Project 2 – Linear Classifier with Least Square and Fisher Discriminant Analysis

Ming-Ju Li

February 13, 2018

Abstract

To find the linear classifier using the two methods, least square approach, and linear discriminant analysis (Fisher's discriminant analysis).

Contents

1 Introduction

Their are two ways in machine learning that are used to distinguish the data, depends on what data is using. In continuous data, we use regression. For discrete data type, we use classification. In this project, we employ two linear classification approaches, least square and Fisher discriminant , to separate the data points. Here we are doing supervised learning, where the class labels are assigned to each data point.

1.1 Data

In this project, three different datasets are used. 'Wine' dataset, which contains 3 different classes, 90 data points, and 13 dimensions, forming a matrix of training data,$\mathbf{x_n}$, of size 90x13. 'Wallpaper' dataset, with 17 classes, 1700 data points, and dimension D of 100. For 'Taiji' dataset, there are 8 different classes, 11361 data points, and 64 dimensions.

2 Least Square Approach

The first part of this project is to classify the data using least square approach. To find the boundaries, I follow the equation

$$y_k(x) = \mathbf{w}_k^T\mathbf{x} + w_{k0} \tag{1}$$

which can be conveniently denoted as,

$$\mathbf{y}(x) = \tilde{\mathbf{W}}^T\tilde{\mathbf{x}} \tag{2}$$

where $\tilde{\mathbf{W}}$ is a matrix whose k^{th} column comprises $D+1$ dimensional vector $\tilde{w}_k = (w_{k0}, \mathbf{w}_k^T)^T$ and $\tilde{\mathbf{x}}$ is the corresponding augmented input vector $(1, \mathbf{x}^T)^T$, with a dummy input $x_0 = 1$

2.1 Mathematic Derivation

Now, in order to determine the the parameter matrix $\tilde{\mathbf{W}}$, we minimize the sum-of-square err function

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2}Tr\{(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})\} \tag{3}$$

where $\mathbf{T}$ is the target matrix whose k^{th} column of n^{th} row is 1 if the row belongs to the k^{th} class.

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \end{bmatrix} \tag{4}$$

Take 'wine' dataset for example, the corresponding target matrix is

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Let us go back to the maximization of $\tilde{\mathbf{W}}$. Take partial derivatives with respect to the $\tilde{\mathbf{W}}$

$$\begin{aligned} \frac{\partial E(\tilde{\mathbf{W}})}{\partial \tilde{\mathbf{W}}} &= \frac{\partial[(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})]}{\partial \tilde{\mathbf{W}}} && (6) \\ &= \tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) && (7) \\ &= \tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \tilde{\mathbf{X}}^T\mathbf{T} && (8) \\ &= \tilde{\mathbf{X}}(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) && (9) \end{aligned}$$

and set the equation to be zero then we yield the $\tilde{\mathbf{W}}$ minimizing the err function,

$$\tilde{\mathbf{X}}(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) = 0 \tag{10}$$

$$\begin{aligned} \Rightarrow \tilde{\mathbf{W}} &= (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{T}} && (11) \\ &= \tilde{\mathbf{X}}^\dagger\tilde{\mathbf{T}} && (12) \end{aligned}$$

where $\tilde{\mathbf{X}}^\dagger = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T$ is the pseudo-inverse of $\tilde{\mathbf{X}}$. The discriminant function is then become

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T\tilde{\mathbf{x}} = \mathbf{T}^T(\tilde{\mathbf{X}}^\dagger)^T\tilde{\mathbf{x}} \tag{13}$$

To classify the test data, simply multiply the data with the optimized weight matrix $\tilde{\mathbf{W}}$ and extract the largest value of each column.

2.2 Result

As you can see in the Table 1, the classifier works quite well on the datasets. The confusion matrices show reasonable outcomes. The diagonal entries of each confusion matrix is the largest among all the entries of its column. Also, the column-wise direction represents the predicted class label of test data and the row-wise direction represents the true class label. For example, the confusion matrix of 'wine' dataset in Table 2, the (1,1) entry indicates the number of class 1 on the test data being classified as class 1 and the (3,2) entry indicates that there are 2 test data belong to class 3 been classified as class 2.

