AMATH 482 Homework 4
Brad Hong
March 10, 2021

Abstract
This experiment will explore the usefulness of classification algorithms specifically LDA separation. We will train and test using both PCA and SVD to classify data given from MNIST.

Section I. Introduction and Overview

The focus of this report is to see how effective LDA is at classifying. We will be using the MNIST data set that has both training and testing data in the form of 60,000 and 10,000 images and labels respectively. First, we will perform an SVD analysis of the digit images to see how many principle components are needed to effectively classify the images. Then we will use pick specific digits to work around with to build a classifier around. Using different digits will cause the overlap of values to be different meaning classifying will be harder for digits that are similar. After playing around with different digits, we will specifically choose the most separated and least separated digits to see how LDA performs on both cases. Then these results will be compared to SVM and decision trees, methods used for classification until 2014.

Section II. Theoretical Background

The SVD is a method of deconstructing a matrix into different parts much like diagonalizing a matrix. It deconstructs the matrix A below in the following way

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

This equation, while seems unimpressive at first, is actually quite powerful. The V* matrix is used as a change of basis matrix to align the vectors of x to the correct axis. Then multiplying by sigma stretches/shrinks the vector to size and finally the U vector properly orients the vectors. These three matrices explain the process that is undergone when Ax is performed.

SVD is a useful tool in creating low dimensional models. These models, when created from geometric matrices of data, are called the PCA. The PCA consists of the principal components which are represented by this equation:

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^{*} \quad (2)$$

This equation represents the sum of all principal components of A. This means that each iteration of the summation represents a principal component of A. These values are all derived from the SVD of A where σ_j is the j-th diagonal of Sigma, u_j is the j-th column of U, and v_j^{*} is the j-th column of V*. Since the values in Sigma are ordered in descending order, the first principal component represents the component with the largest amount of information. The takeaway here is that, if is σ_j is near zero or small in comparison to other σ_i, then the j-th component can be ignored. This means we are able to create extremely accurate models of data using a handful of components.

The Principal Component Analysis (PCA) utilizes the SVD to separate data into multiple parts. The number of sigma values in the SVD represents the number of principal components. The first principal component will hold the most information as in it will hold the general form of the model. As we go from the first to the last principal component, less information will be held but more details will be added. This is why compiling all principal components of a model will recreate the model. The strength of the PCA lies in not needing all principal components to recreate an effective model. Thus, it is an effective tool in recreating models with only the use of certain principal components.

The Linear Discriminant Analysis is a tool used to effectively classify values. It does this by first using training data to find an optimal line to project values onto that will maximize the distance between classes. Through this line, threshold values can be found that create lines between different classes. This will be the basis for using the LDA for classification. After training, testing data is added to measure how well the LDA performed. If a certain value is not within its specific threshold values, it is marked as misclassified. The number of correctly classified divided by the number of test cases marks the effectiveness of the LDA.

Section III. Algorithm Implementation and Development

First, an SVD analysis was done to see how the images differed from one principal component to the next. To do this we used a wavelet to scale the data and performed a SVD on the scaled data to get the principal components of the MNIST data. Once we had the SVD matrices, we used the Sigma matrix to view the singular value spectrum. This would determine the number of modes needed to effectively reproduce a model using PCA. We then plotted a 3D scatterplot of the data on three selected principal components. In our case, components 2, 3, and 5.

After creating a plot of the data, two digits were selected to perform an LDA on. After some playing around with different digits, two sets of digits were found that were the most separated and least separated. LDA was used on these values to see how well LDA would perform on very separated classes and very intertwined classes. Effectiveness of the LDA was based on the percent classification of testing data, also from MNIST. Finally, we were to perform SVM and decision trees on the same data to compare results with the LDA.

Section IV. Computational Results

The results of the singular value spectrum are shown below.

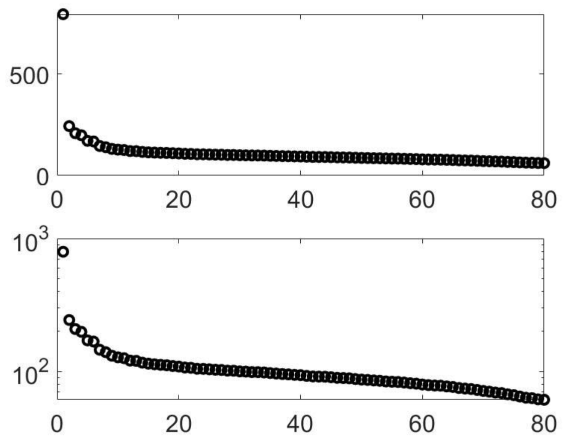

Figure 1: Singular Value Spectrum of Scaled Data

As you can see, much of the information is explained within the first principal components but not all. In fact, a lot of the principal components are not completely zero, meaning there is some information being held within each principal component. After looking at the singular value spectrum, it was decided that the first 10 components would hold the most information about the model.

