AMATH 482 Homework 4

Erik Huang

March 10, 2021

Abstract

Experiment the building process of Linear Classifier(LDA), Support Vector Machines(SVM) and decision tree. Test the performance of different classification methods on the MNIST database of hand-written digits.

1 Introduction and Overview

This assignment is about the usage of inear Classifier(LDA) and Support Vector Machines(SVM). We are provided with a data set containing 60000 images of hand-written digits. To train our classifiers, we need to first project the data in PCA space and then perform LDA and SVM analysis.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

The concept of Singular Value Decomposition is introduced in the last assignment. The SVD is a factorization of a matrix in linear algebra, and one of the most powerful and versatile tool in data analysis. The singular value decomposition of a $m \times n$ matrix A is:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

$\mathbf{U}$ is a $m \times n$ unitary matrix with orthonormal columns. $\mathbf{\Sigma}$ is a $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal. $\mathbf{V}$ is a $m \times n$ complex unitary matrix. $\mathbf{U}$ and $\mathbf{V}$ contain information on rotation and $\mathbf{\Sigma}$ describes the stretch in each direction. Applying SVD on data set allows us to get the principal components from the data.

2.2 Linear Discriminant Analysis(LDA)

With the principal components acquired from the data, we need a method to demonstrate the difference between different classes(or labels). Linear Discriminant Analysis projects data sets onto new bases, which would maximize the distance between the between-class data while minimizing the within-class data.
We will define a between-class scatter matrix for the variance between the groups

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

And a within-class scatter matrix for the variance within the groups

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

μ_j are column vectors indicating the means for each of the groups for each feature. With these 2 matrices obtained, the goal is to find vector $\mathbf{w}$, defined as

$$\mathbf{w} = \operatorname{argmax} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

This vector will translate to the goal of LDA that we stated at first. To solve for this vector in MATLAB, we need to transform this question into a generalized eigenvalue problem:

$$\mathbf{S}_B\mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}$$

This equation is solvable in MATLAB using eigen-decomposition methods.

2.3 Support Vector Machines(SVM)

Support Vector Machines are supervised learning models with learning algorithms for the purpose of classification on data. It is a very commonly used classification tool in statistics. The way it classifies the data set is via constructing a hyper plane or a set of hyper planes that best separate data points. In this study, we have training data entries as such:

$$\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_n, y_n\right)$$

where the $\mathbf{x}_n$ are data points and the y_n are either 0 if data is correctly labeled or 1 if data is incorrectly labeled. An example of ideal hyper plane definition for two groups would be

$$\mathbf{w}^T\mathbf{x} - b = 0$$

where all data points $\mathbf{w}^T\mathbf{x} - b \leq 0$ are in one group and all data points $\mathbf{w}^T\mathbf{x} - b > 0$ are in the other group. To obtain the goal, we transform it into an optimization problem:

$$\text{"Minimize } \|\mathbf{w}\| \text{ subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1 \text{ for } i = 1, \ldots, n \text{ ."}$$

2.4 Decision Tree

Decision tree is another technique we will explore in this study. This method constructs a tree-like model with each branch represents the outcome of the test, and each leaf node(node with no branch) represents a class label. Before one data point reaches the leaf node, it would first go through one or more decision nodes. The classification process is complete as the data point reaches the leaf node.

3 Algorithm Implementation and Development

The entire task can be split into:

1. Data processing with SVD
2. Different cases of LDA analysis on digits
3. SVM and decision tree analysis and comparison

3.1 Data processing with SVD

1. Load the train, test data and labels using the provided `mnist_parse` function.
2. Reshape each image input into a column vector (Subtract row-mean from the original train data)
3. Perform Singular Value Decomposition

3.2 Different cases of LDA analysis on digits

1. First construct S_B and S_w matrices
2. Iterate through all the class labels(or a subset) and populate S_B and S_w using between-class variance and in-class variance.
3. Obtain the classifier by performing decomposition on S_B/S_w