Dataset	Accuracy	Standard Deviation
Wine	0.9867	0.0231
Wallpaper	0.5831	0.1873
Taiji	0.9126	0.0669

Table 1: Accuracy and the standard deviation of different dataset under least square approach

$$\begin{pmatrix} 29 & 0 & 0 \\ 0 & 33 & 0 \\ 0 & 2 & 24 \end{pmatrix} \qquad \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0.04 & 0.96 \end{pmatrix}$$

Table 2: Left is the confusion and the right is the classification matrix of the 'wine' dataset

$$\begin{pmatrix}
12 & 12 & 2 & 7 & 0 & 9 & 0 & 1 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
18 & 19 & 2 & 10 & 0 & 6 & 3 & 4 & 3 & 14 & 2 & 0 & 0 & 0 & 1 & 1 & 0 \\
4 & 2 & 61 & 2 & 0 & 4 & 14 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\
21 & 15 & 0 & 37 & 0 & 4 & 2 & 6 & 0 & 9 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 90 & 3 & 0 & 1 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 10 & 3 & 4 & 0 & 36 & 2 & 0 & 1 & 5 & 11 & 0 & 1 & 0 & 1 & 0 & 0 \\
2 & 7 & 26 & 6 & 0 & 9 & 70 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
5 & 8 & 0 & 11 & 0 & 2 & 1 & 46 & 0 & 3 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
5 & 3 & 1 & 0 & 2 & 3 & 0 & 0 & 63 & 1 & 2 & 0 & 1 & 0 & 0 & 0 & 4 \\
11 & 8 & 1 & 5 & 0 & 8 & 1 & 0 & 0 & 24 & 9 & 0 & 0 & 0 & 1 & 0 & 1 \\
2 & 3 & 0 & 3 & 0 & 11 & 1 & 0 & 1 & 21 & 69 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 2 & 1 & 4 & 0 & 0 & 2 & 37 & 0 & 6 & 2 & 93 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 1 & 1 & 4 & 2 & 1 & 0 & 1 & 5 & 1 & 0 & 71 & 2 & 12 & 2 & 0 \\
5 & 3 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 11 & 98 & 1 & 0 & 0 \\
6 & 2 & 1 & 4 & 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 & 14 & 0 & 81 & 2 & 0 \\
1 & 1 & 0 & 1 & 2 & 0 & 1 & 1 & 6 & 1 & 0 & 0 & 1 & 0 & 3 & 86 & 9 \\
0 & 1 & 1 & 2 & 0 & 2 & 2 & 0 & 4 & 2 & 2 & 0 & 0 & 0 & 0 & 7 & 82
\end{pmatrix}$$

Table 3: Confusion matrix of the 'wallpaper' dataset. The diagonal has the largest value of the corresponding column, which suggests that the classifier is accurate.

