AMATH 482 Homework 4

Clayton Heath

March 10, 2021

Abstract

We use the *Singular Value Decomposition* (SVD), *Principal Component Analysis* (PCA), and *Linear Discriminant Analysis* (LDA) along with set of training images to train our computer to classify images of the ten digits, which we test using a set of testing images. We then compare how accurate the LDA is compared to SVM and decision tree methods of classification.

1 Introduction and Overview

We will first import our training images and labels as well as our testing images and labels. We will then apply SVD to our training image matrix, and then use the decomposition in order to use LDA to train our computer to classify the images. We will then compare how well the LDA works as compared to SVM and decision tree for classification.

2 Theoretical Background

Linear Discriminant Analysis (LDA) finds a suitable projection that maximizes the distance between the inter-class data while minimizes the distance between intra-class data. For using LDA on 2 data sets, we first calculate the means of each of our groups for each of the features. This gives us two cloumn vectors μ_1 and μ_2. Then, we have the between-class scatter matrix:

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

This gives us the variance between the groups. Then we can define the within-class scatter matrix

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

This gives us the variance within each group. We then want to find some vector $\mathbf{w}$ such that

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

And, the vector $\mathbf{w}$ that maximizes the above quotient is the eigenvector corresponding to the largest eigenvalue of

$$\mathbf{S}_b \mathbf{w} = \lambda \mathbf{S}_\omega \mathbf{w} \tag{4}$$

Then after solving this, we can choose some cutoff between the two data sets, such that when we are giving some piece of data, we can assign to either of the two groups.

3 Algorithm Implementation and Development

First, under the `Prep` section of the MATLAB code, we import the training images and labels as well as the testing images and labels. Then under the `Reshape` section, we take the $28 \times 28 \times 60000$ matrix of the training images, and turn each of the 60000 28×28 images, and turn them into column vectors of length

784. This gives us a 784×60000 matrix. Now we reorganize this matrix so that all the images of 0 are first, then 1, and so on all the way to 9, we also reorganize our labels to match this as well.

Next, under the `SV Spectrum` section, we begin preparing our new training images matrix for SVD. We do this by subtracting the mean. We then use the `svd()` function on the matrix. Then we plot the singular values as a percent of the total information as seen in Figure 1. Looking at this, we take 70 modes to be good for image reconstruction, which we calculate to contain over 50% of the information. `U` is our PCA modes, Σ is energies captured (singular values) and `V` is coefficients for reconstructing the images.

Next, we plot the projection onto the 2nd, 3rd, and 5th columns of the V matrix. The indices used came from when we reorganized the matrix and found how many images of each digit there were. We can see this plot in Figure 2.

Next, under the `Testing Prep` section, we reshape and reorganize the testing images matrix and label vector. Then, we project the test images by multiplying it by our `U(:,1:70)'` (only using the first 70 columns since we determined 70 modes was good enough for reconstruction). Then we use the `classify()` function to do LDA as described in the Theoretical Background section. We first use `classify()` for the digits 0 and 1, thus we are only using a section of our projected testing images matrix. We also only use a section of our V matrix for training for same reason, as well as only the first 70 columns just as before. The results from this test can be seen in Figure 3. Similarly we use classify for the digits 0,1, and 2, and the results are seen in Figure 4. We also calculate the accuracy of these test by using the labels given for the testing images.

Finally, we used SVM and decision tree classifiers for all ten digits, 0 and 1, 1 and 2, and 0 and 2; calculating their accuracy.

4 Computational Results

Looking at the singular value spectrum in Figure 1, we take all of the values before it "elbows", which we found to be the first 70. We then found that these first 70 singular values contain 50.31% of the total information.

In Figure 3, we see the results of our LDA for the digits 0 and 1. We can spot a few miss-classified images on the graph, but it appears to be very accurate. This is confirmed when calculating its accuracy, we have 99.86%. Then in Figure 4, we have our LDA for three digits, 0,1, and 2. Here we see quite a bit more miss-classification, however, it still seems very accurate. In fact, we get 93.61% accuracy. Looking at the graph more closely, we see quite a few 1's being identified as 2's and quite a few 2's being identified as 1's, with very few miss-identifications involving 0. This makes sense since 0 is much more round than 1 and 2, and to me seems like a more distinct digit out of the three. So I would understand why the computer would struggle more telling the difference between 1 and 2.

Still looking at Figure 4, we see only one 0 was miss-classified as a 1 and zero 1's were miss-classified as a 0. This makes sense since 1 and 0 have very different traits, 0 is very round, while 1 is pretty much just a straight line, making it easy to differentiate.

The SVM for all 10 digits gave a 72.55% accuracy on the test data, and the decision tree gave 39.04% accuracy on the test data.

