AMATH 482 Homework 4: LDA Analysis on Digit Classification

Leqi Wang

March 10, 2020

Abstract

In this assignment, we use SVD and LDA analysis to train model to classify digits to different labels in data set MNIST, and finally test our model on test data set and analyze on the effectiveness of our model.

1 Introduction and Overview

We used SVD to understand the statistical difference among different groups. After we learned the difference by using SVD, we set a threshold by using linear discrimination analysis to differentiate what digits belong to what labels. To provide comparison of efficiency of algorithm, we also introduced the MatLab built-in methods SVM to show the success rate of our training model.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

Compared to Spectral Decomposition, SVD can be applied to non-square matrix m x n. Matrix A can always be written as:

$$AV = U\Sigma \tag{1}$$

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

Where V is a unitary eigen vector matrix of A, U is m x m complex unitary matrix, Σ is m x n rectangular diagonal matrix. To write SVD in a reduced form, where Σ gets rid of zero rows and columns, A can be written as:

$$A = \hat{U}\hat{\Sigma}V^* \tag{3}$$

The rank of $\hat{\Sigma}$ should not exceed rank of A, each sigma, it contains the singular values of A and represents the energy of each singular component. By convention, we arrange these singular values from largest to smallest, along with its corresponding vector basis in U and V. Thus the SVD is just least-square fitting algorithm that projects the original matrix onto lower dimensional basis. The action of A on a vector $\vec{x}$ is now $\hat{U}\hat{\Sigma}V^*\vec{x}$, which represents the action of rotation - stretch - rotation and provides a result of low-rank approximation.

2.2 Linear Discrimination Analysis

LDA is used to determine the labels of certain digits. The idea of LDA is to project data on new basis so that the distance between the inter-class data is maximized, and the inter-class data distance is minimized. For a 2-class LDA, we introduced $w = argmax(\frac{w^T S_B w}{w^T S_W w})$, where the scatter matrix for between -class S_b and with-in class S_w are:

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

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

And high dimensional data set (j ¿= 3), we can write:

$$S_B = \sum_{j=1}^{N} (\mu_j - \mu)(\mu_j - \mu)^T \tag{6}$$

where μ is the overall mean, and μ_j is the row mean, while

$$S_W = \sum_{j=1}^{N} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{7}$$

which essentially could be generalized as the eigen problem solved by:

$$S_b w = \lambda S_w w \tag{8}$$

The maximum eigenvalue λ and associated eigenvector will be the quantity of interet and the projection basis.

3 Algorithm Implementation and Development

3.1 Part I: SVD Analysis

First reshape the data of images into two dimensional, so that each column of the matrix represent a different image. There are 60000 images in the training set, therefore the image matrix is 784(28x28) x 60000. Then perform reduced SVD on the matrix, the projection onto three selected V modes(which chose to be 1,3,5) should be completed by S * V'.

3.2 PartII: Linear Discrimination Analysis

By picking two random digits, I choose two with label 0 and 9 in the training set. After calculating the within class and inter class variance matrix, I found the best projection line represented by vector w and I calculate the threshold by having (approximately) same number of wronged label 0 and 9. Then I pick out the test data with same label 0 and 9, and examine how many of them are in the right category. For 3-digit classification, I add a label 3, and then calculate S_b and S_w separately to get new line w, and calculate the threshold. I examine how many of them are in the right category just like 2-digit classification.

To find out which data set are the most hard to classify, I in turn examine all the data set with regard to each other. I calculate the success rate, and with the highest success rate meaning these two data sets are easiest to separate because they are less meshed up, vice versa.

Given the SVM and classification tree methods, they directly train data (the projection S * V' instead of raw image) with given labels.

4 Computational Results

4.1 PartI: SVD Analysis

According to Figure 1, after the first singular value, the rest of the energy drops drastically. After about first 30 singular values, the rest does not play a large part anymore. Independently speaking, the singular value decomposition result shows that U are the principal components, the diagonal of S represents the energy of each singular values, and V represents how well the singular values are projected on the new basis. Figure 2 shows the projection S * V' on the new basis (the 1st, 3rd, and 5th columns of V).