$$\begin{pmatrix}
0.2449 & 0.2449 & 0.04082 & 0.1429 & 0 & 0.1837 & 0 & 0.02041 & 0 & 0.102 & 0 & 0 & 0 & 0 & 0 & 0.02041 & 0 \\
0.2169 & 0.2289 & 0.0241 & 0.1205 & 0 & 0.07229 & 0.03614 & 0.04819 & 0.03614 & 0.1687 & 0.0241 & 0 & 0 & 0 & 0.01205 & 0.01205 & 0 \\
0.04211 & 0.02105 & 0.6421 & 0.02105 & 0 & 0.04211 & 0.1474 & 0.01053 & 0.02105 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01053 & 0.04211 \\
0.2188 & 0.1562 & 0 & 0.3854 & 0 & 0.04167 & 0.02083 & 0.0625 & 0 & 0.09375 & 0.02083 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.008929 & 0.008929 & 0 & 0 & 0.8036 & 0.02679 & 0 & 0.008929 & 0.1429 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.03896 & 0.1299 & 0.03896 & 0.05195 & 0 & 0.4675 & 0.02597 & 0 & 0.01299 & 0.06494 & 0.1429 & 0 & 0.01299 & 0 & 0.01299 & 0 & 0 \\
0.016 & 0.056 & 0.208 & 0.048 & 0 & 0.072 & 0.56 & 0.016 & 0.008 & 0.008 & 0 & 0 & 0.008 & 0 & 0 & 0 & 0 \\
0.06024 & 0.09639 & 0 & 0.1325 & 0 & 0.0241 & 0.01205 & 0.5542 & 0 & 0.03614 & 0 & 0.08434 & 0 & 0 & 0 & 0 & 0 \\
0.05882 & 0.03529 & 0.01176 & 0 & 0.02353 & 0.03529 & 0 & 0 & 0.7412 & 0.01176 & 0.02353 & 0 & 0.01176 & 0 & 0 & 0 & 0.04706 \\
0.1594 & 0.1159 & 0.01449 & 0.07246 & 0 & 0.1159 & 0.01449 & 0 & 0 & 0.3478 & 0.1304 & 0 & 0 & 0 & 0.01449 & 0 & 0.01449 \\
0.01802 & 0.02703 & 0 & 0.02703 & 0 & 0.0991 & 0.009009 & 0 & 0.009009 & 0.1892 & 0.6216 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.02649 & 0.01325 & 0.006623 & 0.02649 & 0 & 0 & 0.01325 & 0.245 & 0 & 0.03974 & 0.01325 & 0.6159 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.0283 & 0.009434 & 0.009434 & 0.03774 & 0.01887 & 0.009434 & 0 & 0.009434 & 0.04717 & 0.009434 & 0 & 0.6698 & 0.01887 & 0.1132 & 0.01887 & 0 \\
0.03968 & 0.02381 & 0 & 0.02381 & 0.01587 & 0 & 0 & 0 & 0 & 0.02381 & 0 & 0 & 0.0873 & 0.7778 & 0.007937 & 0 & 0 \\
0.05263 & 0.01754 & 0.008772 & 0.03509 & 0 & 0.008772 & 0 & 0.008772 & 0.01754 & 0 & 0 & 0 & 0.1228 & 0 & 0.7105 & 0.01754 & 0 \\
0.00885 & 0.00885 & 0 & 0.00885 & 0.0177 & 0 & 0.00885 & 0.00885 & 0.0531 & 0.00885 & 0 & 0 & 0.00885 & 0 & 0.02655 & 0.7611 & 0.07965 \\
0 & 0.009524 & 0.009524 & 0.01905 & 0 & 0.01905 & 0.01905 & 0 & 0.0381 & 0.01905 & 0.01905 & 0 & 0 & 0 & 0 & 0.06667 & 0.781
\end{pmatrix}$$

Table 4: Classification matrix of 'Wallpaper' dataset

$$\begin{pmatrix}
256 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
32 & 369 & 0 & 0 & 0 & 0 & 0 & 0 \\
49 & 0 & 738 & 0 & 0 & 0 & 0 & 0 \\
26 & 0 & 0 & 369 & 0 & 0 & 0 & 0 \\
104 & 0 & 0 & 0 & 369 & 0 & 0 & 0 \\
47 & 0 & 0 & 0 & 0 & 738 & 0 & 0 \\
25 & 0 & 0 & 0 & 0 & 0 & 369 & 0 \\
64 & 0 & 0 & 0 & 0 & 0 & 0 & 369
\end{pmatrix}$$

Table 5: Confusion matrix of 'Taiji' dataset. There are some mis-classified data to class 1.

$$\begin{pmatrix}
1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.0798 & 0.9202 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.06226 & 0 & 0.9377 & 0 & 0 & 0 & 0 & 0 \\
0.06582 & 0 & 0 & 0.9342 & 0 & 0 & 0 & 0 \\
0.2199 & 0 & 0 & 0 & 0.7801 & 0 & 0 & 0 \\
0.05987 & 0 & 0 & 0 & 0 & 0.9401 & 0 & 0 \\
0.06345 & 0 & 0 & 0 & 0 & 0 & 0.9365 & 0 \\
0.1478 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8522
\end{pmatrix}$$

Table 6: Classification Matrix of 'Taiji' dataset. The most error rate happens at the middle class. The 5^{th} class has the largest rate of being mis-classified to the 1^{th} class.

