Machine Learning – Bayes Decision Rule vs Fisher Discriminant Analysis

Tan Jian An

Introduction

Given a 3-class pattern classification problem, two classifiers, namely Bayes decision rule and Fisher discriminant analysis, are trained using the training data, and the class labels of the testing data are predicted. The data has four variables, and they are classified into three different classes.

Splitting the data

When a classifier is trained too well on a training data to the extent that it fits even the noise or outliers in the data, overfitting may occur. Therefore, the trained classifier may not be able to generalize well on new, unseen data. To mitigate this issue, the training data is split into training and validation sets. The training set will be used to update the weights of the classifier, while the validation set is used to evaluate the performance of the classifier. By separating the validation set out and not use it to train the classifier, a more accurate estimate of the general performance of the classifier on new, unseen data can be obtained, and thus prevent overfitting.

In this experiment, the ratio between train data and validation data is 7:3.

```
% Test-train split
rng('default');
% Cross validation (train: 70%, test: 30%)
cv = cvpartition(size(Data_Train,1),'HoldOut',0.3);
idx = cv.test;
% Separate to training and test data
dataTrain = Data_Train(~idx,:);
labelTrain = Label_Train(~idx, :);
dataVal  = Data_Train(idx,:);
labelVal = Label_Train(idx, :);
dataTest = Data_test;
```

After splitting the indices of the samples of train data and validation data are as follow:

Train data

Table 1 Index of train data

2	17	31	46	62	76	90	103	116
3	19	32	47	63	77	92	104	117
4	21	34	48	66	79	94	106	118
5	23	35	50	67	80	95	107	119
10	24	36	53	69	81	96	109	
11	25	39	54	70	82	97	110	
12	26	41	55	71	83	98	112	
13	28	42	56	73	85	100	113	
15	29	43	57	74	87	101	114	
16	30	45	61	75	89	102	115	

Validation data

Table 2 Index of validation data

1	33	60	93
6	37	64	99
7	38	65	105
8	40	68	108
9	44	72	111
14	49	78	120
18	51	84	
20	52	86	
22	58	88	
27	59	91	

Bayes decision rule

Estimated mean and covariance matrix

Firstly, the parameters of the multivariate normal probability density function will be estimated. The estimated mean vector $\hat{\boldsymbol{\mu}}$ and estimated covariance matrix $\hat{\boldsymbol{\Sigma}}$ will be approximated through the maximum likelihood parameter estimation. For instance, the estimated mean vector of the train data labelled class 1 can be computed as:

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{n_1}\sum_{\mathbf{x}_k \in D_1} \mathbf{x}_k$$

Where n_1 is the total number of data labelled as class 1 in the train data; $\mathbf{x}_k \in D_1$ are the data in the train data belonging to class 1.

The covariance matrix of the train data labelled class 1 can be estimated through:

$$\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{n_1}\sum_{\mathbf{x}_k \in D_1} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_1)\,(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_1)^T$$

Similarly, the estimated mean and covariance matrix of the train data are listed as follow:

Table 3 Estimated mean and covariance matrix of each class

Train data	Estimated mean	Estimated covariance matrix
Class 1	$\hat{\boldsymbol{\mu}}_1 = \frac{1}{n_1}\sum_{\mathbf{x}_k \in D_1} \mathbf{x}_k = \begin{bmatrix} 4.9400 \\ 3.3767 \\ 1.4467 \\ 0.2500 \end{bmatrix}$	$\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{n_1}\sum_{\mathbf{x}_k \in D_1} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_1)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_1)^T = \begin{bmatrix} 0.1131 & 0.0959 & 0.0301 & 0.0150 \\ 0.0959 & 0.1618 & 0.0204 & 0.0085 \\ 0.0301 & 0.0204 & 0.0385 & 0.0067 \\ 0.0150 & 0.0085 & 0.0067 & 0.0105 \end{bmatrix}$
Class 2	$\hat{\boldsymbol{\mu}}_2 = \frac{1}{n_2}\sum_{\mathbf{x}_k \in D_2} \mathbf{x}_k = \begin{bmatrix} 6.5308 \\ 2.9346 \\ 5.5615 \\ 1.9808 \end{bmatrix}$	$\hat{\boldsymbol{\Sigma}}_2 = \frac{1}{n_2}\sum_{\mathbf{x}_k \in D_2} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_2)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_2)^T$

		$= \begin{bmatrix} 0.4429 & 0.0755 & 0.3473 & 0.0721 \\ 0.0755 & 0.0692 & 0.0417 & 0.0380 \\ 0.3473 & 0.0417 & 0.3262 & 0.0631 \\ 0.0721 & 0.0380 & 0.0631 & 0.0785 \end{bmatrix}$
Class 3	$\widehat{\boldsymbol{\mu}}_3 = \frac{1}{n_3}\sum_{\mathbf{x}_k \in D_3} \mathbf{x}_k = \begin{bmatrix} 5.9393 \\ 2.7786 \\ 4.2857 \\ 1.3321 \end{bmatrix}$	$\widehat{\boldsymbol{\Sigma}}_3 = \frac{1}{n_3}\sum_{\mathbf{x}_k \in D_3} (\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_3)(\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_3)^T$ $= \begin{bmatrix} 0.2674 & 0.0758 & 0.2063 & 0.0591 \\ 0.0758 & 0.0881 & 0.0868 & 0.0407 \\ 0.2063 & 0.0868 & 0.2412 & 0.0740 \\ 0.0591 & 0.0407 & 0.0740 & 0.0379 \end{bmatrix}$