For telling the difference between 0 and 1, LDA has a 99.86% accuracy, SVM has a 99.43%, and the decision tree has a 88.7%. Here LDA has the best, while SVM is very close behind, and the decision tree is far behind both of them.

For telling the difference between 1 and 2, LDA has a 98.94% accuracy, SVM has a 93.82%, and the descision tree has a 89.89% accuracy. Here we see both LDA and SVM got worse than the 'easier' to differentiate 0 and 1, while the decision tree, while still having a worse accuracy relative to LDA and SVM, got better when compared to how it performed on the 0 and 1.

5 Summary and Conclusions

We were able to successfully a linear classifier (LDA) to classify images of digits. We then compared its accuracy to that of SVM and decision tree classifications, in which LDA showed itself to be the most accurate from out testing.

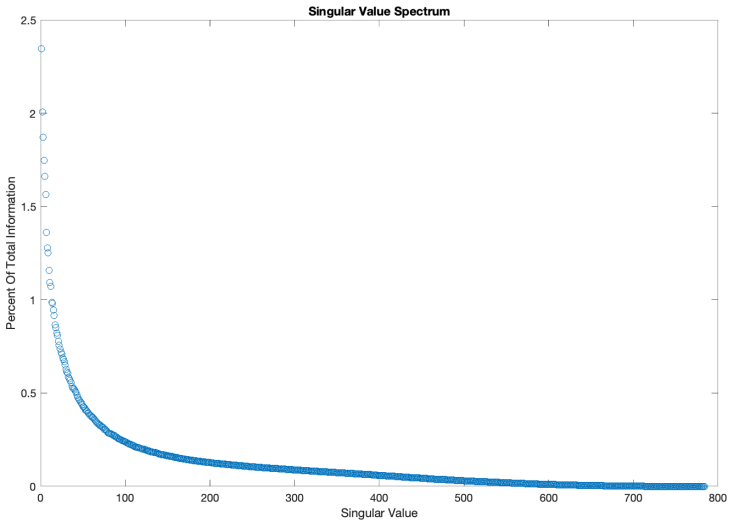

Figure 1: This is the singular value spectrum of the singular values from our Σ matrix from SVD, where the singular values are plotted as a percent of the total information.

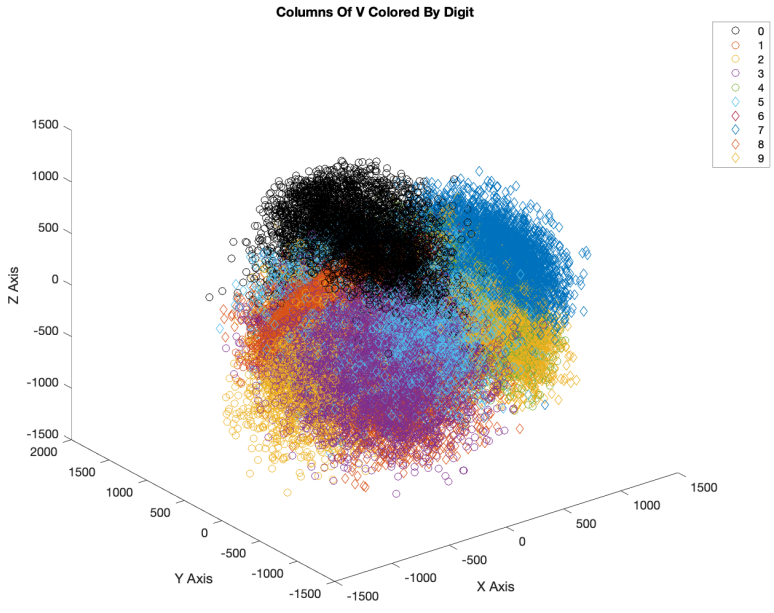

Figure 2: Here is the 2nd, 3rd, and 5th columns of V projected such that each digit is colored.

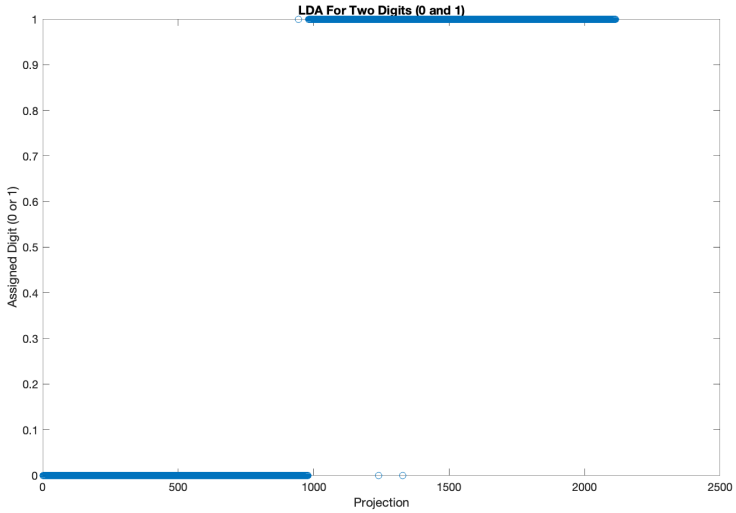

Figure 3: Here is the LDA for the digits 0 and 1.