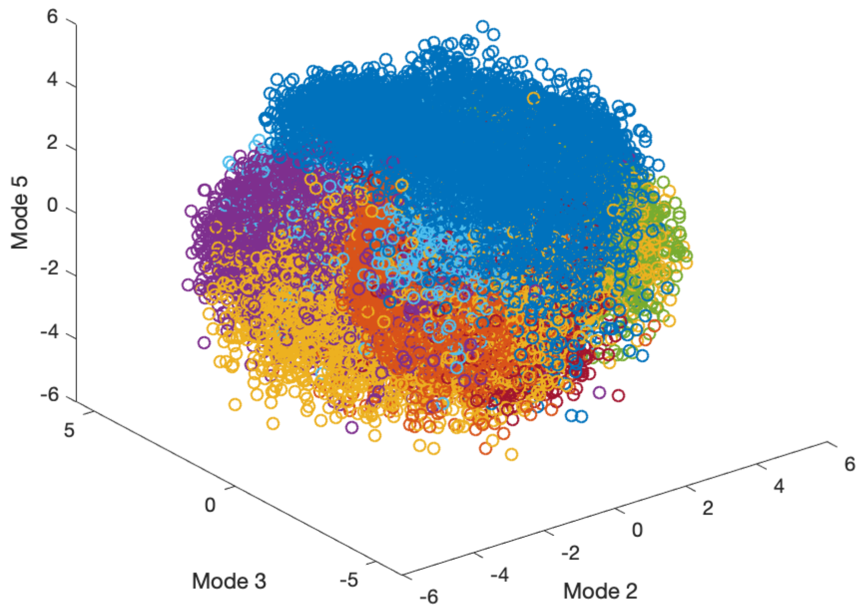

Figure 1: This is the plot of all digits projected to modes 2, 3 and 5.

3.3 SVM and decision tree analysis and comparison

1. Train SVM model using the builtin MATLAB library `fitcecoc`
2. Train decision tree model using the builtin MATLAB library `fitctree`
3. Compare the accuracy of each model by checking the prediction accuracy.

4 Computational Results

We first plotted the projection of all digits with 3 selected modes. On the figure 1, each color represents one digit label. These labels are clustered but close to each other.

Figure 2 and 3 demonstrates 2 cases of easy and hard seperation between digits. We can see that data points of digit 0 and 9 can be well separated by modes we choose. However, data points of digit 4 and 9 are mixed, which seems harder to separate. Although this plot is not a complete representation of the situation since 3 dimensional plot can only showcase the effect of 3 mode while there are a total of 784 modes. The accuracy calculation would provide more evidence:

The accuracy of the LDA filter on digit 0 and 9 is 0.9864.
The accuracy of the LDA filter on digit 4 and 9 is 0.9146.
Furthermore, we can plot the LDA and look at the distance of the projection. Figure 4 and 5 demonstrates the case of a 2 digit classification using LDA filter and a case of 3 digit. We can confirm that the seperation distance between digit 0 and 9 is relative large in comparison to digit 4.
Lastly, we will take a look at the accuracy level of SVM and Decision Tree models. Since our data set contains a total of 10 classes, we need to fit multiclass models for support vector machines. The resulting accuracy is around 86.38% on digit 4 and 9. Due the limitation in hardware, I couldn't finish running `fitcecoc` to train all 10 digits.
The accuracy for decision tree model is 88.52% at maximum splits of 10.

