AMATH 482 Homework 4

Fiona Fu
March 2020

Abstract

Reshaping the training digit images by Singular Value Decomposition (SVD) analysis and apply on the test images. Building a classifier by Linear Discriminant Analysis (LDA) to identify individual digits in training set.

1 Introduction and Overview

Starting by performing an analysis on the training data, we make analysis with specific interpretations by applying the method listed below on both data and compare their performance:

1.1 SVD (Singular Value Decomposition) analysis

By reshaping each image into column vectors, we can find the **U, Σ, V** matrices.

1.2 LDA (Linear Discriminant Analysis) classifier

After having the data projected into PCA space, we can build a linear classifier to identify each digit in training data.

1.3 SVM (Support Vector Machines) & Decision trees classifier

See how this classifier works for separate between all ten digits. Compare this method and LDA.

2 Theoretical Background

In this assignment, we are applying **Singular Value Decomposition (SVD)** to do the analysis on the matrix representing the dimension of images ($m \times n$). We construct a matrix **U** from $\boldsymbol{U}$^ by adding an addition *m* - *n* columns that are orthonormal to the already existing set **U**^. In this way, the matrix **U** becomes a square $m \times m$ matrix, and in order to make the decomposition work,
and additional *m* - *n* rows of zeros is also added to the matrix **Σ**^, resulting in the matrix **Σ**. This leads to the **singular value decomposition** of the matrix **A**

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

By definition, the value on the diagonal of **Σ** are called the **singular values** of the matrix **A**. The vectors **u***n* which make up the columns of **U** are called the **left singular vectors** of **A**. The vectors **v***n* which make up the columns of **V** are called the **right singular vectors** of **A**.

3 Algorithm Implementation and Development

We first need to load the training data. Then we will reshape each image into a column vector that each column represents a different image.

```
[images, labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte');
images = reshape(images, [28*28], 60000);
images = double(images);
```

Then we will apply the MATLAB build-in function svd() to do the SVD analysis of the digit images.

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

To find the dominant features and visualize necessary modes, we plot the singular value spectrum.

```
figure(1)
plot(diag(S)/sum(diag(S)), 'r*','Linewidth',[2] )
ylabel('Singular Value')
xlabel('Images')
title('Singular Value Spectrum')
```

In order to project onto three selected V-modes, we use the following code that by locating the labels that are the same to the ten digits.

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

After projected onto PCA space, we use **S*V'** for the data training for all the following classifier of LDA, tree, and SVM.

In order to find the right line for projection of data in LDA, we need to maximize the between-class scatter matrix and minimize within-class scatter matrix. Then, we use the code to find the best projection line:

```
[V2, D] = eig(between class variance, within class variance) → LDA
[lambda, ind] = max(abs(diag(D)));
W = V2(:,ind);
W = w/norm(w,2);
```

For SVM classifier, we use the following code:

```
Md1 = fitcsvm(training data, labels);
Label = predict(Md1, test data);
```

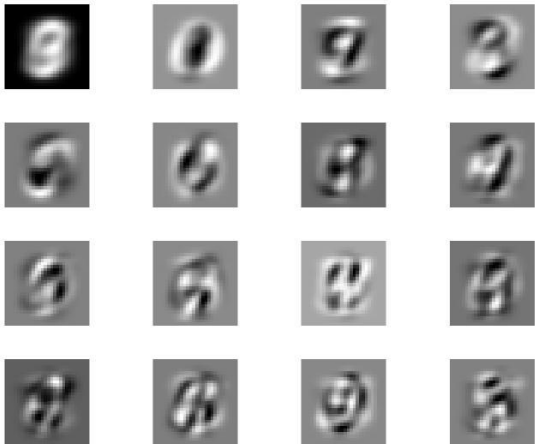
And implement the code for tree classifiers by the built-in code: tree = fitctree().

To calculate the accuracy of each methods, we have similar code that check if each label from the classifier and the label in the test data are the same. When label of predict(classifier, test data) == label of test data, we add 1 for each equality and accuracy equals to the sum divided by the rank, which is 10 in this case.