Prior probability

The prior probability can be obtained as the ratio of the number of samples belonging to each class to the total number of samples in training data (84 after splitting):

$$p(\omega_1) = \frac{n_1}{n} = 0.3571$$

$$p(\omega_2) = \frac{n_2}{n} = 0.3095$$

$$p(\omega_3) = \frac{n_3}{n} = 0.3334$$

Class conditional probability

The class conditional probability densities can then be obtained as:

$$p(\mathbf{x}|\omega_1) = \frac{1}{\left((2\pi)^4 \left|\widehat{\boldsymbol{\Sigma}}_1\right|\right)^{\frac{1}{2}}} [-\frac{1}{2}(\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_1)^T \widehat{\boldsymbol{\Sigma}}_1^{-1} (\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_1)]$$

$$p(\mathbf{x}|\omega_2) = \frac{1}{\left((2\pi)^4 \left|\widehat{\boldsymbol{\Sigma}}_1\right|\right)^{\frac{1}{2}}} [-\frac{1}{2}(\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_2)^T \widehat{\boldsymbol{\Sigma}}_2^{-1} (\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_2)]$$

$$p(\mathbf{x}|\omega_3) = \frac{1}{\left((2\pi)^4 \left|\widehat{\boldsymbol{\Sigma}}_1\right|\right)^{\frac{1}{2}}} [-\frac{1}{2}(\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_3)^T \widehat{\boldsymbol{\Sigma}}_3^{-1} (\mathbf{x}_k - \widehat{\boldsymbol{\mu}}_3)]$$

Discriminant function

The discriminant function is then:

$$g_1(\mathbf{x}) = p(\mathbf{x}|\omega_1)p(\omega_1)$$

$$g_2(\mathbf{x}) = p(\mathbf{x}|\omega_2)p(\omega_2)$$

$$g_3(\mathbf{x}) = p(\mathbf{x}|\omega_3)p(\omega_3)$$

For any given sample $\mathbf{x}$,

1. The given sample is classified as class 1 if $g_1(\mathbf{x})$ has maximum value among the three discriminant functions.

$$\max\{g_1(\mathbf{x}), g_2(\mathbf{x}), g_3(\mathbf{x})\} = g_1(\mathbf{x})$$

2. The given sample is classified as class 2 if $g_2(\mathbf{x})$ has maximum value among the three discriminant functions.

$$\max\{g_1(\mathbf{x}), g_2(\mathbf{x}), g_3(\mathbf{x})\} = g_2(\mathbf{x})$$

3. The given sample is classified as class 1 if $g_3(\mathbf{x})$ has maximum value among the three discriminant functions.

