AMATH 482 HW4: Linear Analysis on MNIST Dataset

Tiffany Tang
Department of Applied Mathematics
University of Washington
leqitang@uw.edu

Abstract

In this paper, we look at famous MNIST data set which contains images of handwritten numbers 0 to 9. Several supervised machine learning classifiers were built to identify the digits presented in each image. First, SVD analysis of these digit images were performed and data are projected into PCA space. Then a linear classifier was built to identify the digits. Second, support vector machines and decision treen classifier are trained to classify images in test dataset. Lastly, the accuracy rate of above three classifiers are computed and compared.

1 Introduction and Overview

The MNIST database of handwritten digits is a large database of handwritten digits. This database is commonly used for training various image processing systems. The dataset used in this paper has a training set of 60000 examples and a test set of 10000 examples. The digits have been size-normalized and centered in 28*28-pixel images. Many types of machine learning methods have been used on the dataset in previous researches, including K-nearest neighbor, deep neural network and convolutional neural network, etc.

Singular value decomposition (SVD) that implement principal component analysis (PCA) is used to pick out some of the most important features of data. Then linear discriminant analysis (LDA) is computed to differentiate the images that contain different 2 or 3 digits. Support vector machines and decision tree classifiers are models that are trained through train datasets and give prediction to test dataset.

2 Theoretical Background

The singular valued decomposition is a method of decomposing a matrix into three other matrices:

$$A = USV^T$$

Where we have A as a m*n matrix. V as a n*n unitary matrix, U is m*m unitary matrix, and S is a m*n diagonal matrix with σ on its diagonals. V^T leads to first rotation of images matrix A, stretching it into a parallelogram with the diagonal matrix, and then rotating the parallelogram via the matrix U. The values σ_n on the diagonal matrix are the singular values of the matrix A. The singular values are nonnegative and ordered from largest to smallest.

Linear discriminant analysis (LDA) is a classifier method that find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data.

To classify 2 datasets with implementing LDA, we first calculate the means for each group for each feature:μ_1 and μ_2. Then we define between-class scatter matrix as:

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

This matrix measures the variance between the groups.

Then we define the within-class scatter matrix that measure the variance within each group:

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

Then a vector w is found:

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

The vector w is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem

$$S_B w = \lambda S_w w$$

We could use LDS to classify between more than two groups by make changes to the scatter matrices:

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

Where μ is the overall mean and μ_j is the mean of each of the N>2 groups. The within-class scatter matrix is :

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

Decision tree classifier is a non-parametric supervised learning method that predicts the value of a target variable by learning simple decision rules inferred from the data features. Decision trees learn from data to give approximation of test data classification. The deeper the tree, the more complex the decision rules and the fitter the model.

Support vector machine (SVM) is also a supervised learning method. The objective of SVM is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. This plane needs to have the maximum margin, i.e the maximum distance between data points of both classes.

3 Algorithm Implementation and Development

Three classifiers are investigated: linear classifier, decision tree classifier, and support vector machines through MNIST train and test dataset.

To prepare the data for analyzing, we first read and loaded the MNIST image database binary file of train data and test data by function `mnist_parse(path_to_digits, path_to_labels)`. The images data were reshaped into a column vector and each

column of the data matrix was a different image. Then SVD was performed to both train data and test data(`svd()`command). Singular value spectrum was plotted by acquiring the diagonal value of S diagonal matrix. The first 70 largest singular values were selected as PCA modes(features) in the following analysis. The projection onto principal components were computed by multiplying the S diagonal matrix and V unitary matrix. Several pairs of two digits are used to perform LDA. For LDA, we calculated the between-class scatter matrices and within-class variances as matrices. The best projection line was found by linear discriminant analysis (`eig(Sb, Sw)` command). The images of two digits were projected onto the line by multiplying w'. Then we set a threshold for our classifier. The threshold is the midpoint value of the values in the two different digit groups. In order to keep consistent for one digit below threshold, we made the smaller digit always below the other digit. A linear classifier for 3 digits was also performed with `classify()` command.
The accuracy of liner classifier was calculated by function `accuracy_rate_LDA.` In this function, we counted the number of correctly identified digit images for both digits and got the accuracy by dividing the total number of 2 digit images.
SVM classifier was performed with `fitcecoc()` command and the predicted labels for 10 digits are acquired with `predict()` command. Decision tree classifier was performed by `fitctree()` command and the predicted labels for 10 digits were also acquired by `predict()` command. The accuracy rate of both above classifier are computed in `accuracy_rate()` function. In this function, we counted the number of digits that were correctly identified and divided the total number of tested digit images to get the accracy rate for each classifier.

