Amath 482 Homework 4

Oscar Zhang

Mar.10th 2021

Abstract

In this assignment, we are going to discuss about how to train a machine to learn. Machine learning has been widely used in our life – it extracts some specific features of diverse data and category them into different categories. In this assignment, we are going to apply Single Value Decomposition and Linear Discriminant Analysis to our data to see how our algorithm recognize and classify our given data.

1. Introduction

In this assignment, we are going to train on images of digits from 0 to 9. The training data and test data are all from the MNIST dataset from the website http://yann.lecun.com/exdb/mnist/. For the data, we have 60000 training images and 10000 testing images. In order to test and train on the images, we are going to apply the following three techniques – Linear Discriminant Analysis(LDA), Support Vector Machines(SVM), and Decision Tree Learning(DTL). Before applying these three algorithm, we first need to reshape our SVD since SVD is restricted on translated data - we need to make the data centered at the mean and normalize it so that we can make SVD more effective. Then we can perform PCA to get a reduced projection of the center data. After these, we can process the three algorithm listed above to check how they performed and compare one to another.

2. Theoretical Background

SVD, known as Single Value Decomposition, is to transform one matrix, any dimensions n*m, into orthogonal directions. To be more specific,

$$A = U\Sigma V^*$$

where A is a n*m matrix, the column of U is the left singular vector of A, the column of V is the right singular vector of A. Σ is a diagonal matrix with the diagonal as the singular values of A, from large to small and all are non-negative. Also, the singular value is just the square of the eigenvalues of A^*A, which is the same eigenvalue of AA^*.

In order to find suitable projection using SVD, we need to apply the Linear Discriminant Analysis, LDA, as the following:

$$w = argmax \frac{w^T S_B w}{w^T S_w w}$$

Where $S_B$is the between-class scatter matrix and $S_\omega$is the within-class scatter matrix, and defined as the following:

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

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

Where μ is the overall mean and μ_j is the mean of the class j.
The vector w that maximizes the above quotient is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem:

$$S_B w = \lambda S_\omega w$$

Support Vector Machines, SVM, is an optimization problem defined as the following:

$$wx + b = 0$$

If the above equation is greater than 0, it will be considered as one class, if it's less than 0, it's the second class. By SVM, we can not only minimize the number of errors but also maximize the distance between each single data.

3. Algorithm Implementation and Development

We first load the data downloaded from the website and get two matrix – one by 784x60000 and the other one by 1x10000. Then we reshape our data by subtracting the mean of the dataset. Then we perform the SVD to the dataset. We also find that the rank after the SVD is 350. After reshaping the data, we are able to make some plots. Then we can apply the PCA to our data to project them on a 3-D graph. After we projecting the data on a 3-D graph, we can choose any two numbers to compare. The number I choose are 4 and 9, since they are very similar to each other. In order to compare it, we need to apply the LDA, which is an algorithm to compare two categories of data. In order to apply the LDA, we need to find the Sw and Sb. I first created two empty matrices. Then based on the formula above, we can easily find out what Sw and Sb are and then we can calculate the eigenvalue and therefore we can process LDA.