The 3d scatterplot composed of the second, third, and fifth principal components is shown below:

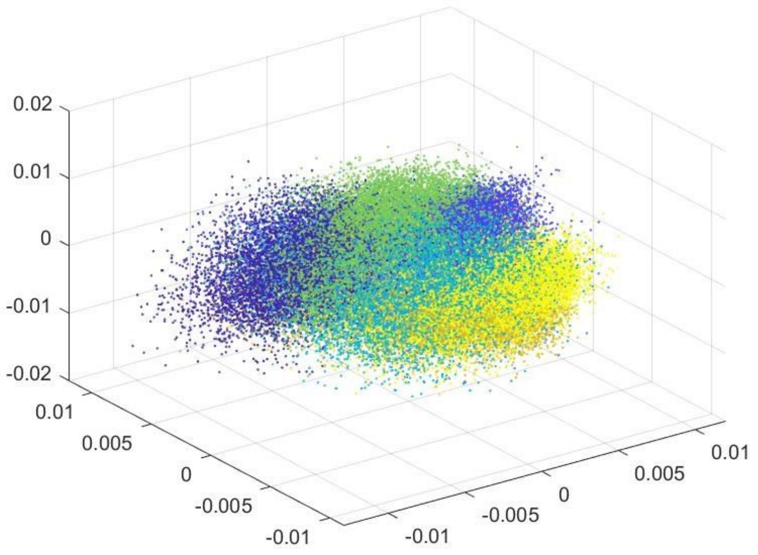

Figure 2: 3D Scatterplot Created from V-modes 2, 3, and 5

Based on the image above, it is hard to tell that there are 10 different digits within the same plot. However, it is clear to see that there are clumps of specific digits. This guarantees that there will be digits that are similar and digits that are completely different. This is why we will select specific digits to see how well the LDA can classify between them.

Although I was unable to find a way to classify 3 digits, I went on to using two digit classification. For the farthest apart digits, I found digits 0 and 7 to be farthest away when looking at the scatterplot:

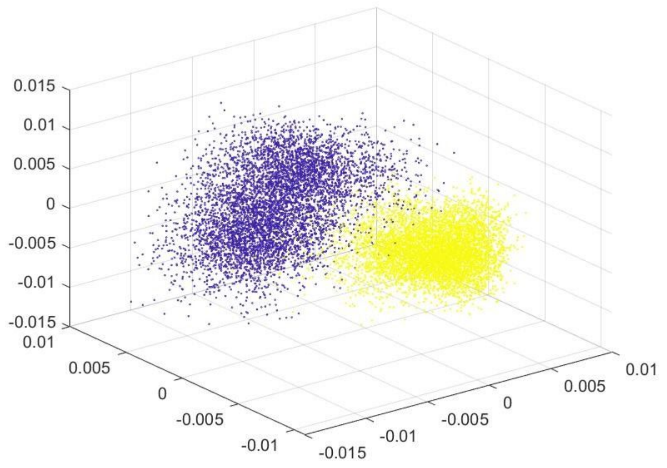

Figure 3: Scatterplot of Digits 0 and 7

After performing a LDA on the 0 and 7 data, we got the following graph:

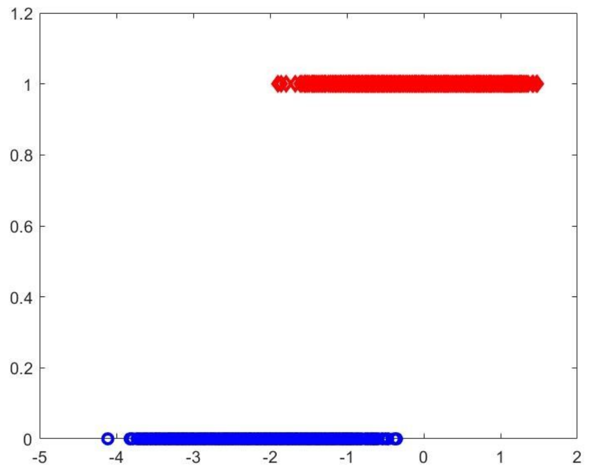

Figure 4: Projection line for Digits 0 and 7 with classified data

The projection line was surprisingly overlapping even though the original plot didn't show them having much.

After finding the two most separated digits, I found digits 2 and 5 to be very overlapping:

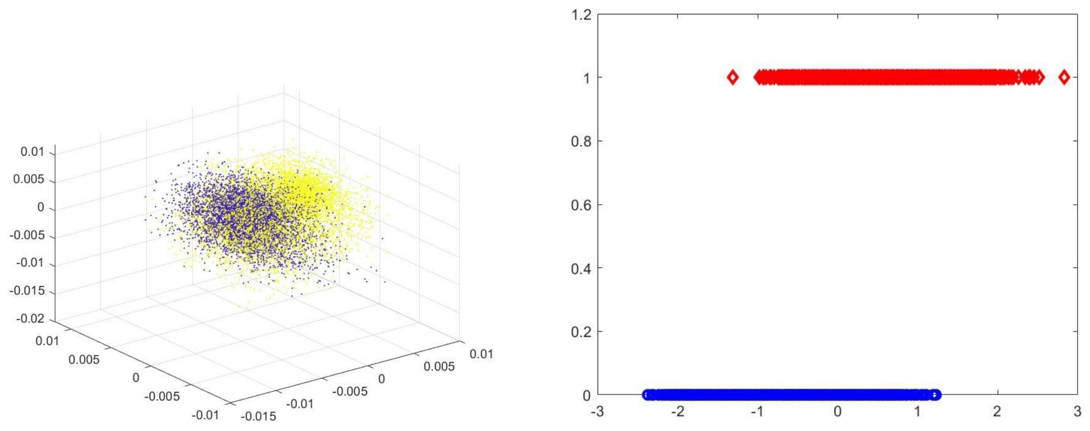

Figure 5: The Scatterplot (left) and Projection Line (right) of the Digits 2 and 5

