Classifying Digits

AMATH 482 HW4

Xuanhui Chen

Abstract

This report aims to classify different digits by using the linear discriminant analysis (LDA) on a provided an extremely useful dataset called MNIST that consists of a large amount of digital data of images. It starts by doing an SVD analysis of the digit images and projecting the data into PCA (principle component analysis) space. Then we will build a classifier to identify individual digits in the training dataset. Finally, the performance of the classifiers on both the training and test datasets will be discussed.

I. Introduction and Overview

We are given the MNIST dataset, which contains 60,000 training images with corresponding classifying labels as well as 10,000 testing ones. With the image recognition techniques, we are going to present a data-driven approach (machine learning). Our goal is to perform an analysis (SVD) of the dataset and build classifiers with linear discriminant analysis (LDA) to distinguish each individual digit in the dataset. And we will find out how effective these classifiers are.

II. Theoretical Background

Singular Value Decomposition(SVD)

The Singular Value Decomposition is in the form of:

$$\mathbf{A} = \mathbf{U\Sigma V}^*, \quad (1)$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, and $\mathbf{\Sigma}$ is a diagonal matrix with σ_j on its diagonals.
One important property of SVD is: Every matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ has an SVD.

Principle Component Analysis (PCA)

The Principle Component Analysis (PCA) is a method that analyzes the low-rank approximations geometrically, while the SVD can be used to produce low-rank approximations of a dataset. Here we can see the connections between these two.

Linear Discriminant Analysis (LDA)

The goal of LDA is to find a suitable projection that maximizes the distance between the inter-class while minimizing the intra-class data:

- LDA for Two Datasets:
  The between-class scatter matrix is defined as:

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T, \quad (2)$$

where μ are means of each group for each feature. It is a measure of the variance between group means.

The within-class scatter matrix that measures the variance within each group is defined as:

$$\mathbf{S}_w = \boldsymbol{\Sigma}_{j=1}^{2} \boldsymbol{\Sigma}_{\mathbf{x}} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T. \quad (3)$$

We can then solve for the following vector **w:**

$$\mathbf{w} = \text{argmax} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}} \quad (4)$$

with

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}. \quad (5)$$

- LDA for More Datasets:

  The between-class scatter matrix is defined as:

$$\mathbf{S}_B = (\mu_j - \mu)(\mu_j - \mu)^T, \quad (6)$$

where μ is the overall mean and μ_j is the mean of the $N \geq 3$ groups. each feature.

And the within-class scatter can be defined as:

$$\mathbf{S}_w = \boldsymbol{\Sigma}_{j=1}^{N} \boldsymbol{\Sigma}_{\mathbf{x}} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T. \quad (7)$$

III. Algorithm Implementation and Development

Part I

- Load the given MNIST dataset:
  - Use the mnist_parse(X,Y) function to load the data and reshape it into column vectors. (See **Appendix B** Part I and minist_parse function.)
- Do an SVD analysis of the digits images. (See **Appendix B** Part I: SVD analysis.)
- Draw the singular value spectrum and compute the rank of digit space:
  - Plot the singular value spectrum for both sigma and energy. (See **Appendix B** Part I.)
  - Compute the sum of energy for each point until it reaches 90%. The number of points is the rank of digit space. (See **Appendix B** Part I.)
- Draw 3-D projection onto selected 2nd, 3rd, and 5th V-modes. (See **Appendix B** Part I.)

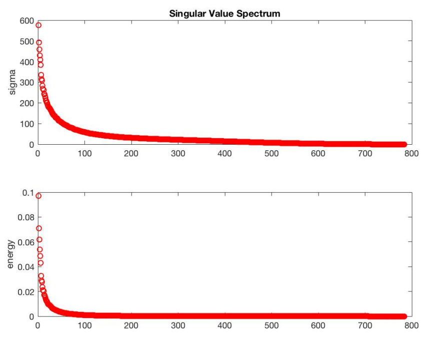
Part II

- Build a linear classifier with LDA for two or three randomly picked digits:
    - Extract data of the selected digits with the digitdata function. (See **Appendix B** digitdata function.)
    - Calculate the scatter matrices inside the trainer function. (See **Appendix B** trainer1 function.)
    - Find vector **w** using built-in MATLAB function eig()(See **Appendix** A and Appendix **B** trainer1 function.)
    - Project onto vector **w** by multiplying by **w′**. (See **Appendix B** trainer1 function.)
    - Set a threshold for the classifier. (See **Appendix B** getThreshold function.)
    - Plot the histogram of each digit. Draw a vertical line for the decision-making threshold. (See **Appendix B** Part II.)
- Quantify the accuracy of the separation with LDA on the test data to determine the difficulties of separating two digits or three digits in the dataset (See **Appendix B** Part III):
    - Do the PCA projection on the test, and then project onto the line we found from LDA.
    - Check which the classification of selected digits. For the two-digit combination, 0 represents the first digit, and 1 represents the second. For the three-digit combinations, 1 represents the first digit, 2 represents the second digit, and 3 represents the third. Check if these digits are below the threshold or not.
    - Check the errors and the success rate on the test data.
- Compute the accuracies of support vector machines (SVM) and decision tree classifiers and compare with the LDA algorithm.