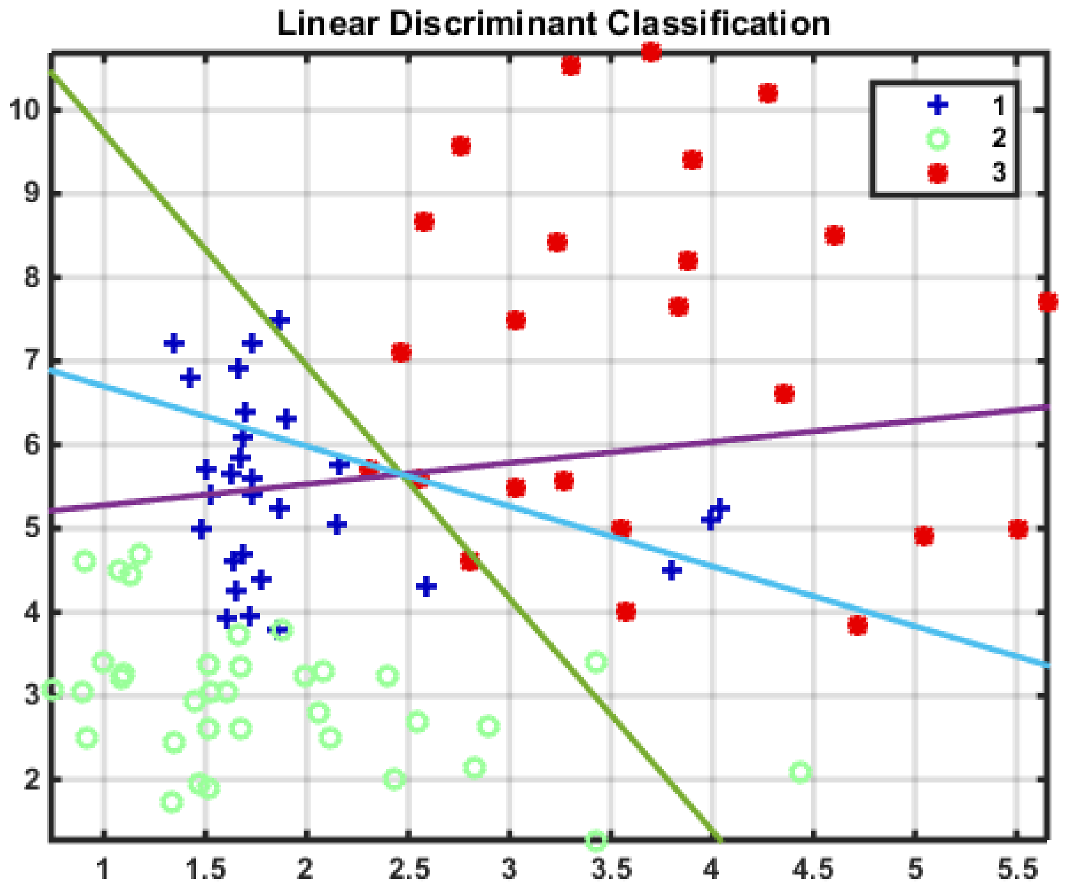


Figure 1: **Linear classifier using least square on 'wine' dataset**

3 Fisher's Discriminant Analysis

The "Fisher's Discriminant Analysis", or the "Linear Discriminant Analysis" is a way to deal with the problem of using least square to find the classifier, which is prone to the outliers. Therefore, we have to come up with an approach.For K>2 classes, assume that the dimensionality of the input space is greater than the number K of classes. We can then generalize the Fisher's discriminant by grouping the linear features $y_K = \mathbf{w}_k^T\mathbf{x}$, where $k = 1, \ldots, D$ Same as the least square, we start from the equation

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} \tag{14}$$

but instead of minimizing the err function, we now project the data onto a lower dimensional plane. However, randomly project the data onto a lower dimensional plane may cause overlapping in the data that are well separated in the original dimension.