4 Computational Results

We plotted the projection onto 3 arbitrarily selected modes:2, 3, and 5 to have an idea which digits are easy to separate and which digits are hard to separate.
First, we performed the linear analysis for both the train and test dataset. The accuracy for differentiating digit 0 and 1 is 54.04% in test dataset which is similar to the accuracy rate in train dataset. 0 and 1 is the most distinguishable digits. The accuracy for differentiating digit 3 and 5 is 53.66% in test dataset which is similar to the accuracy rate in the train dataset. 3 and 5 is the hardest pair to separate. The accuracy rate for identifying 3 digits was also computed by LDA. The accuracy rate of identifying 0, 1, and 2 are 32.38% in the train dataset, which is much lower than the accuracy of differentiating two digits.
After training decision tree classifier, I used the classifier to check the prediction for test set. The accuracy rate of identifying 3 and 5 is 80.86% and the accuracy rate of identifying 0 and 1 is 94.27%.
From the SVM classifier, we have the accuracy rate of identifying 3 and 5 is 91.42% and the accuracy rate of identifying 0 and 1 is 96.57%.

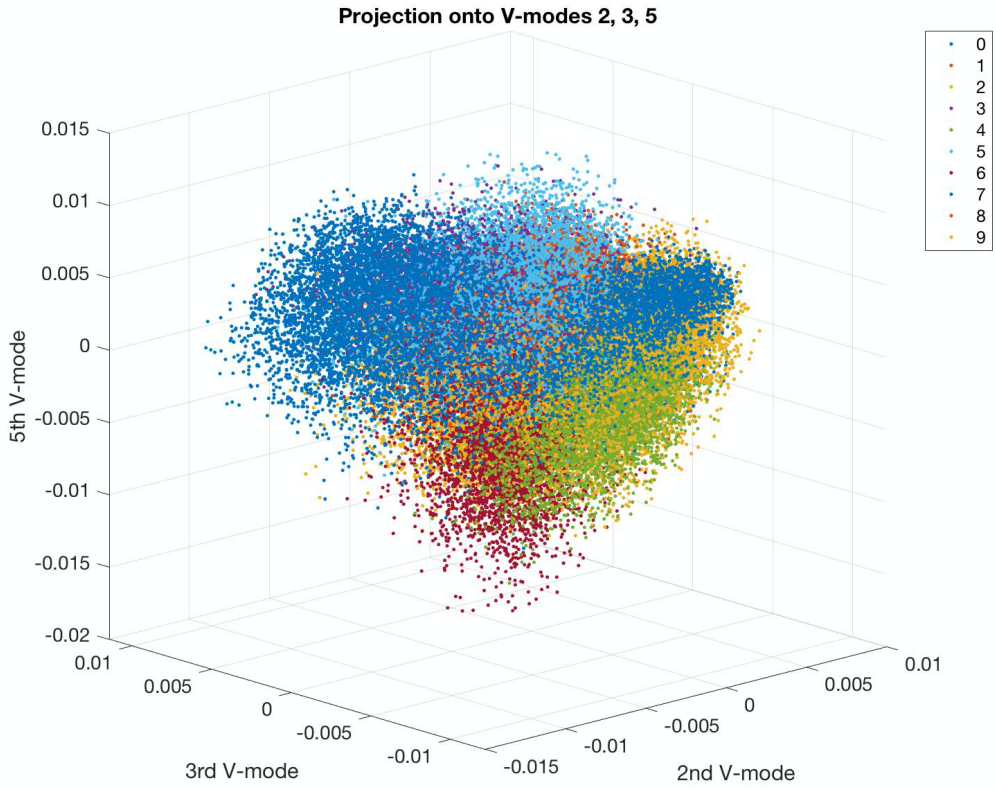

figure 1: projection onto columns 2, 3, and 5 V-modes colored by digit labels.