IV. Computational Results

4.1 Singular Spectrum

As we can see in the Figure 1, the first mode is dominant, while there is no sharp drop off in the singular values. The value of r, which is the rank of the digit space, is 87, meaning these 87 modes are necessary for good image construction. The **V** matrix is represented as the pattern of the digits.



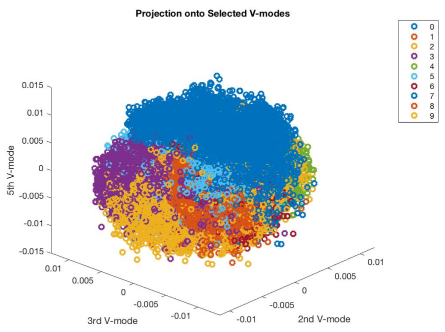
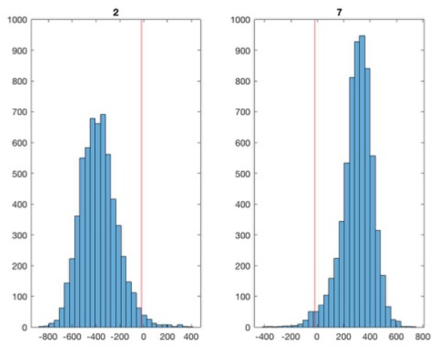


Figure 1(left): Singular value spectrum with sigma y-axis on the top and energy y-axis at the bottom.

Figure 2 (right): Projection onto the 2nd, 3rd, 5th V-Modes.



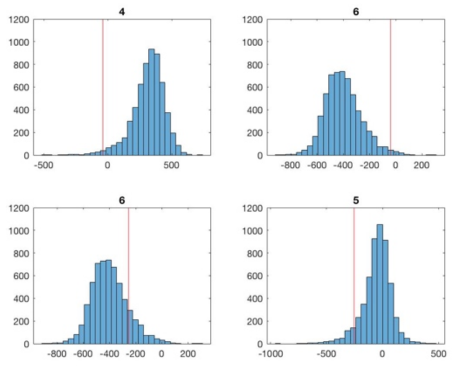


Figure 3 (left): Histograms of digit 2 and 7 where the red line represents the decision-making threshold.

Figure 4 (right): Histograms of digit 4, 5 and 6 where the red line represents the decision-making threshold.

4.2 3-D projection onto selected V-modes: 2nd, 3rd, 5th

The data seems to be hard to classify because they are intersected and tangled with each other, as shown in Figure 2. The ideal caricature for LDA will show completely separate data.

4.3 LDA on two digits and three digits:

From Figure 3, we can see there are only 8 or 9 wrong, which are separated by the threshold line, out of 87 for the two chosen digits: 2 and 7. The quantified accuracy of the separation with LDA on the test data is 0.9757, which means a 97.57% accuracy. On the other hand, figure 4 also shows only few wrong classifications for the three selected digits: 4, 5, and 6. The quantified accuracy of the separation with LDA on the test data is 0.9431, which means a 94.31% accuracy. This means building a LDA can reasonably identify the two-digit-combination of 2 and 7 as well as the three-digit-combination of 4, 5, and 6.

4.4 Accuracy of applying LDA on two digits and three digits:

After applying the LDA on all combinations of two digits, the highest is 0.9970, which means a 99.70% accuracy. This happens when identifying number 6 and 7. After applying the LDA on all combinations of three digits, the highest is 0.9985, which means a 99.85% accuracy. This happens when identifying number 5, 6 and 7.

V. Summary and Conclusion

In conclusion, the machine learning and LDA is helpful for identifying digits. We have pretty good accuracies overall.

Appendix A: MATLAB functions used and brief implementation explanation

- **M = size(X,DIM)** returns the length of the dimension specified the scalar DIM.
- **[U,S,V] = svd(X,'econ')**: produces a diagonal matrix S, of the same as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that X = U*S*V', which is the "economy size" decomposition.
- **diag(X):** returns the main diagonal of X.
- **S = sum(X)**: the sum of the elements of the vector X.
- **I = find(X)** returns the linear indices corresponding to the nonzero entries of the array X (the condition).
- **E = eig(A)**: produces a column vector E containing the eigenvalues of a square matrix A.
- **norm(X,2)**: returns the 2-norm of X.
- **B = sort(A)**: sorts in ascending order.