3.1 Mathematic Derivation

The Fisher's discriminant suggests that we project the data onto a dimension that the classes are separated the most, but the data of the same class be the most compact. That is, we want the between class covariance, $\mathbf{S}_W$, to be as large as possible, while the within class covariance, $\mathbf{S}_B$, to be as small as possible. The two covariance are expressed respectively,

$$\mathbf{S}_W = \sum_{k=1}^{K}\sum_{n\in C_k}(\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \tag{15}$$

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T \tag{16}$$

where

$$\mathbf{m}_k = \frac{1}{N_k}\sum_{n\in C_k}\mathbf{x}_n \tag{17}$$

and the mean of the total dataset is

$$\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \frac{1}{N}\sum_{k=1}^{K}N_k\mathbf{m}_k \tag{18}$$

And the criteria that we desired will become

$$\begin{aligned} J(\mathbf{w}) &= Tr\{(\mathbf{W}\mathbf{S}_W\mathbf{W}^T)^{-1}(\mathbf{W}\mathbf{S}_B\mathbf{W}^T)\} \\ \Rightarrow J(\mathbf{w}) &= Tr\{(\mathbf{W}^T)^{-1}\mathbf{S}_W^{-1}\mathbf{W}^{-1}\mathbf{W}\mathbf{S}_B\mathbf{W}^T\} \\ &= Tr\{(\mathbf{W}^T)^{-1}\mathbf{S}_W^{-1}\ \mathbf{I}\ \mathbf{S}_B\mathbf{W}^T\} \\ &= Tr\{(\mathbf{W}^T)^{-1}\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{W}^T\} \end{aligned}$$

The $\mathbf{W}_{opt}$ that maximize the criteria is the eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S}_B$ that corresponds to the largest D' eigenvalues. We now find the hyperplane that we are able to project our

Dataset	Accuracy	Standard Deviation
Wine	0.9025	0.049
Wallpaper	0.7093	0.2338
Taiji	0.9513	0.0277

Table 7: Accuracy and the standard deviation of different dataset after Fisher projection.

$$\begin{pmatrix} 29 & 0 & 0 \\ 0 & 34 & 1 \\ 0 & 1 & 23 \end{pmatrix} \qquad \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 0.9714 & 0.02857 \\ 0 & 0.04167 & 0.9583 \end{pmatrix}$$

Table 8: Left is the confusion and the right is the classification matrix of 'wine' dataset using the Fisher discriminant. Comparing to the classification matrix under least square approach, the Fisher projection has less accuracy. This is because some loss occur after projecting to a lower dimensionality.

raw data onto with the minimal loss. We then later determine the classifier on this plane that could classify the projected data.

Project the data using the projection matrix we just found $\mathbf{W}_opt$ by simply multiply the feature vectors, the train dataset and the test dataset, with it $\mathbf{x}_{proj} = \mathbf{W}_{opt}^T\,\mathbf{x}_{feature}$. Later pass the projected feature vectors to the classifier for predicting the labels for the test data.

Here I use K-nearest-neighbor(KNN) for classifying my test data. The concept of KNN is as what the name suggests, find the k nearest neighbor of the coming test data points. To begin with, I calculate the distance of all the data points to the test data in case there might be multiple points that have the same distance. Sort the distance in ascended order and extract the k ones with the lowest value of distance, which simply means the nearest. Then I count the frequency of the extracted k points to see which label appears the most. Assign the test data to the class of the most common label.

3.2 Result

Comparing Table (7) with Table (1), we can see some obvious observation. The accuracy of 'wine' dataset decreases but increase in the other two datasets. This might occur because the 'wine' dataset is a relatively small dataset, where the projection loss plays a significant role on classification. As for other two datasets with large amount of data, projection loss is not as affective.

$$\begin{pmatrix}
30 & 19 & 2 & 12 & 0 & 12 & 3 & 2 & 0 & 12 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 27 & 3 & 15 & 0 & 9 & 2 & 1 & 0 & 21 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 3 & 69 & 2 & 0 & 3 & 17 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
24 & 13 & 2 & 44 & 0 & 0 & 2 & 7 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 94 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 11 & 2 & 2 & 0 & 54 & 1 & 0 & 0 & 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 22 & 5 & 0 & 7 & 70 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 4 & 0 & 10 & 0 & 0 & 2 & 64 & 0 & 7 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 79 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
13 & 19 & 0 & 9 & 0 & 7 & 1 & 0 & 0 & 37 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 0 & 1 & 0 & 8 & 2 & 0 & 0 & 16 & 75 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 2 & 0 & 94 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 96 & 0 & 5 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 100 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 93 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 2 & 92 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 91
\end{pmatrix}$$