4 Computational Results

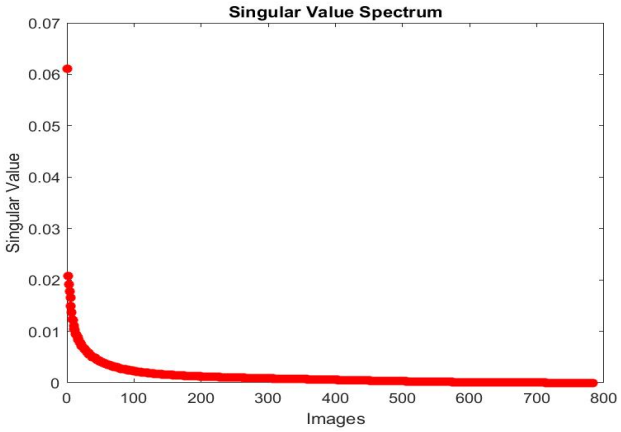
4.1: SVD Analysis

By using SVD to decompose the image matrix and reshape each image into column vectors, we obtained the dominant features and apply SVD to those edge images. Figure 1 shows the first 16 columns of the images collected in training data.



(Figure 1)

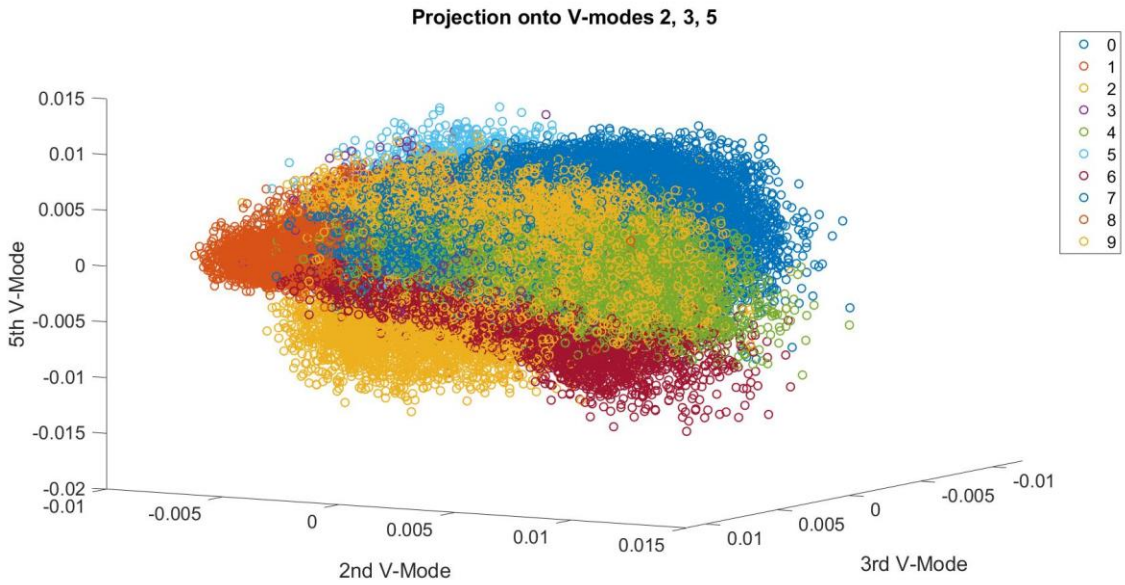
Then, we plot the singular values spectrum (Figure 2) for the training data, and we can approximate rank r = 10. In other words, we are able to get a relatively good image for the images by using the first ten columns.



(Figure 2)

By projecting the data onto columns 2, 3, 5 on a 3D plot (Figure 3), we can find clusters of

data points and each represents one of the digits as labeled in different colors.



(Figure 3)

This helps us to build classifier in the next step to identify individual digits in training set.