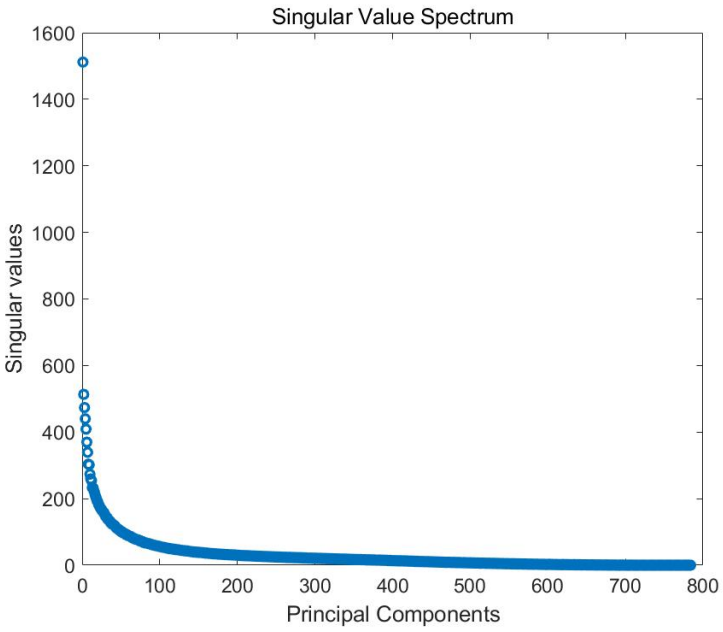

Figure 1: Singular Value Spectrum

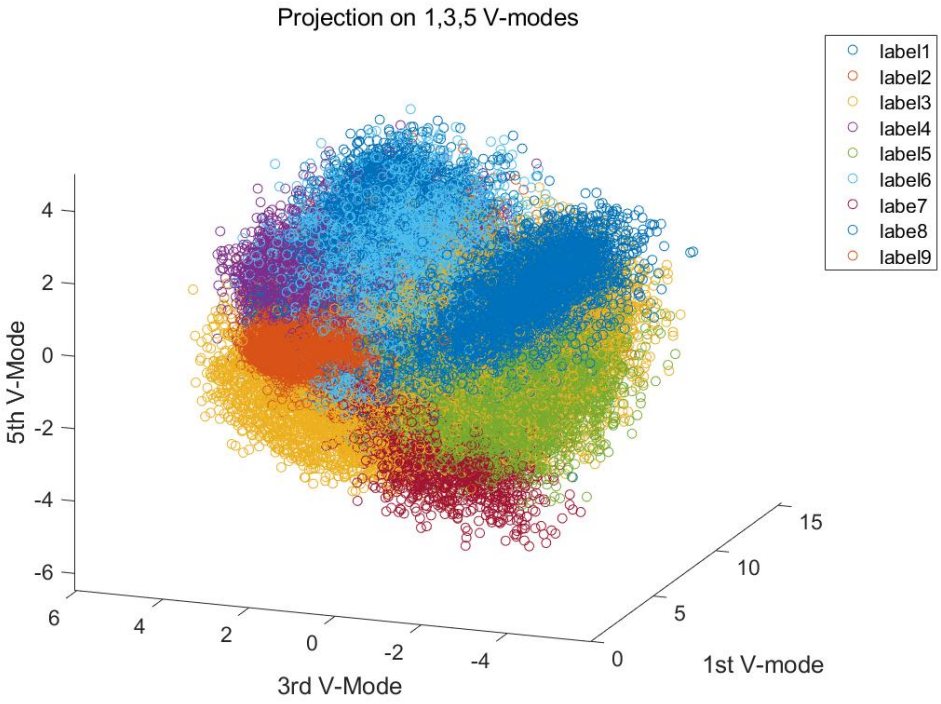

Figure 2: Projection on V-modes

Projection Distribution

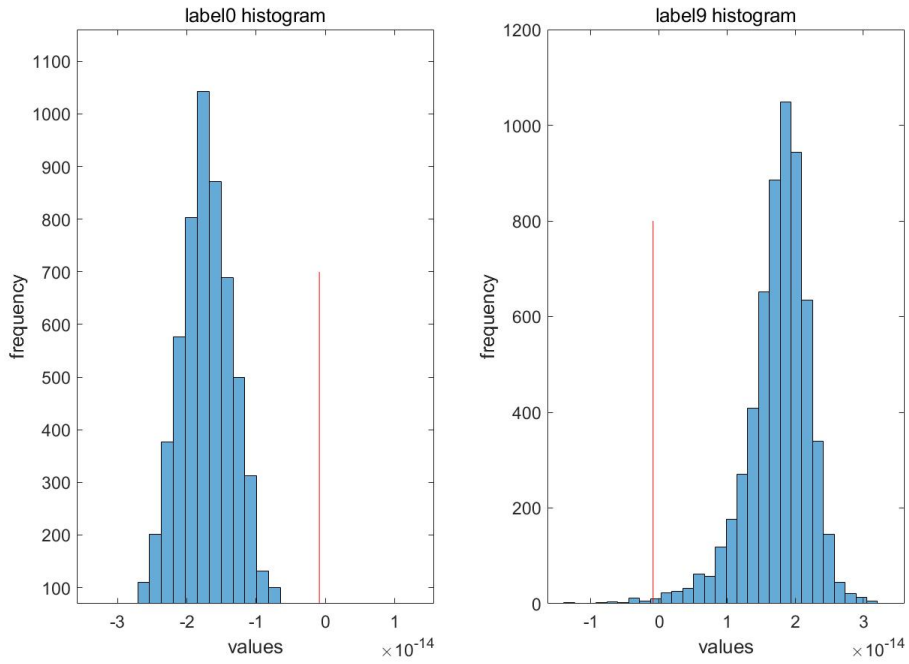

Figure 3: 2-digits classification

4.2 Part II LDA Analysis

Figure 3 demonstrates the histogram of accuracy of the classification of test data with label 0 and 9(2-digits). And for 3-digit classification, I build the LDA model with label 0, 1, and 2, get two threshold of 2.0078 and -1.3671, with success rate around 0.34.

Data sets most difficult to separate are those with data meshed together when projecting. By calculation, group 4 and 5 are the hardest to separate, with only success rate 0.0337; group 0 and 9 are the easiest to separate, with success rate 0.9925.

The classification tree provides class error = 0.1755, meaning the success rate of classification = 0.8245. (I eliminate the argument maxNumSplit to increase accuracy, though the time will increase tremendously.). Compared to LDA, which extremely reduces all the data to one dimension and accuracy drops when the number of categories of data is higher, SVM assumes every group is totally separable and on data that are hard to classify, and therefore optimize more on outliers.

5 Summary and Conclusions

Combined with SVD, Linear Discrimination Analysis is useful for classifying data with evident features. It is used as a data-driven methods to distinguish statistically the difference between each vector of data, and project them onto the most suitable data set so that the test data can be easily categorized. LDA focuses on all data that are in the training set. The more clusters there are for data, then it is harder to project them on a single basis since they'll be too many overlaps.

Appendix A MATLAB Functions

- Loading section: `mnist parse`: load training data and test data with their labels to different variables.