$$\max\{g_1(\mathbf{x}), g_2(\mathbf{x}), g_3(\mathbf{x})\} = g_3(\mathbf{x})$$

Code snippet

```
load Data_test(1).mat;
load Data_Train(1).mat;
load Label_Train(1).mat;

data_class = [1, 2, 3];
data_size = size(Data_Train);

% Test-train split
rng('default');
% Cross validation (train: 70%, test: 30%)
cv = cvpartition(size(Data_Train,1),'HoldOut',0.3);
idx = cv.test;
% Separate to training and test data
dataTrain = Data_Train(~idx,:);
labelTrain = Label_Train(~idx, :);
dataVal  = Data_Train(idx,:);
labelVal = Label_Train(idx, :);
dataTest = Data_test;

data = dataTrain;
label = labelTrain;
num_class = length(data_class);

% train_class_idx identifies the data indices belonging to each class
% train_index shows the total number of each class in the data
[train_class_idx, train_index] = class_loc_idx(label, data_class);

prior_prob = prior_probability(train_index);
[class_mean, class_covariance] = find_class_mean(data, train_class_idx,
train_index);

val = predict(dataVal, num_class, class_mean, class_covariance, prior_prob);
```

```
validation_accuracy = score(val, labelVal);

test = predict(Data_test, num_class, class_mean, class_covariance, prior_prob);

function accuracy = score(predict, target)
    accuracy = 0;
    for i = 1:length(predict)
        if predict(i) == target(i)
            accuracy = accuracy + 1;
        end
    end
    accuracy = accuracy/length(predict);
end
function [class_idx, index] = class_loc_idx(labelData, dataClass)
    class_idx = zeros(length(dataClass), length(labelData));
    index = zeros([length(dataClass), 1]);
    for i = 1:length(labelData)
        for j = 1:length(dataClass)
            if labelData(i) == dataClass(j)
                index(j) = index(j) + 1;
                class_idx(j, index(j)) = i;
            end
        end
    end
end

function output = predict(data, num_class, class_mean, class_covariance,
prior_prob)
    discriminant_fn = find_df(data, num_class, class_mean, class_covariance,
prior_prob);
    output = zeros([length(discriminant_fn), 1]);
    for i = 1:length(output)
        largest_discriminant_fn = max(discriminant_fn(i, :));
        for j = 1:length(discriminant_fn(i, :))
            if discriminant_fn(i, j) == largest_discriminant_fn
                output(i) = j;
            end
        end
    end
end

function [class_mean, class_covariance] = find_class_mean(data, class_idx,
index)
    class_mean = zeros([size(data, 2),length(index)]);
    class_covariance = zeros([size(data, 2),(size(data, 2)*length(index))]);
    for i = 1:length(index)
        class = class_idx(i, 1:index(i));
        data_class = data(class, :);
        mean = find_mean(data_class);
        covariance = find_covariance(data_class, mean);
        class_mean(:, i) = mean;
        class_covariance(:, (i - 1)*size(data, 2)+1:(i - 1)*size(data, 2)+4) =
covariance;
    end
end

function mean_v = find_mean(data)
    mean_v = zeros([size(data, 2) 1]);
```

```
    for i = 1:length(mean_v)
        mean_v(i) = mean(data(:, i));
    end
end

function covariance_m = find_covariance(data, mean_v)
    covariance_m = zeros([size(data, 2) size(data, 2)]);
    for j = 1:length(data)
        covariance_m = covariance_m + (data(j, :)' - mean_v) * (data(j, :)' -
mean_v)';
    end
    covariance_m = covariance_m/length(data);
end

function discriminant_fn = find_df(data, num_class, class_mean,
class_covariance, prior_prob)
    discriminant_fn = zeros([size(data, 1) num_class]);
    for i = 1:length(discriminant_fn)
        for j = 1:num_class
            mean = class_mean(:, j);
            covariance = class_covariance(:, (j - 1)*size(data, 2)+1:(j -
1)*size(data, 2)+4);
            discriminant_fn(i, j) = class_con_prob(data(i, :), mean,
covariance) ...
                * prior_prob(j);
        end
    end
end

function prior_prob = prior_probability(data_index)
    prior_prob = zeros([length(data_index), 1]);
    sum = 0;
    for i = 1:length(data_index)
        sum = sum + data_index(i);
    end
    for i = 1:length(data_index)
        prior_prob(i) = data_index(i)/sum;
    end

end

function class_con_prob = class_con_prob(x, mean_v, covariance_m)
    class_con_prob = (1/((2*pi)^(length(mean_v))*det(covariance_m)^(1/2))) ...
        * exp(-0.5 * ((x' - mean_v)' * inv(covariance_m) * (x' - mean_v)));
end
```

Validation result

The Validation result can be obtained as follow:

Table 4 Bayes decision rule validation accuracy

No	Validation Label	Bayes Decision Rule Prediction		No	Validation Label	Bayes Decision Rule Prediction
1	2	2		58	3	3
6	1	1		59	2	3
7	2	2		60	1	1
8	3	3		64	2	2
9	3	3		65	1	1
14	1	1		68	2	2
18	2	2		72	3	3
20	2	2		78	2	2
22	1	1		84	3	3
27	1	1		86	3	3
33	2	2		88	2	2
37	2	2		91	2	2
38	2	2		93	2	2
40	3	3		99	1	1
44	3	3		105	1	1
49	1	1		108	3	3
51	3	3		111	3	3
52	3	3		120	1	1
					Accuracy	**0.972222**

Fisher discriminant analysis

For a 3-class problem. The natural generalisation of the Fisher's linear discriminant involves 2 discriminant functions. Hence, the 4-dimensional space of the dataset is projected onto the 2-dimensional space. The generalisation of the within-class scatter matrix $\boldsymbol{S}_W$ is given as follow:

$$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$$
$$\mathbf{S}_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$
$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$

Where c refers to the total number of classes, i.e., 3; $\mathbf{m}_i$ is the mean vector of class i.

The total mean vector can be defined as:

$$m = \frac{1}{n} \sum \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$$

Total scatter matrix is:

$$\begin{aligned}
\mathbf{S}_T &= \sum_{c} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T \\
&= \sum_{i=1} \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i + \mathbf{m}_i - \mathbf{m})(\mathbf{x} - \mathbf{m}_i + \mathbf{m}_i - \mathbf{m})^T \\
&= \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T + \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T \\
&= \mathbf{S}_W + \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T
\end{aligned}$$

The second term refers to the between-class scatter matrix, $\boldsymbol{S}_B$

$$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

The data is projected from the 4-dimensional space to a 2 dimensional space by 2 discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x}$$
$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \end{bmatrix}$$
$$\mathbf{W}^T = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \end{bmatrix}$$
$$\mathbf{g}(\mathbf{x}) = \mathbf{W}^T \mathbf{x}$$