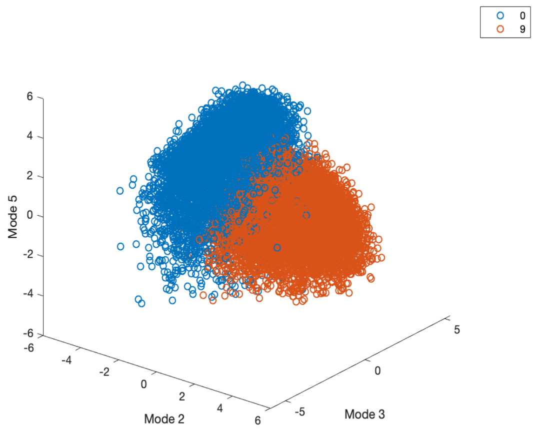

Figure 2: Easy separation: digit 0 and 9

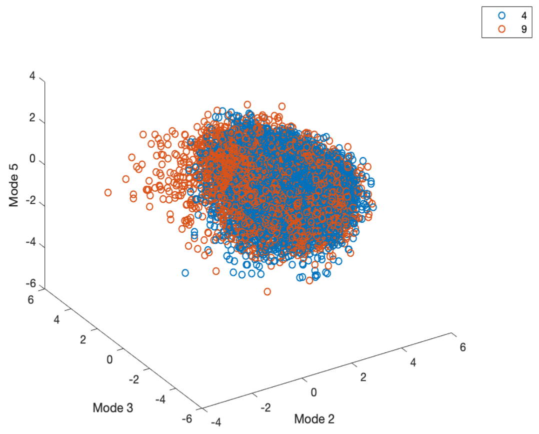

Figure 3: Hard separation: digit 4 and 9

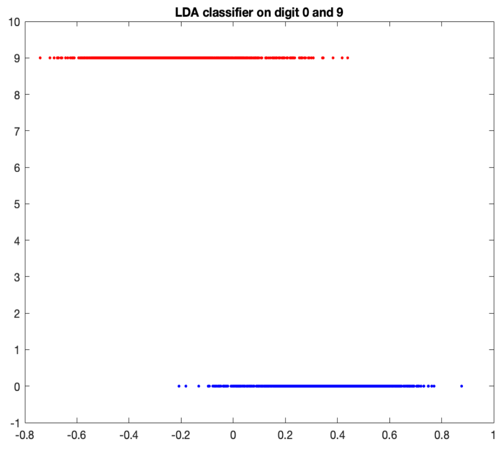

Figure 4: LDA Classifier on digit 0 and 9

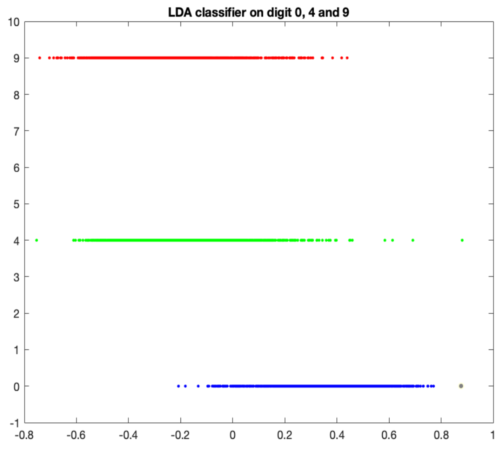

Figure 5: LDA Classifier on digit 0, 4 and 9

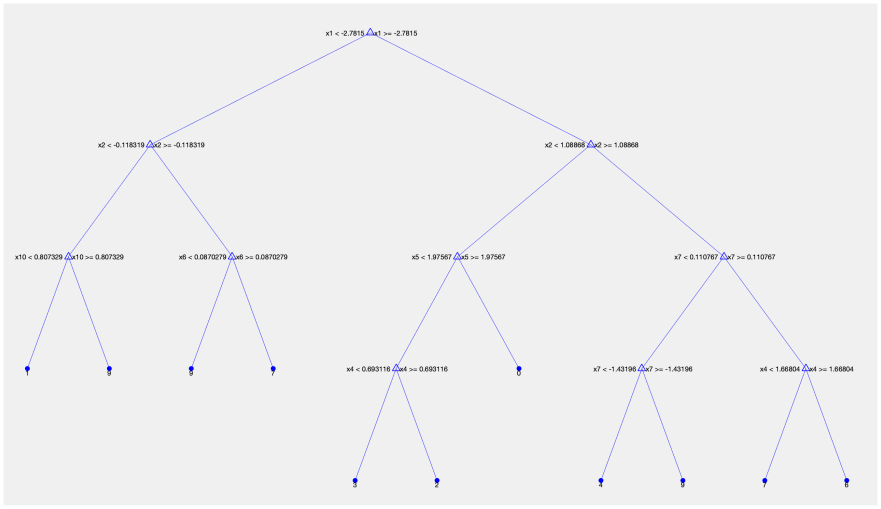

Figure 6: Visualized decision tree

5 Summary and Conclusions

From the study, we have the following observations. SVD can yield hundreds of modes from the data set. Simply using the first few ones cannot accommodate for all classes. LDA filter works great when projecting two classes but starts to fail as the number of classes increases. SVM and decision tree are both good classification tools in general. However, LDA filter seems to have the best performance overall.

Appendix A MATLAB Functions

- `im2double` Convert image to double precision
- `find` Find indices and values of nonzero elements
- `repmat` Repeat copies of array elements
- `svd` Singular Value Decomposition
- `fitcecoc` Fit multiclass models for support vector machines or other classifiers
- `fitcsvm` Train support vector machine (SVM) classifier for one-class and binary classification
- `fitctree` Fit binary decision tree for multiclass classification