Table 9: Confusion matrix of 'wallpaper' dataset using Fisher discriminant

$$\begin{pmatrix}
0.3226 & 0.2043 & 0.02151 & 0.129 & 0 & 0.129 & 0.03226 & 0.02151 & 0 & 0.129 & 0.01075 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.2157 & 0.2647 & 0.02941 & 0.1471 & 0 & 0.08824 & 0.01961 & 0.009804 & 0 & 0.2059 & 0.01961 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.02062 & 0.03093 & 0.7113 & 0.02062 & 0 & 0.03093 & 0.1753 & 0 & 0.01031 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.2449 & 0.1327 & 0.02041 & 0.449 & 0 & 0 & 0.02041 & 0.07143 & 0 & 0.04082 & 0.02041 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.8952 & 0 & 0 & 0 & 0.1048 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.07317 & 0.1341 & 0.02439 & 0.02439 & 0 & 0.6585 & 0.0122 & 0 & 0 & 0.0122 & 0.06098 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.01835 & 0.2018 & 0.04587 & 0 & 0.06422 & 0.6422 & 0.02752 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.01064 & 0.04255 & 0 & 0.1064 & 0 & 0 & 0.02128 & 0.6809 & 0 & 0.07447 & 0 & 0.06383 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0119 & 0 & 0 & 0 & 0.9405 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04762 \\
0.1287 & 0.1881 & 0 & 0.08911 & 0 & 0.06931 & 0.009901 & 0 & 0 & 0.3663 & 0.1485 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.01887 & 0.01887 & 0 & 0.009434 & 0 & 0.07547 & 0.01887 & 0 & 0 & 0.1509 & 0.7075 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1933 & 0 & 0.01681 & 0 & 0.7899 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.04673 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8972 & 0 & 0.04673 & 0.009346 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01961 & 0.9804 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04902 & 0 & 0 & 0 & 0.01961 & 0 & 0.9118 & 0.01961 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02941 & 0 & 0 & 0 & 0 & 0 & 0.01961 & 0.902 & 0.04902 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01031 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05155 & 0.9381
\end{pmatrix}$$

Table 10: Classification matrix of 'Wallpaper' dataset

$$\begin{pmatrix} 430 & 1 & 10 & 0 & 0 & 0 & 0 & 0 \\ 16 & 368 & 0 & 0 & 0 & 0 & 0 & 0 \\ 29 & 0 & 728 & 0 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 369 & 0 & 0 & 0 & 0 \\ 48 & 0 & 0 & 0 & 369 & 0 & 0 & 0 \\ 29 & 0 & 0 & 0 & 0 & 738 & 0 & 0 \\ 15 & 0 & 0 & 0 & 0 & 0 & 369 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 369 \end{pmatrix}$$

Table 11: Confusion matrix of 'Taiji' dataset

$$\begin{pmatrix} 0.9751 & 0.002268 & 0.02268 & 0 & 0 & 0 & 0 & 0 \\ 0.04167 & 0.9583 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.03831 & 0 & 0.9617 & 0 & 0 & 0 & 0 & 0 \\ 0.04651 & 0 & 0 & 0.9535 & 0 & 0 & 0 & 0 \\ 0.1151 & 0 & 0 & 0 & 0.8849 & 0 & 0 & 0 \\ 0.03781 & 0 & 0 & 0 & 0 & 0.9622 & 0 & 0 \\ 0.03906 & 0 & 0 & 0 & 0 & 0 & 0.9609 & 0 \\ 0.04651 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9535 \end{pmatrix}$$