The projected class mean vector and projected mean vector are as follow:

$$\tilde{\mathbf{m}}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{g}(\mathbf{x})$$

$$\tilde{\mathbf{m}} = \frac{1}{n} \sum_{i=1}^{c} n_i \tilde{\mathbf{m}}_i$$

The within-class and between-class scatter matrix of the projected data are therefore:

$$\begin{aligned} \tilde{\mathbf{S}}_W &= \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} (\mathbf{g}(\mathbf{x}_\mathrm{i}) - \tilde{\mathbf{m}}_{\,i})(\mathbf{g}(\mathbf{x}_\mathrm{i}) - \tilde{\mathbf{m}}_{\,i})^T \\ &= \mathbf{W}^{\boldsymbol{T}} \mathbf{S}_{\boldsymbol{W}} \mathbf{W} \\ \tilde{\mathbf{S}}_B &= \sum_{i=1}^{c} n_i (\tilde{\mathbf{m}}_i - \tilde{\mathbf{m}})(\tilde{\mathbf{m}}_i - \tilde{\mathbf{m}})^T \\ &= \mathbf{W}^{\boldsymbol{T}} \mathbf{S}_{\boldsymbol{B}} \mathbf{W} \end{aligned}$$

The objective is to find the weight matrix $\mathbf{W}$ such that the between-class scatter is maximised while the within-class scatter is minimised. The criterion function is given by:

$$J = \frac{|\tilde{\mathbf{S}}_B|}{|\tilde{\mathbf{S}}_W|} = \frac{|\mathbf{W}^{\boldsymbol{T}} \mathbf{S}_{\boldsymbol{B}} \mathbf{W}|}{|\mathbf{W}^{\boldsymbol{T}} \mathbf{S}_{\boldsymbol{W}} \mathbf{W}|}$$

The columns of the weight matrix $\mathbf{W}$ are the generalised eigenvectors corresponding to the largest eigenvalues of $\mathbf{S}_B$ and $\mathbf{S}_W$:

$$\begin{aligned} \mathbf{S}_B \mathbf{w}_i &= \lambda_i \mathbf{S}_W \mathbf{w}_i \\ \mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w}_i &= \lambda_i \mathbf{w}_i \\ [\boldsymbol{\Lambda}, \mathbf{W}] &= eig(inv(\mathbf{S}_W), \mathbf{S}_B) \end{aligned}$$

Given the train data after the validation-train-split, the within-class and between-class scatter are calculated as:

$$\mathbf{S}_W = \begin{bmatrix} 22.3942 & 6.9639 & 15.7105 & 3.9800 \\ 6.9639 & 9.1197 & 4.1287 & 2.3816 \\ 15.7105 & 4.1287 & 16.3905 & 3.9136 \\ 3.9800 & 2.3816 & 3.9136 & 3.4165 \end{bmatrix}$$

$$\mathbf{S}_B = \begin{bmatrix} 36.5157 & -11.7065 & 95.6928 & 39.7046 \\ -11.7065 & 5.6027 & -32.1454 & -12.6995 \\ 95.6928 & -32.1454 & 251.9362 & 104.0264 \\ 39.7046 & -12.6995 & 104.0264 & 43.1725 \end{bmatrix}$$

The eigenvalues and eigenvectors obtained are:

$$\boldsymbol{\Lambda} = \begin{bmatrix} 33.0609 & 0 & 0 & 0 \\ 0 & 0.2086 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 0.3945 & -0.0889 & -0.8639 & -0.4468 \\ 0.2377 & 0.6582 & 0.1107 & 0.3945 \\ -0.6632 & -0.1808 & 0.1490 & 0.4845 \\ -0.5899 & 0.7254 & 0.4682 & -0.6404 \end{bmatrix}$$

Therefore the 2 weight vectors obtained are:

$$\mathbf{w}_1 = \begin{bmatrix} 0.3945 \\ 0.2377 \\ -0.6632 \\ -0.5899 \end{bmatrix}$$

$$\mathbf{w}_2 = \begin{bmatrix} -0.0889 \\ 0.6582 \\ -0.1808 \\ 0.7254 \end{bmatrix}$$

The first column in $\mathbf{W}$ is selected as $\mathbf{w}_1$ as it corresponds to larger eigenvalue.