Appendix B MATLAB Code

```
clear all; close all; clc;

[train_image, train_label] = mnist_parse('train-images-idx3-ubyte', 'train-labels-idx1-ubyte
    ');
[test_image,  test_label] = mnist_parse('t10k-images-idx3-ubyte', 't10k-labels-idx1-ubyte');
train_image_reshaped = reshape(train_image, size(train_image,1)*size(train_image,2), []).';
test_image_reshaped = reshape(test_image, size(test_image,1)*size(test_image,2), []).';
train_image = im2double(train_image_reshaped)';
test_image = im2double(test_image_reshaped)';
train_label = im2double(train_label);
test_label = im2double(test_label);

row_mean = mean(train_image,2);
train_image = double(train_image)-repmat(row_mean, 1, length(train_image));

% Singular Value Decomposition
[U, S, V ] = svd(train_image, 'econ');


energy = 0;
r = 0;
while energy < 0.95
    r = r + 1;
    energy = energy + S(r,r)/sum(diag(S));
end
train_image = (U(:, 1:r))'*train_image;
test_image = (U(:, 1:r))'*test_image;
%% PCA PLOT on digit 0, 9
for i = [0,9]
    disp(i);
    modes = train_image(:, find(train_label == i));
    plot3(modes(2,:), modes(3,:), modes(5,:),'o');
    hold on
end
xlabel('Mode 2')
ylabel('Mode 3')
zlabel('Mode 5')
legend('0', '9')
```

```matlab

%% LDA classifier

X = train_image;
T = test_image;

Sb = zeros(row_size);
Sw = zeros(row_size);
train_size = size(train_image, 2);
test_size = size(test_image, 2);
row_size = size(train_image,1);
Mu = mean(train_image, 2);

for i = [0,7]
    mask = (train_label ==  i);
    x = X(:, mask);
    ni = size(x, 2);
    pi = ni / train_size;
    mu_i = mean(x, 2);
    Si = (x - repmat(mu_i, [1,ni]))*(x - repmat(mu_i, [1,ni]))';
    Sw = Sw + Si ;
    Sb = Sb + (mu_i - Mu) * (mu_i - Mu)';
end
M = pinv(Sw) * Sb;
[U, D, V] = svd(M);

%% PLOT CLASSIFIER

G2 = U(:,1:size(M,1));
Y2 = G2' * X;
mask = (train_label == 0);
a = Y2(1,mask);
b = Y2(2,mask);
d = [a'; b'];
plot(d,0*ones(size(d)),'.b','Linewidth',2)
hold on
mask = (train_label == 4);
a = Y2(1,mask);
b = Y2(2,mask);
d = [a'; b'];
plot(d,4*ones(size(d)),'.g','Linewidth',2)
hold on
mask = (train_label == 9);
a = Y2(1,mask);
b = Y2(2,mask);
d = [a'; b'];
plot(d,9*ones(size(d)),'.r','Linewidth',2)
ylim([-1 10])
title(['LDA classifier on digit 0, 4 and 9']);

%% Accuracy Test for LDA
digit1 = 4;
digit2 = 9;

Y = G2' * X;
Y_t = G2'* T;

train_n = Y(:,find(train_label == digit1|train_label ==digit2));
test_n = Y_t(:,find(test_label == digit1|test_label ==digit2));
accuracy = accurCal(test_n, train_n,...
    test_label(find(test_label == digit1 |test_label ==digit2)), ...
    train_label(find(train_label == digit1 |train_label ==digit2)));
disp(accuracy);


%% SVM on two digit

train_trimed = train_image(:,find(train_label == digit1|train_label == digit2));
```

```
label_trimed = train_label(find(train_label == digit1|train_label == digit2));
test_trimed = test_image(:,find(test_label == digit1|test_label == digit2));
test_label_trimed = test_label(find(test_label == digit1|test_label == digit2));
svm_best = fitcsvm(train_trimed',label_trimed);
svm_loss = loss(svm_best, test_trimed.', test_label_trimed);


%% Build Decision Tree
tree = fitctree(train_image',train_label, 'MaxNumSplits',10,'CrossVal','on');
rfL = kfoldLoss(rfMdl, 'LossFun','ClassifErr');
view(tree.Trained{1},'Mode','graph');


%%
function [accuracy] = accurCal(test_data, train_data, test_label, train_label)
test_size = size(test_data, 2);
cnt = zeros(test_size, 1);
parfor test_digit = 1:1:test_size
    test_mat = repmat(test_data(:, test_digit), [1,size(train_data, 2)]);
    distance = sum(abs(test_mat - train_data).^2);
    [M,I] = min(distance);
    if train_label(I) == test_label(test_digit)
        cnt(test_digit) = cnt(test_digit) + 1;
    end
end
accuracy = double(sum(cnt)) / test_size;
end
```