1 = t1 - 1;
        t3 = t3 + 1;
    end
    t13 = (s1(t1) + s3(t3))/2;
    t2 = length(s2);
    t3=1;
    while s2(t2) > s3(t3)
        t2 = t2 - 1;
        t3 = t3 + 1;
    end
    t23 = (s2(t2) + s3(t3))/2;
end
```