Projections of samples on $\mathbf{w}_1$

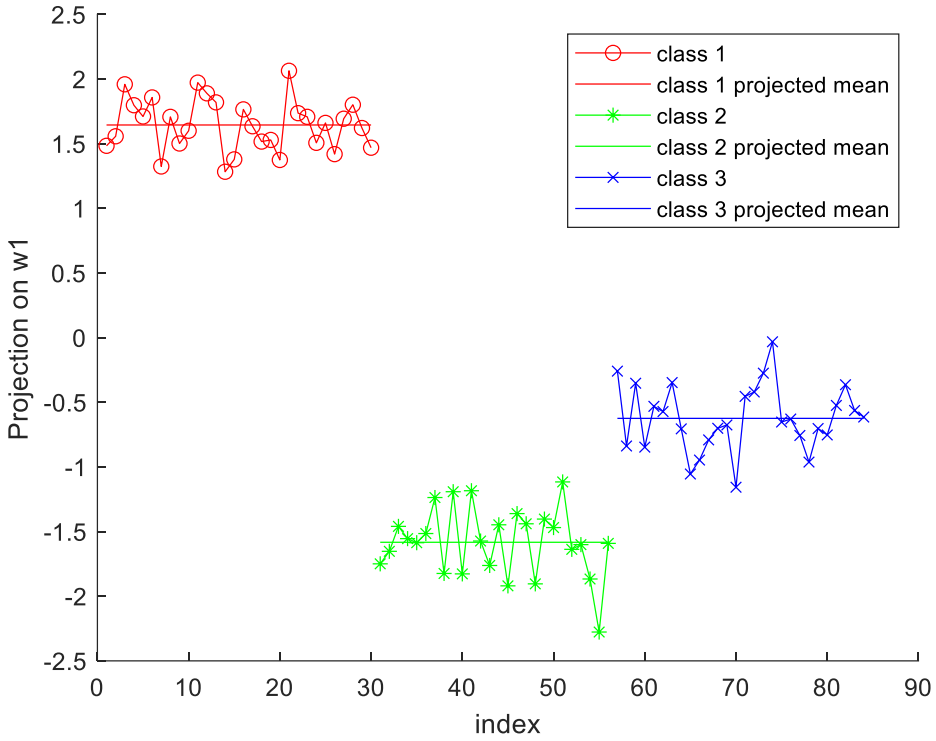

Figure 1 Projections of samples on w1

For the samples projection on $\mathbf{w}_1$,

$$\tilde{m}_1 = 1.6447$$
$$\tilde{m}_2 = -1.5827$$
$$\tilde{m}_3 = -0.6244$$

Samples in class 1 are well separated from the other classes. Moreover, the samples in class 3 are closer to that of class 1. Therefore, the weight biases w_{01} can be set as:

$$w_{01} = -\frac{\tilde{m}_1 + \tilde{m}_3}{2} = -0.5101$$

Projections of samples on $\mathbf{w}_2$

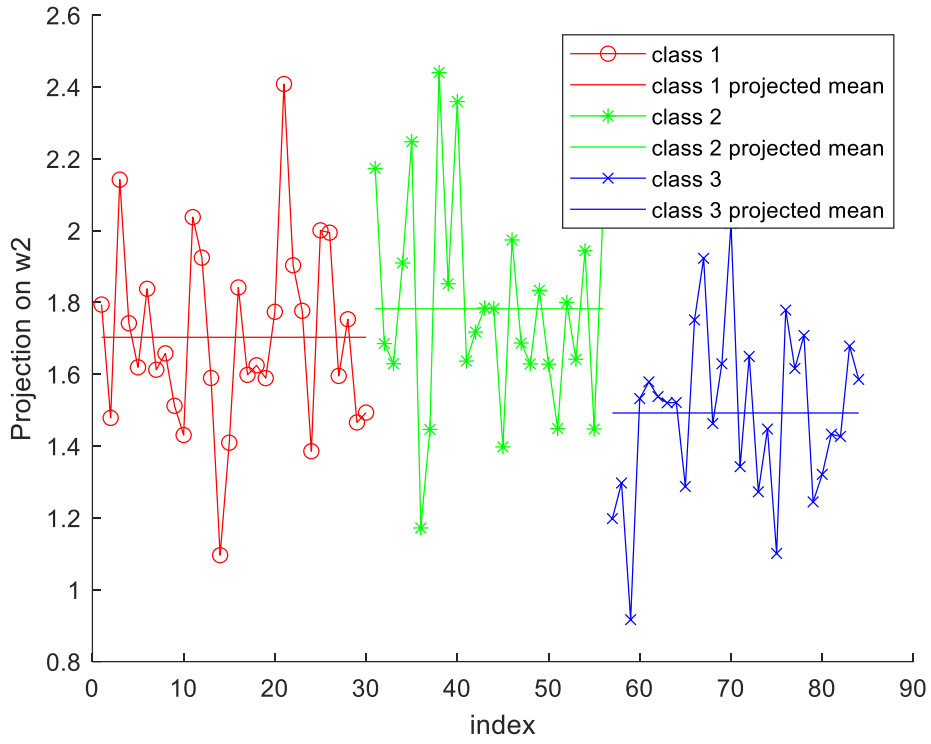

Figure 2 Projections of samples on w2

For the samples projection on $\mathbf{w}_2$,

$$\tilde{m}_1 = 1.7031$$
$$\tilde{m}_2 = 1.7823$$
$$\tilde{m}_3 = 1.4924$$

Samples in class 3 are separated from the other classes. Moreover, the samples in class 1 are closer to that of class 3. Therefore, the weight biases w_{01} can be set as:

$$w_{02} = -\frac{\tilde{m}_1 + \tilde{m}_3}{2} = -1.5977$$

Hence the 2 discriminant functions can be written as follow:

$$g_1(\mathbf{x}) = \begin{bmatrix} 0.3945 \\ 0.2377 \\ -0.6632 \\ -0.5899 \end{bmatrix}^T \mathbf{x} - 0.5101$$

$$g_2(\mathbf{x}) = \begin{bmatrix} -0.0889 \\ 0.6582 \\ -0.1808 \\ 0.7254 \end{bmatrix}^T \mathbf{x} - 1.5977$$