Table 12: Classification matrix of 'Taiji' dataset

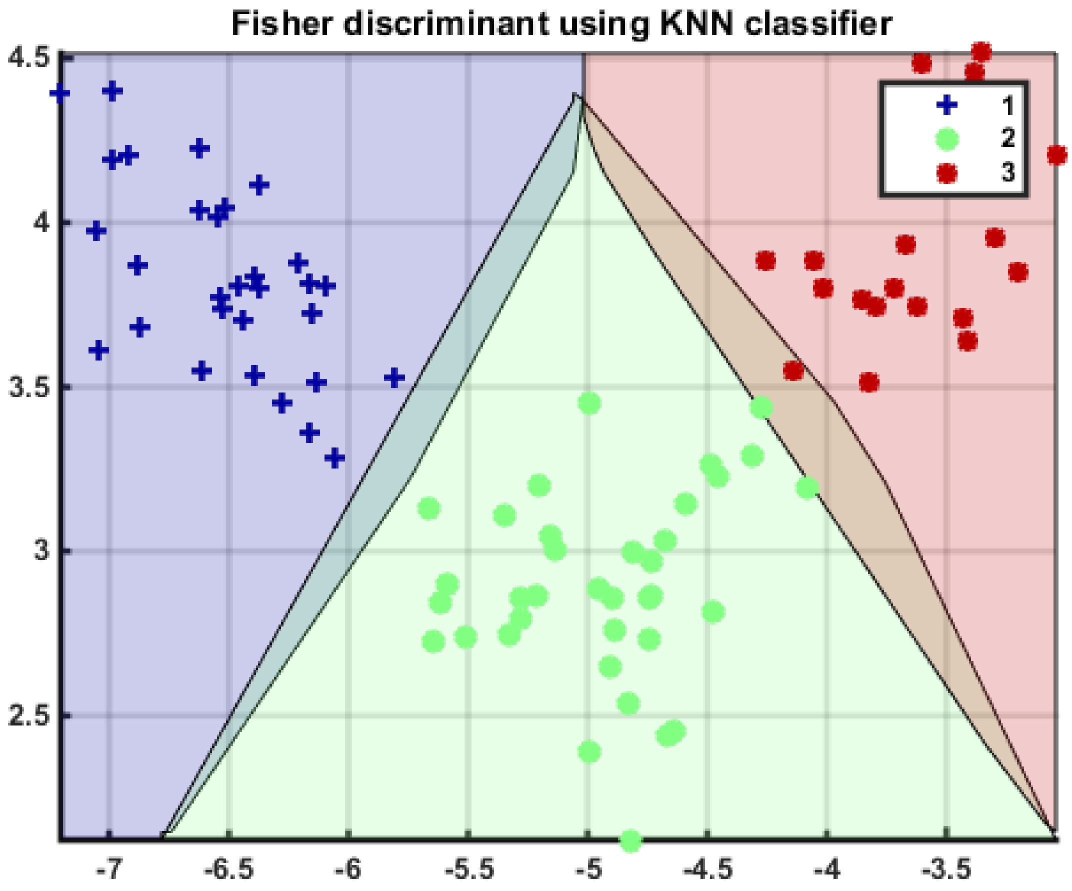


Figure 2: **Classifier of Fisher's discriminant of 'wine' dataset**

4 Conclusion

In this project, we implement two approaches to classify supervised data(data already has label). The first approach, discussed in section 2.1, is the least square approach. It uses the training data to calculate the weight function $\mathbf{W}$ that used to classified the data. The way it trains the model is by minimizing the error function (equation (3)) with respect to the $\mathbf{W}$ to get the $\mathbf{W}_{opt}$ (equation (6)). However, there is a problem to this approach. It is prone to the outliers. A cluster of data of the same class could lead to changes in the decision boundaries. Fisher projection is able to deal with this situation.

Fisher projection is a way to pre-process the data by projecting onto a lower dimensional hyperplane. The principle of projection is to separate different classes while keep the data of same class closer. This indicates that we want the largest of the inter-class covariance and the smallest the intra-class covariance. Under this principle, we get the optimal projection matrix $\mathbf{W}$ (equation (19)). We then calculate the decision boundaries on the projected hyperplane, either with KNN or the decision theory. We could predict the label of the test data using this decision boundaries. It turns out the Fisher discriminant isn't affected by the outliers so much. Also, since the data has been projected onto a lower dimension, the curse of dimensionality could be less likely to happen.