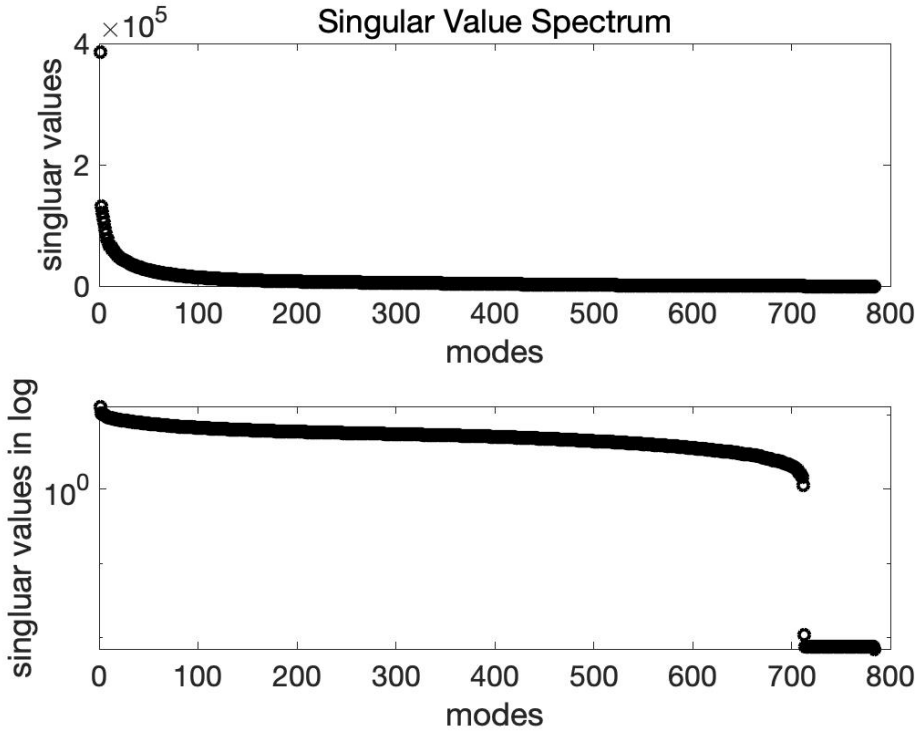

Figure 2: Singular value spectrum in regular scale and log scale.

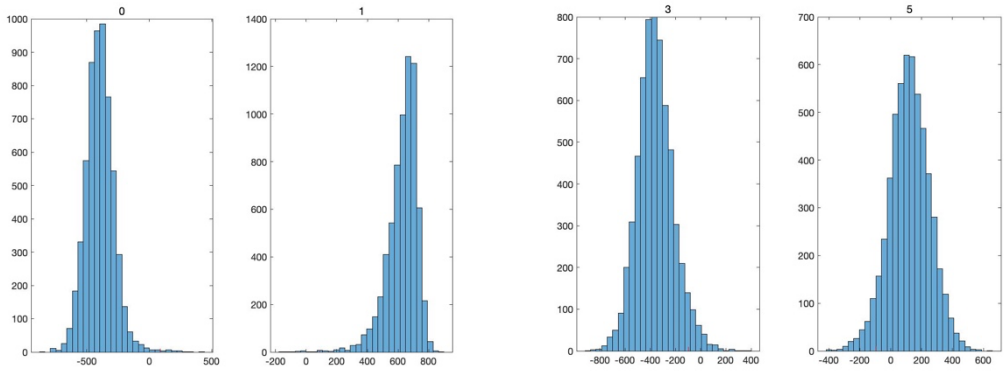

Figure 3: the left figure presents the histogram of the distribution of the projection value of 0 and 1 separately. A red vertical line of threshold has been showed.
Figure 4: the right figure presents the histogram of the distribution of the projection value of 3 and 5 separately. A red vertical line of threshold has been showed.