- `im2double(reshape())`: reshape the loaded data into the form where column represents an image and convert them to double;
- `svd(data, 'econ')`: conduct reduced SVD.
- Plotting section: `plot3(projection(1, index), projection(3, index), projection(5, index), "o")`: plot the projection onto three selected V-modes, which are 1st, 3rd, and 5th.
- `find()` used to find index of given label.
- `dc trainer()`: a self-implemented function, given 2 data set, used to calculate the SVD of the data and threshold.
- `size(data, 2)` calculate the number of columns of data.
- `fitctree()`: Fit binary decision tree for multiclass classification
- `kfoldLoss()`Classification loss for cross-validated classification model, used to calculate accuracy.
- `fitcsvm()`: Train support vector machine (SVM) classifier for one-class and binary classification.
- `predict()`: used to predict the classification of given test data and trained model(Mdl).

Appendix B MATLAB Code

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

%% SVD Analysis
% for i = 1:20 % show images
%     imshow(images(:, :, i));
% end
[x, y, z] = size(images);
I = im2double(reshape(images, [x * y, z]));
[U,S,V] = svd(I, 'econ');

%% Plot singular values
figure(1)
plot(diag(S),'o','Linewidth',2)
title('Singular Value Spectrum')
xlabel('Principal Components')
ylabel('Singular values')
set(gca,'Fontsize', 14)
% set(gca,'Fontsize',16,'Xlim',[0 80]) % plot first 80 singular values
%% Projection onto V
figure(2)
projection = S * V';
for i = 0:9
    index = find(labels == i); % find which index of data contains label i;
    plot3(projection(1, index), projection(3, index), projection(5, index), "o
        ");
    hold on;
end
xlabel('1st V-mode'), ylabel('3rd V-Mode'), zlabel('5th V-Mode')
```

```
title('Projection on 1,3,5 V-modes')
legend('label1', 'label2', 'label3', 'label4','label5', 'label6',...
    'labe7', 'labe8', 'label9')
set(gca,'Fontsize', 14)