The decision rule can be written as:

1. If $g_1(\mathbf{x}) > 0$ and $g_2(\mathbf{x}) > 0$, decide ω_1
2. If $g_1(\mathbf{x}) < 0$ and $g_2(\mathbf{x}) < 0$, decide ω_3
3. If $g_1(\mathbf{x}) < 0$ and $g_2(\mathbf{x}) > 0$, decide ω_2
4. If $g_1(\mathbf{x}) > 0$ and $g_2(\mathbf{x}) < 0$, decide ω_1 if $g_1(\mathbf{x}) > g_2(\mathbf{x})$; else decide w_3 if $g_1(\mathbf{x}) < g_2(\mathbf{x})$

Code snippet

```
load Data_test(1).mat;
load Data_Train(1).mat;
load Label_Train(1).mat;

data_class = [1, 2, 3];
data_size = size(Data_Train);
plot_marker = {'-ro', '-g*', '-bx'};
% Test-train split
rng('default');
% Cross validation (train: 70%, test: 30%)
cv = cvpartition(size(Data_Train,1),'HoldOut',0.3);
idx = cv.test;
% Separate to training and test data
dataTrain = Data_Train(~idx,:);
labelTrain = Label_Train(~idx, :);
dataVal  = Data_Train(idx,:);
labelVal = Label_Train(idx, :);
dataTest = Data_test;

data = dataTrain;
label = labelTrain;
num_class = length(data_class);

% train_class_idx identifies the data indices belonging to each class
% train_index shows the total number of each class in the data
[train_class_idx, train_index] = class_loc_idx(label, data_class);
class_mean = find_class_mean(data, train_class_idx, train_index);
total_mean = find_mean(data);
s_w = scatter_within(class_mean, data, train_class_idx, train_index);
s_b = scatter_between(class_mean, total_mean, train_index);
w = weights(s_w, s_b, num_class);

projected_mean_1 = projection(w, 1, data, train_class_idx, train_index,
plot_marker);
w0_1 = -0.5 * (projected_mean_1(1) + projected_mean_1(3));
w_1 = w(:, 1);
projected_mean_2 = projection(w, 2, data, train_class_idx, train_index,
plot_marker);
w0_2 = -0.5 * (projected_mean_2(1) + projected_mean_2(3));
w_2 = w(:, 2);

val = find_df(dataVal, w_1, w0_1, w_2, w0_2);
validation_accuracy = score(val, labelVal)

test = find_df(dataTest, w_1, w0_1, w_2, w0_2);
% figure
% g1 = zeros([length(data), 1]);
% idx = zeros([length(data), 1]);
% for i = 1:length(data)
%     idx(i) = i;
%     g1(i) = w(:, 1)'*data(i, :)';
% end
%
% class1 = train_class_idx(1, 1:train_index(1));
% class2 = train_class_idx(2, 1:train_index(2));
% class3 = train_class_idx(3, 1:train_index(3));
%
% plot(idx(1:length(class1)), g1(class1),'-o');
```

```
% hold on;
% plot(idx(length(class1) + 1:length(class1) + length(class2)), g1(class2),'-
*');
% plot(idx(length(class1) + length(class2) + 1:length(class1) + length(class2) +
length(class3)), g1(class3),'-+');

function accuracy = score(predict, target)
    accuracy = 0;
    for i = 1:length(predict)
        if predict(i) == target(i)
            accuracy = accuracy + 1;
        end
    end
    accuracy = accuracy/length(predict);
end

function output = find_df(data, w_1, w0_1, w_2, w0_2)
    output = zeros([length(data), 1]);
    for i = 1:length(data)
        data_i = data(i, :)';
        g1 = w_1'*data_i + w0_1;
        g2 = w_2'*data_i + w0_2;

        if g1 > 0 && g2 > 0
            output(i) = 1;
        elseif g1 < 0 && g2 < 0
            output(i) = 3;
        elseif g1 < 0 && g2 > 0
            output(i) = 2;
        else
            if g1 > g2
                output(i) = 1;
            elseif g1 < g2
                output(i) = 3;
            end
        end
    end
end

function projected_mean = projection(weights, weight_idx, data, class_idx,
index, plot_marker)
    wTx = zeros([length(data), 1]);
    idx = zeros([length(data), 1]);
    projected_mean = zeros([length(index), 1]);
    for i = 1:length(data)
        idx(i) = i;
        wTx(i) = weights(:, weight_idx)'*data(i, :)';
    end
    figure;
    hold on;
    for i = 1:length(index)
        class = class_idx(i, 1:index(i));
        wTx_i = wTx(class);
        projected_mean(i) = mean(wTx_i);

        if i == 1
            idx_i = idx(1:index(i));
        else
            idx_cumulative = 0;
```

```
            for j = 1:(i - 1)
                idx_cumulative = idx_cumulative + index(j);
            end
            idx_i = idx(idx_cumulative + 1: idx_cumulative + index(i));
        end
        mean_y = ones([length(idx_i), 1])*projected_mean(i);

        plot(idx_i, wTx_i, plot_marker{i}, 'DisplayName',['class ',
int2str(i)]);
        plot(idx_i, mean_y, plot_marker{i}(1:2), 'DisplayName',['class ',
int2str(i), ' projected mean']);
    end
    legend;
    xlabel('index');
    ylabel(['Projection on w', int2str(weight_idx)]);
    hold off;
end

function w = weights(s_w, s_b, num_class)
    [v, d] = eig(inv(s_w)*s_b)
    w = v(:, 1:num_class - 1);

end

function [class_idx, index] = class_loc_idx(labelData, dataClass)
    class_idx = zeros(length(dataClass), length(labelData));
    index = zeros([length(dataClass), 1]);
    for i = 1:length(labelData)
        for j = 1:length(dataClass)
            if labelData(i) == dataClass(j)
                index(j) = index(j) + 1;
                class_idx(j, index(j)) = i;
            end
        end
    end
end

function class_mean = find_class_mean(data, class_idx, index)
    class_mean = zeros([size(data, 2),length(index)]);
    for i = 1:length(index)
        class = class_idx(i, 1:index(i));
        data_class = data(class, :);
        mean = find_mean(data_class);
        class_mean(:, i) = mean;
    end
end

function mean_v = find_mean(data)
    mean_v = zeros([size(data, 2) 1]);
    for i = 1:length(mean_v)
        mean_v(i) = mean(data(:, i));
    end
end

function s_w = scatter_within(class_mean, data, class_idx, index)
    s_w = zeros([size(data, 2),size(data, 2)]);
    for i = 1:length(index)
        mean = class_mean(:, i);
        class = class_idx(i, 1:index(i));
```

```
        data_class = data(class, :);
        s_i = zeros([size(data_class, 2),size(data_class, 2)]);
        for j = 1:size(data_class, 1)
            x = data_class(j, :)';
            s_i = s_i + ((x - mean)*(x - mean)');
        end
        s_w = s_w + s_i;
    end
end

function s_b = scatter_between(class_mean, mean, index)
    s_b = zeros([length(mean), length(mean)]);
    for i = 1:length(index)
        n_i = index(i);
        class_mean_i = class_mean(:, i);
        s_b = s_b + (n_i * ((class_mean_i - mean)*(class_mean_i - mean)'));
    end
end
```