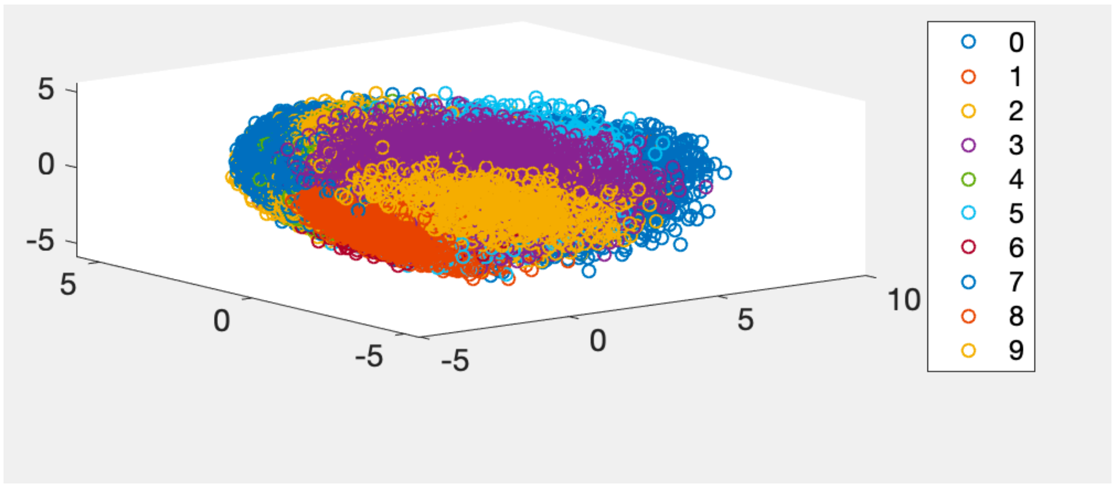

Figure 1: projections of all digits

4. Computational Results

After applying LDA, we are able to get the following graphs:

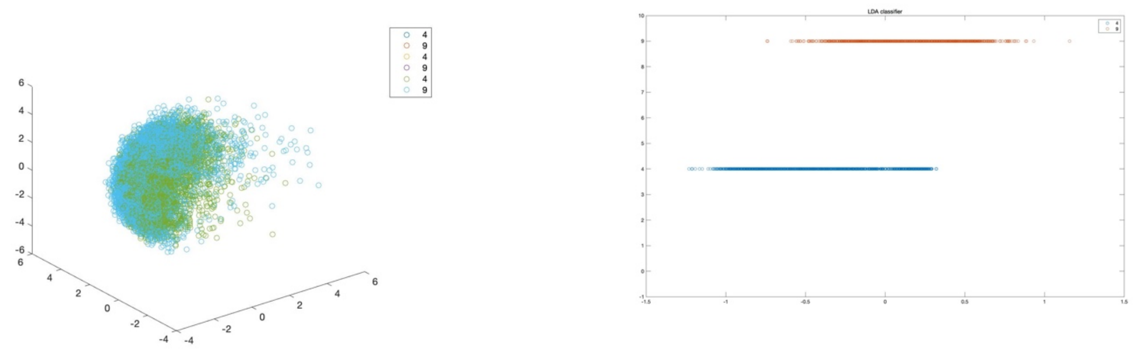

Figure 2: comparison between 4 & 9

From the above plots, it is obvious that the left handed side graph indicated that 4 and 9 have a very huge overlap – this is reasonable since when we are writing digits from 0 to 9, we can easily see that 4 and 9 has the most similar shape. Therefore, the 4 and 9 are the two digits that are the most difficult to separate. In fact, the accuracy of this LDA test is only 0.9354.

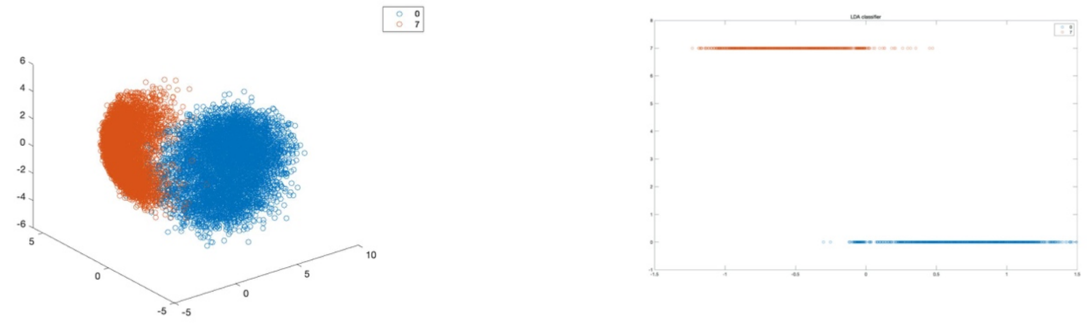

Figure 3: comparison between 0 & 7

From the above plots, we can see that digits 0 and 7 differ from each other the most. By the left handed side graph, we can see that these two digits only overlap each other for a very tiny part and the accuracy is 0.9931, which also indicated that 0 and 7 are the pair that is the easiest to separate.