5 Summary and Conclusions

By perform linear classifier, SVM classifier, and decision tree classifier, we obtained how these machine learning models could classifier different types of images like human do. However, each of them has different classifier ability. The accuracy rate of SVM and decision tree classifier are significantly higher than the accuracy rate of linear classifier. And the accuracy rate of applying SVM classifier is higher than the decision tree classifier. But, the runtime of SVM classifier is significantly larger than decision tree and LDA classifier. Therefore, further investigation of the optimal runtime for each classifier could be researched.

Appendix A. MATLAB Functions

1. [images, labels] = mnist_parse (path_to_digits, path_to_labels) returns images and labels for images from reading MNIST image database binary file
2. [U, S, V] = svd(A) performs a singular value decomposition of matrix A.
3. K = find(X) returns a vector containing the linear indices of each nonzero element I array X.
4. D = diag(v) returns a square diagonal matrix with the element of vector v on the main diagonal.
5. semilogy(X,Y) plots x- and y-coordinates using a linear scale on the x-axis and a base-10 logarithmic scale on the y-axis.
6. e = eig(A) returns a column vector containing the eigenvalues of square matrix A.
7. Mdl = fitcecoc(Tbl, Labels) returns a full, trained, multiclass, error-correcting output codes (ECOC) model using the predictors in table Tbl and the class labels of Tbl.
8. tree = fitctree(Tbl, Labels) returns a fitted binary classification decision tree based on the input variables (also known as predictors, features, or attributes) contained in the table Tbl and output (response or labels) contained in Labels.