5 Extra Problem

In the extra problem, I picked the first two classes of 'wine' dataset to be my special case. The sum-of-square error function is

$$E = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n + w_0 - t_n)^2 \tag{19}$$

We want to find the $\mathbf{w}$ and w_0 that minimize the error function. Therefore, we differentiate the error function with respect to $\mathbf{w}$ and w_0 to get

$$\sum_{n=1}^{N}\left(\mathbf{w}^T\mathbf{x}_n + w_0 - t_n\right) = 0 \tag{20}$$

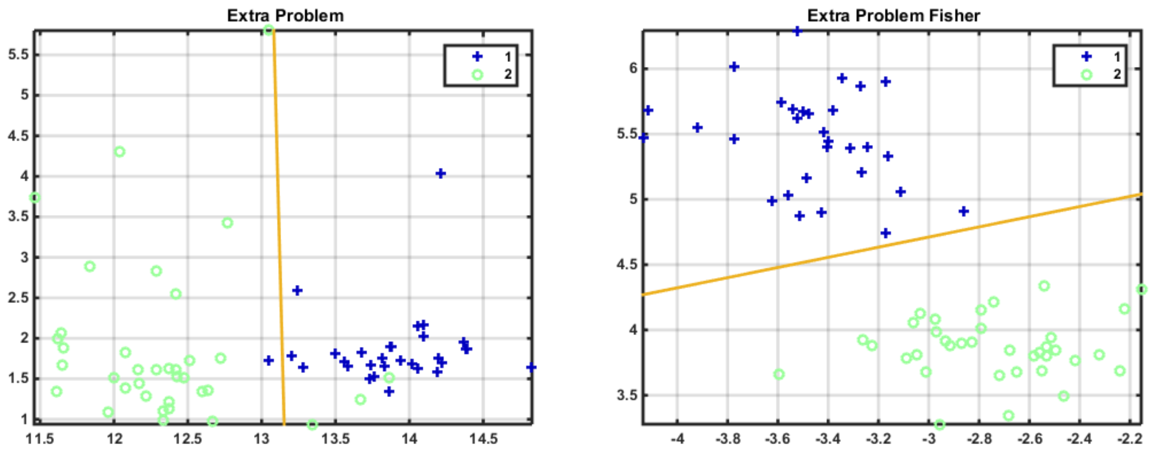
$$\sum_{n=1}^{N}\left(\mathbf{w}^T\mathbf{x}_n + w_0 - t_n\right)\mathbf{x}_n = 0 \tag{21}$$

Equation (20) would become

$$\sum_{n=1}^{N} w_0 = -\sum_{n=1}^{N}\mathbf{w}^T\mathbf{x}_n \tag{22}$$

$$\Rightarrow Nw_0 = -\mathbf{w}^T\sum_{n=1}^{N}\mathbf{x}_n \tag{23}$$

$$\Rightarrow w_0 = -\mathbf{w}^T\mathbf{m} \tag{24}$$



(a) Least square with two classes (b) Fisher projection with two classes

Figure 3: As the figures show, the two figures are basically identical except for the rotational difference.

where we set t_n to be

$$\sum_{n=1}^{N} t_n = N_1\frac{N}{N_1} - N_2\frac{N}{N2} = 0 \tag{25}$$

and $\mathbf{m}$ is the mean of the total dataset

$$\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{26}$$

Equation (21) would yield to

$$(\mathbf{S}_W + \frac{N_1N_2}{N}\mathbf{S}_B)\mathbf{w}) = N(\mathbf{m}_1 - \gtrdot_2) \tag{27}$$

Note that $\mathbf{S}_B\mathbf{w}$ always have the same direction as $(\mathbf{m}_1 - \mathbf{m}_2)$. Thus, equation (27) would become

$$\mathbf{w} \ \propto \ \mathbb{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \tag{28}$$

This actually has the same result as the special case of two classes of Fisher projection, which can be obtained by finding the optimal solution of the Fisher criterion

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \tag{29}$$

This yields to

$$\mathbf{w} \ \propto \ \mathbb{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \tag{30}$$

It is obvious that the two-class Fisher projection is the special case of least squares approach by setting the target matrix of the least squares, $\mathbf{T}$ for class C_1 to be N/N_1 and for class C_2 to be $-N/N_2$.