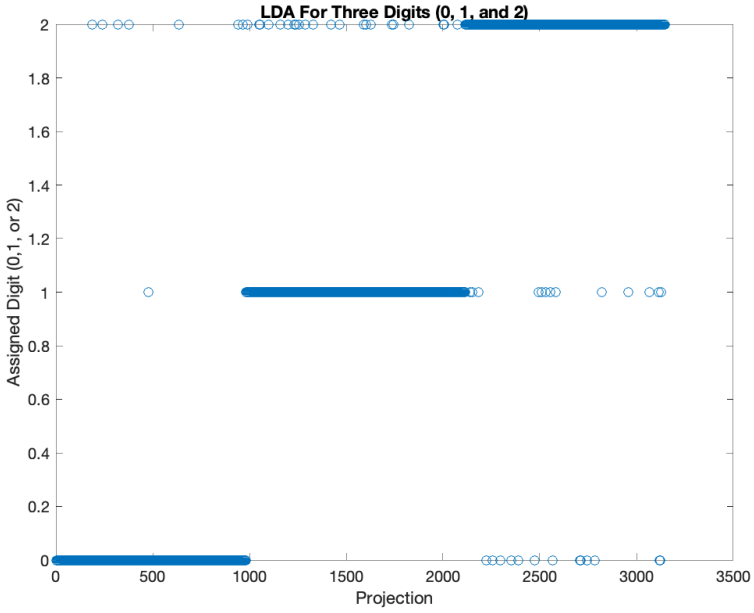

Figure 4: Here is the LDA for the digits 0, 1, and 2.

Appendix A MATLAB Functions

- `svd(A,'econ')` returns the SVD of matrix `A`, $[U, \Sigma, V^*]$, without the rows of zeros padding the Σ matrix.
- `classify(testData, trainingData, trainingLabels)` uses LDA on the the `trainingData` and `trainingLabels` and chooses the thresholds. Then returns a vector of the classes that the LDA and threshold assign to the `testData`.