Appendix B. Matlab Code

```
clc; clear all;close all;
% load train data
[images, labels] = mnist_parse('train-images-idx3-ubyte', 'train-
labels-idx1-ubyte');
images = double(reshape(images, size(images,1)*size(images,2), []));
images = double(images);
% load test data
[images_t, labels_t] = mnist_parse('t10k-images-idx3-ubyte', 't10k-
labels-idx1-ubyte');
images_t = double(reshape(images_t,
size(images_t,1)*size(images_t,2), []));
images_t = double(images_t);
% svd

[U,S,V] = svd([images],'econ');
[Ut, St, Vt] = svd([images_t], 'econ');
% singular value
figure(1)
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',2)
title('Singular Value Spectrum')
set(gca,'Fontsize',16,'Xlim',[0 800])
xlabel('modes')
ylabel('singluar values')
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',2)
set(gca,'Fontsize',16,'Xlim',[0 800])
xlabel('modes')
ylabel('singluar values in log')
%% Projection onto 3 V-modes

for label=0:9
    label_indices = find(labels == label);
    plot3(V(label_indices, 2), V(label_indices, 3), V(label_indices,
5),...
        '.', 'DisplayName', sprintf('%i',label), 'Linewidth', 2)
    hold on
end
figure(2)
xlabel('2nd V-Mode'), ylabel('3rd V-Mode'), zlabel('5th V-Mode')
title('Projection onto V-modes 2, 3, 5')
legend('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')
```

```
set(gca,'Fontsize', 14)
%% pick feature
feature = 70;

projection = S*V'; % projection onto principal components: X = USV' -
-> U'X = SV'
TestMat = U(:, 1:feature)'* images_t;
%% LDA
label_0 =find(labels == 0);
proj_0 = projection(1:feature,label_0);
label_1 = find(labels == 1);
proj_1 = projection(1:feature, label_1);

% Calculate scatter matrices

m0 = mean(proj_0,2);
m1 = mean(proj_1,2);

Sw = 0; % within class variances
for k = 1:size(label_0)
    Sw = Sw + (proj_0(:,k) - m0)*(proj_0(:,k) - m0)';
end
for k = 1:size(label_1)
   Sw =  Sw + (proj_1(:,k) - m1)*(proj_1(:,k) - m1)';
end

Sb = (m0-m1)*(m0-m1)'; % between class
% Find the best projection line

[V2, D] = eig(Sb,Sw); % linear disciminant analysis
[lambda, ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);

% Project onto w

v0 = w'*proj_0;
v1 = w'*proj_1;

% Make one digit below the threshold

if mean(v0) > mean(v1)
    w = -w;
    v0 = -v0;
```

```
    v1 = -v1;
end

% Find the threshold value

sort0 = sort(v0);
sort1 = sort(v1);

t1 = length(sort0);
t2 = 1;
while sort0(t1) > sort1(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold = (sort0(t1) + sort1(t2))/2;

% Plot histogram of results

figure(5)
subplot(1,2,1)
histogram(sort0,30); hold on, plot([threshold threshold], [0 10],'r')
%set(gca,'Xlim',[-3 4],'Ylim',[0 10],'Fontsize',14)
title('8')
subplot(1,2,2)
histogram(sort1,30); hold on, plot([threshold threshold], [0 10],'r')
%set(gca,'Xlim',[-3 4],'Ylim',[0 10],'Fontsize',14)
title('9')
accuracy_LDA = accuracy_rate_LDA(w, TestMat, labels_t, 0, 1,
threshold );


%% a linear classifier that identify 3 arbitrarily picked digits
projection_3 = projection(1:feature, :);
projection_3= projection(find(labels == 0| labels == 1| labels ==
2));
labels_3 = labels(find(labels == 0| labels == 1| labels == 2));
test_3 = TestMat(find(labels_t == 0| labels_t == 1| labels_t == 2));
labels_3t = labels_t(find(labels_t == 0| labels_t == 1| labels_t ==
2));
class = classify(test_3, projection_3, labels_3, 'linear');
count = 0;
for j = 1:length(test_3)
    if class(j) == labels_3t(j)
        count = count+1;
```

```
    end
end
accuracy_3 = count/length(test_3)

%% SVM classifier with training data, labels and test set

Mdl = fitcecoc(projection(1:feature, :)', labels);
predict_labels_svm = predict(Mdl, TestMat');
%% prediction with svm classifier
pairs01_svm = accuracy_rate(predict_labels_svm, labels_t, 0, 1);
pairs35_svm = accuracy_rate(predict_labels_svm, labels_t, 3, 5);
%% decision tree classifier
tree = fitctree(projection(1:feature, :)', labels);
predict_labels_tree = predict(tree, TestMat');

%% prediction with decision tree classifier
pairs01_tree = accuracy_rate(predict_labels_tree, labels_t, 0, 1);
pairs35_tree = accuracy_rate(predict_labels_tree, labels_t, 3, 5);
%% Accuracy rate for LDA
function accuracy_LDA = accuracy_rate_LDA(w, TestMat, labels_t,
digit1, digit2, threshold)
    pval = w'*TestMat;
    digit1_t = find(labels_t ==digit1);
    digit2_t = find(labels_t == digit2);
    count = 0;
    index = 0;
    for j = 1:length(digit1_t)
        index = index+1;
        if pval(digit1_t(index)< threshold)
            count = count +1;
        end
    end
    index1 = 0;
    for j = 1:length(digit2_t)
        index1 = index1 + 1;
        if pval(digit2_t(index) > threshold)
            count = count +1;
        end
    end
    accuracy_LDA = count/(length(digit1_t)+length(digit2_t));
end
%% function for calculating accuracy rate for desicion tree and SVM
classifer
function accuracy = accuracy_rate(predict_labels, labels_t, digit1,
```

```
digit2)
    predict_digit = predict_labels(find(labels_t== digit1 | labels_t
== digit2));
    test_digit = labels_t(find(labels_t== digit1 | labels_t ==
digit2));
    count = 0;
    for i = 1: length(predict_digit)
        if predict_digit(i) == test_digit(i)
            count = count+1;
        end
    end
    accuracy = count/length(test_digit);
end
```