%% load test data
[test_images, test_labels] = mnist_parse('t10k-images.idx3-ubyte', 't10k-
    labels.idx1-ubyte');
[x, y, z] = size(test_images);
test_I = im2double(reshape(test_images, [x * y, z]));

%% 2 digits LDA
feature = 20;
label1 = 0; label2 = 9;
index1 = find(labels == label1); index2 = find(labels == label2);
data1 = I(:, index1); data2 = I(:, index2);
[U2,S2,V2,threshold,w,sort1,sort2] = dc_trainer(data1, data2,feature);
test_index1 = find(test_labels == 0); test_data1 = test_I(:, test_index1);
test_index2 = find(test_labels == 9); test_data2 = test_I(:, test_index2);

TestSet = [test_data1, test_data2];
TestNum = size(TestSet,2);
TestLabel = zeros(1,TestNum);
TestLabel(1, size(test_data1, 2) + 1:end) = 1;
TestMat = U2'*TestSet; % PCA projection
pval = w'*TestMat;

ResVec = (pval > threshold); err = abs(ResVec - TestLabel);
errNum = sum(err);
success_rate = 1 - errNum / TestNum;


%% random 2 digit LDA
success_rate_lowest = 1;
group1 = 0;
group2 = 1;
success_rate_highest = 0;
group1_high = 0;
group2_high = 1;
for i = 0:9
    for j = i + 1 : 9
        label1 = i; label2 = j;
        index1 = find(labels == label1); index2 = find(labels == label2);
        data1 = I(:, index1); data2 = I(:, index2);
        [U2,S2,V2,threshod,w,sortdog,sortcat] = dc_trainer(data1, data2,
            feature);
        test_index1 = find(test_labels == 0); test_data1 = test_I(:,
            test_index1);
        test_index2 = find(test_labels == 9); test_data2 = test_I(:,
            test_index2);

        TestSet = [test_data1, test_data2];
        TestNum = size(TestSet,2);
```

```
        TestLabel = zeros(1,TestNum);
        TestLabel(1, size(test_data1, 2) + 1:end) = 1;
        TestMat = U2'*TestSet; % PCA projection
        pval = w'*TestMat;

        ResVec = (pval > threshold); err = abs(ResVec - TestLabel);
        errNum = sum(err);
        success_rate = 1 - errNum / TestNum;
    end
    if success_rate < success_rate_lowest
        group1 = i;
        group2 = j;
        success_rate_lowest = success_rate;
    end
    if success_rate > success_rate_highest
        group1_high = i;
        group2_high = j;
        success_rate_highest = success_rate;
    end
end

%% Three digits LDA
label1 = 0; label2 = 1; label3 = 2;
index1 = find(labels == label1); index2 = find(labels == label2); index3 =
    find(labels == label3);
data1 = I(:, index1); data2 = I(:, index2); data3 = I(:, index3);
[U5,S5,V5,threshold1,threshold2,w,max_ind,min_ind] = dg3_trainer(data1,data2,
    data3,feature);

test_index1 = find(test_labels == 0); test_data1 = test_I(:, test_index1);
test_index2 = find(test_labels == 1); test_data2 = test_I(:, test_index2);
test_index3 = find(test_labels == 2); test_data3 = test_I(:, test_index3);

TestSet = [test_data1, test_data2, test_data3];
TestNum = size(TestSet,2);
TestLabel = zeros(1,TestNum);
TestLabel(1, size(test_data1, 2) + 1:end) = 1;
TestMat = U2'*TestSet; % PCA projection
pval = w'*TestMat;

ResVec = (pval > threshold); err = abs(ResVec - TestLabel);
errNum = sum(err);
success_rate = 1 - errNum / TestNum;


%% SVM and Decision tree
% 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 classifier with training data, labels and test set
Mdl = fitcsvm(xtrain,label);
```

```
test_labels = predict(Mdl,test);

%%
tree = fitctree(projection', labels, 'CrossVal', 'on');
classError = kfoldLoss(tree); % tree error  0.1755
tree_sucrate = 1 - classError; % 0.8245
label1 = 0; label2 = 9; % easy
index1 = find(labels == label1); index2 = find(labels == label2);
data1 = I(:, index1); data2 = I(:, index2);
label1 = 1; label2 = 9; % hard
index1 = find(labels == label1); index2 = find(labels == label2);
data1 = I(:, index1); data2 = I(:, index2);
% SVM
Mdl =  fitcecoc(projection(1:50, :)', labels);
predict_labels_svm = predict(Mdl, TestMat');
test_labels = predict(Mdl,test1);
```