Appendix B MATLAB Code

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

```
%% Prep
clear all; close all; clc
[images, labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte');
[testImages, testLabels] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte');

%% Reshape
X = zeros(784,60000);
for j = 1:60000
    img = transpose(images(:,:,j));
    X(:,j) = img(:);
end

ind0 = find(labels == 0);
ind1 = find(labels == 1);
ind2 = find(labels == 2);
ind3 = find(labels == 3);
ind4 = find(labels == 4);
ind5 = find(labels == 5);
ind6 = find(labels == 6);
ind7 = find(labels == 7);
ind8 = find(labels == 8);
ind9 = find(labels == 9);

[label, idx] = sort(labels);
trainImg = X(:,idx);
%% SV Spectrum
[m,n] = size(trainImg);
avg = mean(trainImg,2);
trainImg = trainImg - repmat(avg,1,n);
[U,S,V] = svd(trainImg,'econ');
sig = diag(S);
figure(1)
plot(100*sig/sum(sig),'O')
sum(sig(1:70))/sum(sig)
title("Singular Value Spectrum")
xlabel("Singular Value")
ylabel("Percent Of Total Information")
%% 4
figure(2)
view(3)
hold on
plot3(U(:,2)'*trainImg(:, 1:5923),U(:,3)'*trainImg(:,1:5923),U(:,5)'*trainImg(:,1:5923), 'blackO')
plot3(U(:,2)'*trainImg(:, 5924:12665),U(:,3)'*trainImg(:,5924:12665),U(:,5)'*trainImg(:,5924:12665), 'O')
plot3(U(:,2)'*trainImg(:, 12666:18623),U(:,3)'*trainImg(:,12666:18623),U(:,5)'*trainImg(:,12666:18623), 'O')
plot3(U(:,2)'*trainImg(:, 18624:24753),U(:,3)'*trainImg(:,18624:24753),U(:,5)'*trainImg(:,18624:24753), 'O')
plot3(U(:,2)'*trainImg(:, 24754:30596),U(:,3)'*trainImg(:,24754:30596),U(:,5)'*trainImg(:,24754:30596), 'O')
plot3(U(:,2)'*trainImg(:, 30597:36017),U(:,3)'*trainImg(:,30597:36017),U(:,5)'*trainImg(:,30597:36017), 'd')
plot3(U(:,2)'*trainImg(:, 36018:41935),U(:,3)'*trainImg(:,36018:41935),U(:,5)'*trainImg(:,36018:41935), 'd')
plot3(U(:,2)'*trainImg(:, 41936:48200),U(:,3)'*trainImg(:,41936:48200),U(:,5)'*trainImg(:,41936:48200), 'd')
plot3(U(:,2)'*trainImg(:, 48201:54051),U(:,3)'*trainImg(:,48201:54051),U(:,5)'*trainImg(:,48201:54051), 'd')
plot3(U(:,2)'*trainImg(:, 54052:60000),U(:,3)'*trainImg(:,54052:60000),U(:,5)'*trainImg(:,54052:60000), 'd')
title("Columns Of V Colored By Digit")
xlabel("X Axis")
ylabel("Y Axis")
zlabel("Z Axis")
legend('0','1','2','3','4','5','6','7','8','9')

%% Testing Prep
Y = zeros(784,10000);
for j = 1:10000
    img = transpose(testImages(:,:,j));
    Y(:,j) = img(:);
end
```

```
indTest0 = find(testLabels == 0);
indTest1 = find(testLabels == 1);
indTest2 = find(testLabels == 2);
indTest3 = find(testLabels == 3);
indTest4 = find(testLabels == 4);
indTest5 = find(testLabels == 5);
indTest6 = find(testLabels == 6);
indTest7 = find(testLabels == 7);
indTest8 = find(testLabels == 8);
indTest9 = find(testLabels == 9);

[label2, idy] = sort(testLabels);
testImg = Y(:,idy);
testDigits = U(:,1:70)'*testImg;
%% LDA 0 and 1
class = classify(testDigits(:,1:(size(indTest0)+size(indTest1)))', V(1:12665,1:70), label(1:12665)');
plot(class, 'o')
title("LDA For Two Digits (0 and 1)")
xlabel("Projection")
ylabel("Assigned Digit (0 or 1)")
count = 0;
for j = 1:2115
    if class(j) == label2(j)
        count = count + 1;
    end
end
percent = count/2115
%% LDA 0, 1, and 2
testDigits = U(:,1:70)'*testImg;
class = classify(testDigits(:,1:(size(indTest0)+size(indTest1)+size(indTest2)))', V(1:18623,1:70), label(1:18623)');
plot(class, 'o')
title("LDA For Three Digits (0, 1, and 2)")
xlabel("Projection")
ylabel("Assigned Digit (0,1, or 2)")
count = 0;
for j = 1:3147
    if class(j) == label2(j)
        count = count + 1;
    end
end
percent = count/3147
%% LDA 1 and 2
testDigits = U(:,1:70)'*testImg;
class = classify(testDigits(:,981:3147)', V(5924:18623,1:70), label(5924:18623)');
plot(class, 'o')
title("LDA For Three Digits (0, 1, and 2)")
xlabel("Projection")
ylabel("Assigned Digit (0,1, or 2)")
count = 0;
newlabel = label2(981:3147);
for j = 1:2167
    if class(j) == newlabel(j)
        count = count + 1;
    end
end
percent = count/2167
%% Tree for all digits
tree = fitctree(V(:,1:70),label,'MaxNumSplits',10);
testTree = predict(tree, testDigits');
count = 0;
for j = 1:10000
    if testTree(j) == label2(j)
        count = count + 1;
    end
end
percent = count/10000
```

```
%% Tree for 0 and 1
tree=fitctree(V(1:12665,1:70),label(1:12665),'MaxNumSplits',2);
testTree = predict(tree, testDigits');
count = 0;
for j = 1:2115
    if testTree(j) == label2(j)
        count = count + 1;
    end
end
percent = count/2115
%% Tree for 1 and 2
tree=fitctree(V(5924:18623,1:70),label(5924:18623),'MaxNumSplits',2);
testTree = predict(tree, testDigits(:,981:3147)');
count = 0;
newlabel = label2(981:3147);
for j = 1:2167
    if testTree(j) == newlabel(j)
        count = count + 1;
    end
end
percent = count/2167

%% SVM for 0 and 1
Mdl = fitcsvm(V(1:12665,1:70),label(1:12665));
test_labels = predict(Mdl,testDigits');
count = 0;
for j = 1:2115
    if test_labels(j) == label2(j)
        count = count + 1;
    end
end
percent = count/2115

%% SVM for 1 and 2
Mdl = fitcsvm(V(5924:18623,1:70),label(5924:18623));
test_labels = predict(Mdl,testDigits(:,981:3147)');
count = 0;
newlabel = label2(981:3147);
for j = 1:2167
    if test_labels(j) == newlabel(j)
        count = count + 1;
    end
end
percent = count/2167

%% SVM for all digits
Mdl = fitcecoc(V(:,1:70),label);
test_labels = predict(Mdl,testDigits');
count = 0;
for j = 1:10000
    if test_labels(j) == label2(j)
        count = count + 1;
    end
end
percent = count/10000
```