Appendix A: MATLAB Codes

```
clear all; close all; clc;

% Part I: Loading mnist data and reshape it into column vectors
[traindata, traingnd] = mnist_parse('train-images.idx3-ubyte', ...
    'train-labels.idx1-ubyte');
traindata = double(reshape(traindata, ...
    size(traindata,1)*size(traindata,2), []));
traingnd = double(traingnd);

[testdata, testgnd] = ...
    mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte');
testdata = ...
    double(reshape(testdata, size(testdata,1)*size(testdata,2), []));
testgnd = double(testgnd);

% Part I: SVD analysis
[m,n]= size(traindata);
train = traindata-repmat(mean(traindata,2),1,n);
[U,S,V]= svd(train/sqrt(n-1), 'econ');
sig = diag(S);
energy = sig.^2/sum(sig.^2);

% Part I: Singular value spectrum and rank r
figure(1)
subplot(2,1,1)
plot(sig,'ro','Linewidth',0.3)
ylabel('sigma')
%set(gca,'Fontsize',16,'Xlim',[0 80])
title('Singular Value Spectrum')
subplot(2,1,2)
plot(energy ,'ro','Linewidth',0.3)
ylabel('energy')
%set(gca,'Fontsize',16,'Xlim',[0 80])

total_energy = 0;
for i = 1:length(energy)
    total_energy = total_energy + energy(i);
    if total_energy > 0.9
        break
    end
end
rank = i;

% Part I: 3-D projection onto selected V-modes
figure(2)
for index = 0:9
    label = find(traingnd == index);
    plot3(V(label, 2), V(label, 3), V(label, 5),...
        'o', 'DisplayName', sprintf('%i',index), 'Linewidth', 2)
    hold on
end
xlabel('2nd V-mode'), ylabel('3rd V-mode'), zlabel('5th V-mode')
title('Projection onto Selected V-modes')
legend
set(gca,'Fontsize', 10)

% Part II: LDA on two digits
[d1, n1] = digitdata(traindata, traingnd,2);
```

```matlab
[d2, n2] = digitdata(traindata, traingnd,7);
[U,S,V,w,s1,s2] = trainer1(d1,d2,rank);
threshold = getThreshold(s1,s2);

figure(2)
subplot(1,2,1)
histogram(s1,30); hold on, plot([threshold threshold], [0 1000],'r')
title('2')
subplot(1,2,2)
histogram(s2,30); hold on, plot([threshold threshold], [0 1000],'r')
title('7')

% Part II: accuracy of LDA on separating the two digits
[t1, l1] = digitdata(testdata,testgnd,2);
[t2, l2] = digitdata(testdata,testgnd,7);
testD = [t1 t2];

testL = zeros(l1+l2,1);
testL(l1+1:l1+l2) = 1;

testNum = size(testD,2);
testMat = U'*testD;
pval = w'*testMat;
resVec = (pval > threshold);
err = abs(resVec - testL');
errNum = sum(err);
sucRate = 1 - errNum/testNum;


% Part II: LDA on three digits
[d1,n1] = digitdata(traindata, traingnd,4);
[d2,n2] = digitdata(traindata, traingnd,5);
[d3,n3] = digitdata(traindata, traingnd,6);
[U,S,V,w,s1,s2,s3] = trainer2(d1,d2,d3,rank);
threshold1 = getThreshold(s2,s3);
threshold2 = getThreshold(s3,s1);



figure(3)
subplot(2,2,1)
histogram(s1,30); hold on, plot([threshold2 threshold2], [0 1200],'r')
title('4')
subplot(2,2,2)
histogram(s3,30); hold on, plot([threshold2 threshold2], [0 1200],'r')
title('6')
subplot(2,2,3)
histogram(s3,30); hold on, plot([threshold1 threshold1], [0 1200],'r')
title('6')
subplot(2,2,4)
histogram(s2,30); hold on, plot([threshold1 threshold1], [0 1200],'r')
title('5')

% Part III: Accuracy on two digits
[accuracy2,row,col] = acc2(traindata,traingnd,testdata,testgnd,rank);

% Part III: Accuracy on three digits
[accuracy3,num1,num2,num3] = acc3(traindata,traingnd,testdata,testgnd,rank);
```

```
% Part IV: SVM and decision tree classifiers
digits=fitctree(traindata,trainlabel,'MaxNumSplits',3,'CrossVal','on');
view(tree.Trained{1},'Mode','graph');
classError = kfoldLoss(tree);