As you can see above, the scatterplot depicts the two digits being very similar and the projection line reflecting that as there is much more overlap between the two digits than in the farthest case.

These results are expected as it would be much easier to classify two completely different objects than classifying two similar ones. When testing the effectiveness of these values, the accuracy with the most separated digits was around 83% and the least separated digits was around 68%. These values come from testing the 10,000 testing values provided by MNIST.

After working with the SVM and decision tree, I was unable to get them to work. However, I imagine that due to them both being out of date, the LDA is a much better method for classifying values.

Section V. Summary and Conclusions

To conclude, the LDA is a surprisingly great tool for classifying values. When combined with PCA/SVD to separate data into principal components it becomes an even better tool. Although I was unable to compare the results I got from LDA with past classifying algorithms, I firmly believe that since this field is generally new and being developed upon, the algorithms from seven years ago doesn't compare with the results I got from this experiment. After working around with the LDA, I got a good understanding of its strengths and weaknesses. It is great at finding the best projection line, thus it is able to find the best possible classification area. Even when given two similar classes, it does its best as seen by the two digits with smallest separation.

Appendix A. MATLAB functions used and brief implementation explanation

svd()
Used to get the Single Value Decomposition of a matrix. A second parameter ‘econ’ is used to use the simplified version of the SVD to save time.
This function return values U, S, and V each of which correspond to U, Sigma, and V from the SVD algorithm. Note that V should be transposed to get V*.

scatter3()
Used to create 3D scatterplots of data. User must pass in the X, Y and Z axis along with any other classifications such as color or dot size.

a4_wavelet()
A similar wavelet used in lecture to scale the values of the data.

mnist_parse()
A method to parse the MNIST data set.

Appendix B. MATLAB codes

```
[images, testlabels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte');
sz = size(images);
images = reshape(images, [sz(1)*sz(2), sz(3)]);
%%
im_wave = a4_wavelet(images);
[U,S,V] = svd(im_wave, 'econ');

%%
figure(2)
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',2)
set(gca,'Fontsize',16,'Xlim',[0 80])
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',2)
set(gca,'Fontsize',16,'Xlim',[0 80]) % 10 modes will have a lot of info

%%

figure(3)
scatter3(V(:,2),V(:,3),V(:,5),1,testlabels);
```

```
%%

figure(4)
val1 = 2;
val2 = 5;
X = V(testlabels == val1 | testlabels == val2, :);
lim_labels = testlabels(testlabels == val1 | testlabels == val2);
scatter3(X(:,2),X(:,3),X(:,5),1,lim_labels);

%%
im1 = images(:, testlabels == val1);
im2 = images(:, testlabels == val2);
im1_wave = a4_wavelet(im1);
im2_wave = a4_wavelet(im2);
[U1,S1,V1] = svd([im1_wave im2_wave], 'econ');
%%
feature = 10;
n1 = size(im1_wave,2);
n2 = size(im2_wave,2);
digits = S1*V1'; % projection onto principal components: X = USV' --> U'X = SV'
dig1 = digits(1:feature,1:n1);
dig2 = digits(1:feature,n1+1:n1+n2);
m1 = mean(dig1,2);
m2 = mean(dig2,2);
Sw = 0;
for k = 1:n1
   Sw = Sw + (dig1(:,k) - m1)*(dig1(:,k) - m1)';
end
for k = 1:n2
   Sw =  Sw + (dig2(:,k) - m2)*(dig2(:,k) - m2)';
end
Sb = (m1-m2)*(m1-m2)';

[V2, D] = eig(Sb,Sw);
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);
v1 = w'*dig1;
v2 = w'*dig2;

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

```
    v2 = -v2;
end
%%

figure(5)
plot(v1,zeros(n1),'ob','Linewidth',2)
hold on
plot(v2,ones(n2),'dr','Linewidth',2)
ylim([0 1.2])

%%

sort1 = sort(v1);
sort2 = sort(v2);

t1 = length(sort1);
t2 = 1;
while sort1(t1) > sort2(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold = (sort1(t1) + sort2(t2))/2;

%%

[testimages, testlabels] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-ubyte');
sz = size(testimages);
testimages = reshape(testimages, [sz(1)*sz(2), sz(3)]);

%%

TestNum = size(testimages,2);
Test_wave = a4_wavelet(testimages); % wavelet transform
TestMat = U1'*Test_wave; % PCA projection
pval = w'*TestMat;
```