Validation result

The Validation result can be obtained as follow:

Table 5 Fisher linear discriminant analysis validation accuracy

No	Validation Label	Fisher Linear Discriminant		No	Validation Label	Fisher Linear Discriminant
1	2	2		58	2	3
6	1	1		59	3	3
7	3	2		60	1	1
8	2	3		64	2	2
9	2	3		65	1	1
14	1	1		68	2	2
18	2	2		72	3	3
20	3	2		78	2	2
22	1	1		84	2	3
27	1	1		86	2	3
33	2	2		88	2	2
37	3	2		91	2	2
38	2	2		93	2	2
40	3	3		99	1	1
44	2	3		105	1	1
49	1	1		108	3	3
51	3	3		111	3	3
52	3	3		120	1	1
					Accuracy	**0.75**

Discussion

From the experiment, the Bayes decision rule has better performance in predicting the data class of the samples in the validation set. Therefore, it can be concluded that the dataset follows a Gaussian normal distribution. Therefore, when normal distribution is used in density estimation, the performance of the classifier is accurate.

For Fisher linear discriminant analysis, whilst its performance in predicting samples in class 1 is good, its performance deteriorates when it comes to class 2 and class 3. The experiment results show that for all the mistakes that Fisher linear discriminant analysis in predicting the data classes in the validation set come from either class 2 or class 3. Figure below shows that when the validation set is plotted in the decision boundary, 3 of the samples from class 2 and 6 of the samples from class 3 fell in the incorrect boundary area.

This could be explained from the fact that when the samples are projected on $\mathbf{w}_1$, the samples in class 1 are well separated from the other two classes. Thus, when the discriminant function $g_1(\mathbf{x})$ can distinguish class 1 and non-class 1 samples well. However, to differentiate between class 2 and class 3 samples, the classifier relies on $g_2(\mathbf{x})$. From the projection of samples onto $\mathbf{w}_2$, the difference between class means $\tilde{m}_2$ and $\tilde{m}_3$ are small, making it difficult to differentiate between the two classes. As a result, the accuracy of predicting class 2 and class 3 of the dataset may be compromised.

Moreover, as the boundary is linear, the accuracy of predicting the classes will be affected.