% SVM classifier with training data, labels and test set
Mdl = fitcsvm(xtrain,label);
test_labels = predict(Mdl,test);



function [U,S,V,w,s1,s2] = trainer1(d1_data,d2_data,feature)
    n1 = size(d1_data,2);
    n2 = size(d2_data,2);
    [U,S,V] = svd([d1_data d2_data],'econ');
    digits = S*V';
    U = U(:,1:feature);
    d1 = digits(1:feature,1:n1);
    d2 = digits(1:feature,n1+1:n1+n2);
    m1 = mean(d1,2);
    m2 = mean(d2,2);

    Sw = 0;
    for i = 1:n1
        Sw = Sw + (d1(:,i)-m1)*(d1(:,i)-m1)';
    end
    for i = 1:n2
        Sw = Sw + (d2(:,i)-m2)*(d2(:,i)-m2)';
    end
    Sb = (m1-m2)*(m1-m2)';

    [V2,D] = eig(Sb,Sw);
    [~,ind] = max(abs(diag(D)));
    w = V2(:,ind);
    w = w/norm(w,2);
    v1 = w'*d1;
    v2 = w'*d2;

    if mean(v1)>mean(v2)
        w = -w;
        v1 = -v1;
        v2 = -v2;
    end

    s1 = sort(v1);
    s2 = sort(v2);
end
```

```matlab
function [U,S,V,w,s1,s2,s3] = trainer2(d1_data,d2_data,d3_data,feature)
    n1 = size(d1_data,2);
    n2 = size(d2_data,2);
    n3 = size(d3_data,2);
    [U,S,V] = svd([d1_data d2_data d3_data],'econ');
    digits = S*V';
    U = U(:,1:feature);
    d1 = digits(1:feature,1:n1);
    d2 = digits(1:feature,n1+1:n1+n2);
    d3 = digits(1:feature,n1+n2+1:n1+n2+n3);
    m1 = mean(d1,2);
    m2 = mean(d2,2);
    m3 = mean(d3,2);
    mtotal = mean([m1 m2 m3],2);

    Sw = 0;
    for i = 1:n1
        Sw = Sw + (d1(:,i)-m1)*(d1(:,i)-m1)';
    end
    for i = 1:n2
        Sw = Sw + (d2(:,i)-m2)*(d2(:,i)-m2)';
    end
    for i = 1:n3
        Sw = Sw + (d3(:,i)-m3)*(d3(:,i)-m3)';
    end
    Sb = (m1-mtotal)*(m1-mtotal)'+(m2-mtotal)*(m2-mtotal)'+...
        (m3-mtotal)*(m3-mtotal)';

    [V2,D] = eig(Sb,Sw);
    [~,ind] = max(abs(diag(D)));
    w = V2(:,ind);
    w = w/norm(w,2);
    v1 = w'*d1;
    v2 = w'*d2;
    v3 = w'*d3;

    s1 = sort(v1);
    s2 = sort(v2);
    s3 = sort(v3);
end
```

```matlab
function [digit,num] = digitdata(data, gnd, d)
    index = 1;
    for i = 1:length(gnd)
        if gnd(i) == d
            digit(:,index) = data(:,i);
            index = index + 1;
        end
    end
    num = index-1;
end
```

```
function threshold = getThreshold(s1,s2)
    t1 = length(s1);
    t2 = 1;
    while s1(t1) > s2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold = (s1(t1)+s2(t2))/2;
end
```

```
function [accuracy2,row,col] = acc2(traindata,traingnd,testdata,testgnd,rank)
     for i=1:9
        [d1,~] = digitdata(traindata, traingnd,i);
        [t1,l1] = digitdata(testdata,testgnd,i);
        for j=1:9
            if j<=i
                accuracy2(j,i) = 0;
                continue
            end
            [d2,~] = digitdata(traindata, traingnd,j);
            [U,~,~,w,s1,s2] = trainer1(d1,d2,rank);
            threshold = getThreshold(s1,s2);

            [t2, l2] = digitdata(testdata,testgnd,j);
            testD = [t1 t2];

            testL = zeros(l1+l2,1);
            testL(l1+1:l1+l2) = 1;

            testNum = size(testD,2);
            testMat = U'*testD;
            pval = w'*testMat;
            resVec = (pval > threshold);
            err = abs(resVec - testL');
            errNum = sum(err);
            sucRate = 1 - errNum/testNum;

            accuracy2(j,i) = sucRate;
        end
     end

    % find best match number
    maximum = max(max(accuracy2));
    [row,col]=find(accuracy2==maximum);
end
```

Reference

1. Course note.
2. MATLAB help section.