4.2: LDA Classifier

For the two-digits LDA, I got the following results (approximately):

Digits	0	1	2	3	4	5	6	7	8	9
Accuracy	92%	46%	22%	46%	90%	24%	46%	56%	57%	78%

4.3: SVM & Tree Classifier

Digits	0	1	2	3	4	5	6	7	8	9
Accuracy	60%	70%	40%	60%	90%	70%	40%	60%	60%	70%

3 Summary and Conclusions

We use SVD to decompose the matrix that can help us to identify the dominant features and implement more further projections and analysis on the images. We plot the singular value spectrum based on the diagonal extracted from the SVD method, and we can get a clear sense about how many modes are necessary for a good image reconstruction, r = 10 in this case. Then we project the training data for the machine learning process onto a 3D plot that contain various colors that each represent a digit.

After we project the data into PCA space, we will build different classifier to check individual digits in both the training set and the test set. Comparing the performance of between the methods, the hardest pair of digit is 2, 5 and the easiest one is 0, 4. The SVM seems to have higher accuracy.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[2] MathWork. https://www.mathworks.com/help/matlab/ref/max.html

[3] Jason J. Bramburger, Lecture Notes for AMATH482, Winter, 2021

[4] Prof. Bramburger, Slack, assignment-4

Appendix A MATLAB Functions

- fitcsvm trains or cross-validates a support vector machine (SVM) model for one-class and two-class (binary) classification on a low-dimensional or moderate-dimensional predictor data set.
- tree = fitctree(Tbl,ResponseVarName) 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 Tbl.ResponseVarName.

Appendix B MATLAB Code

```
% clear all, clc, close all
% [images, labels] = mnist_parse('train-images.idx3-ubyte', 'train-labels.idx1-ubyte');
% images = reshape(images, [28*28], 60000);
% images = double(images);
% [U, S, V] = svd(images, 'econ');
% figure(1)
% for k = 1:9
%     subplot(3,3,k)
%     ut1 = reshape(U(:,k),28,28);
%     ut2 = rescale(ut1);
%     imshow(ut2)
% end
% figure(2)
% plot(diag(S)/sum(diag(S)), 'r*','Linewidth',[2] )
% ylabel('Singular Value')
% xlabel('Images')
% title('Singular Value Spectrum')
%
% % Projection onto 3 V-modes
% figure(3)
% for label = 0:9
%     label_ind = find(labels == label);
%     plot3(V(label_ind, 2), V(label_ind, 3), V(label_ind, 5),...
%         'o','DisplayName', sprintf('%i', label), 'Linewidth', 1);
%     hold on
% end
% xlabel('2nd V-Mode'), ylabel('3rd V-Mode'), zlabel('5th V-Mode')
% title('Projection onto V-modes 2, 3, 5')
% legend
% set(gca, 'Fontsize', 14)
% rank = 10
```

```
% proj = S*V';
% % figure(4)
% % feature = 10;
% % U_new = U(:, 1:10);
% % for k = 1:9
% %     subplot(3,3,k)
% %     ut1 = reshape(U_new(:,k),28,28);
% %     ut2 = rescale(ut1);
% %     imshow(ut2)
% % end
% % nd = size(images, 2);
% % proj = S*V';
% % final = proj(1:feature, 1:nd);
% % nd = mean(final, 2)
% % Sw = 0;
% % for k = 1:nd
% %     Sw = Sw +(images(:,k) - nd)*(images(:,k)-nd)';
% % end
% [test_images, test_labels] = mnist_parse('t10k-images.idx3-ubyte', 't10k-labels.idx1-
ubyte');
% test_images = reshape(test_images, [28*28], 10000);
% test_images = double(test_images);
% % classification tree
% % tree = fitctree(images, labels, 'MaxNumSplits',3,'CrossVal','on');
% Md1 = fitcsvm(images, labels);
% label = predict(Md1, test_images)
% good = 0;
% for i = 1:length(label)
%     if label(i) == test_labels(i)
%         good = good + 1;
%     end
% end
% accuracy = good/rank;
% print('Accuracy of SVM is', num2str(accuracy));
%
% wave = dc_wavelet(images);
% pca = U'*wave;
% val = w'*pca;
```

Listing: Excerpt Code from MATLAB