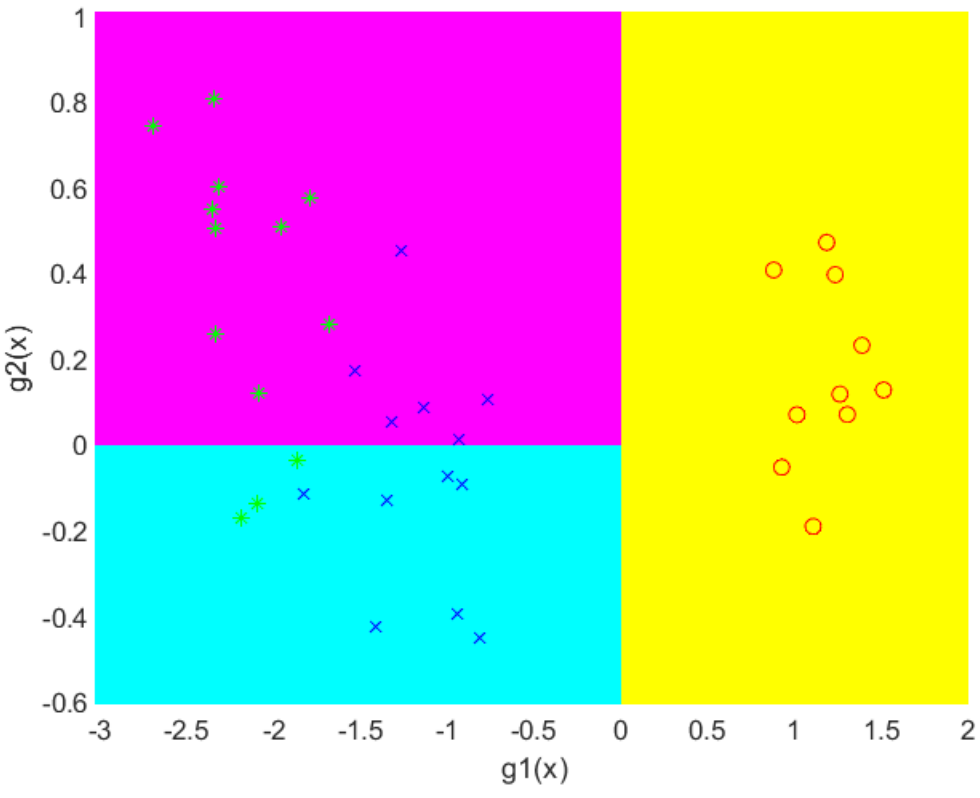

Figure 3 Fisher linear discriminant decision boundary

The performance of the Fisher discriminant analysis classifier can be improved by incorporating the Bayes decision rule for classification. The linear discriminant function is first being computed by maximising the ratio of the between-class scatter matrix to within-class scatter matrix. Then, the linear discriminant score of each class, which is the dot product of the linear discriminant function and the

sample variables. The posterior probability is then computed by multiplying the prior probability with the likelihood of the sample given class k, which is modeled as a multivariate Gaussian distribution with mean and covariance matrix estimated from the training data. The likelihood is calculated using the linear discriminant score. In MATLAB, this method is readily available and can be applied through the 'fitcdiscr' function.

```
Mdl = fitcdiscr(dataTrain,labelTrain);
val = predict(Mdl,dataVal);
test = predict(Mdl,dataTest);
```

Validation Result of the LDA incorporating Bayes decision rule

No	Validation Label	Prediction by 'fitcdiscr' function		No	Validation Label	Prediction by 'fitcdiscr' function
1	2	2		58	3	3
6	1	1		59	3	2
7	2	2		60	1	1
8	3	3		64	2	2
9	3	3		65	1	1
14	1	1		68	2	2
18	2	2		72	3	3
20	2	2		78	2	2
22	1	1		84	3	3
27	1	1		86	3	3
33	2	2		88	2	2
37	2	2		91	2	2
38	2	2		93	2	2
40	3	3		99	1	1
44	3	3		105	1	1
49	1	1		108	3	3
51	3	3		111	3	3
52	3	3		120	1	1
					Accuracy	**0.972222**

Results – Prediction of the test data using Bayes decision rule and Fisher linear discriminant analysis

The predictions for the test data are as follow:

Table 6 Prediction on test data by the two classifiers

No	Prediction by Bayes Decision Rule		No	Prediction by Fisher Linear Discriminant Analysis		No	Prediction by Fisher Linear Discriminant Analysis incorporating Bayes Decision Rule (MATLAB 'fitcdiscr' funciton)
1	1		1	1		1	1
2	1		2	1		2	1
3	1		3	1		3	1
4	1		4	1		4	1
5	1		5	1		5	1
6	1		6	1		6	1
7	1		7	1		7	1
8	1		8	1		8	1
9	1		9	1		9	1
10	1		10	1		10	1
11	2		11	3		11	2
12	2		12	2		12	2
13	2		13	2		13	2
14	2		14	2		14	2
15	2		15	2		15	2
16	2		16	2		16	2
17	2		17	2		17	2
18	2		18	3		18	2
19	2		19	2		19	2
20	2		20	2		20	2
21	2		21	2		21	2
22	3		22	3		22	3
23	3		23	3		23	3
24	3		24	2		24	3
25	3		25	2		25	3
26	3		26	3		26	3
27	3		27	3		27	3
28	3		28	3		28	3
29	3		29	3		29	3
30	3		30	2		30	3