However, all the data are above 90% accuracy, which states that the machine can distinguish the digits in the majority experiments.

5. Conclusions

In this assignment, we find that SVD is very crucial to classifier. Sometimes data is very large and complicated, we can apply SVD to lower the rank and only focus on the part that we need the most. We also find for classifying digits, LDA did a good job – all the accuracy are above 90% and the most difficult and easiest pairs it suggest(4&9 and 0&7) are quite reasonable. In addition, the plot it gave us was pretty straightforward.

Appendix I: Matlab Command

Reshape(A, a): reshape the array A to size a.
Size(x):return the dimension of matrix x
[U, S, V] = svd(A): perform SVD to matrix A and give us the result in the order of U, S, V
While loop: create a while loop to check if the result satisfies the standard. If it is, it continues the loop, if it is not, it exits the loop.
repmat(x, m, n): return a matrix with copies of the given matrix X in the shape of m * n.

Appendix II: Matlab Code

```
load fisheriris

[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 = im2double(reshape(train_image, size(train_image, 1) *
size(train_image, 2), []).');
train_label = im2double(train_label);
train_image = train_image';

test_image = im2double(reshape(test_image, size(test_image, 1) *
size(test_image, 2), []).');
test_label = im2double(test_label);
test_image = test_image';

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

[U, S, V] = svd(train_image, 'econ');

energy = 0;
total = sum(diag(S));

thres = 0.9;
r = 0;

while energy < thres
    r = r + 1;
    energy = energy + S(r, r) / total;
end

rank = r;

train_image = (U(:, 1:rank))' * train_image;
test_image = (U(:, 1:rank))' * test_image;

lamda = diag(S) .^ 2;

for i = [1,2]
    Projection_idx = train_image(:, find(train_label == i));
    plot3(Projection_idx(1,:), Projection_idx(2,:),
Projection_idx(3,:),...
        'o', 'DisplayName', num2str(i));
    hold on
end

scrsz = get(groot, 'ScreenSize');

tr = train_image;
te = test_image;


dimension = size(train_image,1);
Sw = zeros(dimension);
Sb = zeros(dimension);
N = size(train_image, 2);
Nt = size(test_image, 2);
Mu = mean(train_image, 2);

for i = [1, 2]
    mask = (train_label ==  i);
    x = tr(:, mask);
    ni = size(x, 2);
    pi = ni / N;
    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;
```

```matlab
[U, D, V] = svd(M);

G2 = U(:,1:rank);
Y2 = G2' * tr;

for number = [1,2]
   mask = (train_label ==  number);
   a = Y2(1,mask);
   b = Y2(2,mask);
   d = [a'; b'];
   plot(d, 1*number*ones(size(d)),'o',...
      'DisplayName', num2str(number)); hold on
   title(['LDA classifier']);

end
legend show


 ylim([-1 number+1])


na = 1;
nb = 2;

Y = G2' * tr;
Y_t = G2'* te;

train_n= Y(:, find(train_label == na|train_label ==nb));
test_n = Y_t(:, find(test_label == na |test_label ==nb));

accuracy = classifyNN(test_n, train_n,...
   test_label(find(test_label == na |test_label ==nb)), ...
   train_label(find(train_label == na |train_label ==nb)));


function [accuracy] = classifyNN(test_data, train_data, test_label,
train_label)

train_size = size(train_data, 2);
test_size = size(test_data, 2);
counter = zeros(test_size, 1);

parfor test_digit = 1:1:test_size
```

```matlab
    test_mat = repmat(test_data(:, test_digit), [1,train_size]);
    distance = sum(abs(test_mat - train_data).^2);
    [M,I] = min(distance);
    if train_label(I) == test_label(test_digit)
        counter(test_digit) = counter(test_digit) + 1;
    end
end

accuracy = double(sum(counter)) / test_size;
end
```