Homework 3: Linear Discriminant Analysis and Bayesian Decision Rule

Spring 2020
(Due: Friday Feb 28, 2020)

Objective

The objective of this homework is twofold:

(a) Implementing an image classification algorithm, and gaining experience in working with Python functions that manipulate image files along the way;

(b) Understanding important theoretical properties of linear discriminant analysis using the Bayesian decision rule.

Important Concepts

Linear Discriminant Analysis

Recall from the lectures that for classification problems, there are several approaches to constructing decision boundaries for classifiers. In project 2, we studied one example of them, the linear least square. Linear least squares is a **discriminative method**. The Bayesian decision rule is a **generative method**. When framed under linear models, both methods can be derived from the linear discriminant analysis.

Suppose we have an input vector $\boldsymbol{x} \in \mathbb{R}^d$ and K classes with class labels $\{1, 2, ..., K\}$. The **discriminant function** g_j for the j-th class is a mapping which maps an input vector $\boldsymbol{x}$ to a number. We assign the class label $j \in \{1, 2, ..., K\}$ to $\boldsymbol{x}$ if

$$j = \underset{i \in \{1,2,...,K\}}{\operatorname{argmax}} \; g_i(\boldsymbol{x}) \tag{1}$$

This then gives us the hypothesis function $h : \mathbb{R}^d \to \{1, 2, ..., K\}$, which is defined as

$$h(\boldsymbol{x}) = \underset{i \in \{1,2,...,K\}}{\operatorname{argmax}} \; g_i(\boldsymbol{x}) \tag{2}$$

Notice that for two classes, (2) reduces to

$$h(\boldsymbol{x}) = \operatorname{sign}\Big(g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})\Big) = \operatorname{sign}\Big(g(\boldsymbol{x})\Big),$$

which is what we had in homework 2.

Bayesian Decision Rule

In this homework, the general form of the discriminant functions we shall adopt on is based on the **maximum a-posteriori** (MAP) estimation method from Bayesian statistics theory.

Let us examine this concept in the context of our classifier. Notationally, consider $\boldsymbol{X}$ with range $\mathbb{R}^d$ and the classification function Y as random variables. For convenience, denote $p_j(\boldsymbol{x})$ as the class-conditional density of $\boldsymbol{X}$ in class j, i.e., $p_j(\boldsymbol{x}) = p_{\boldsymbol{X}|Y}(\boldsymbol{x}|j)$, and denote π_j as the prior probability of class j, i.e., $\pi_j = p_Y(j)$, so $\sum_{j=1}^{K} \pi_j = 1$. From Bayes' theory, the **posterior distribution** of Y on $\boldsymbol{X}$ is

$$p_{Y|\boldsymbol{X}}(j|\boldsymbol{x}) = \frac{p_j(\boldsymbol{x})\pi_j}{p_{\boldsymbol{X}}(\boldsymbol{x})} \tag{3}$$

we can interpret it as the probability that the feature vector $\boldsymbol{x}$ is of class j. We define our discriminant functions as the log of the posterior:

$$\begin{aligned} g_j(\boldsymbol{x}) &= \log p_{Y|\boldsymbol{X}}(j|\boldsymbol{x}) \\ &= \log p_j(\boldsymbol{x}) + \log \pi_j \end{aligned} \tag{4}$$

for all j and $\boldsymbol{x}$. Therefore, by (2), the class label of every $\boldsymbol{x} \in \mathbb{R}^d$ is obtained using the MAP method.

The Gaussian Model

In this homework, we will focus on analyzing the behavior of the discriminant functions and the decision boundaries they generate when we model each class density p_j as Gaussian functions, mainly due to the analytical tractability of it.

From the lectures, recall that if

$$p_j(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big\} \tag{5}$$

where $|\boldsymbol{\Sigma}_j| = \det\boldsymbol{\Sigma}_j$, then (4) is written as

$$g_j(\boldsymbol{x}) = \frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_j| + \log(\pi_j) \tag{6}$$

In practice we do not know the mean $\boldsymbol{\mu}_j$, covariance matrix $\boldsymbol{\Sigma}_j$ and prior probability π_j for any class, and so we need to estimate them based on our training data. Let $\mathcal{D}_1 = \{(\boldsymbol{x}_n^{(1)}, y_n^{(1)})\}_{n=1}^{N_1}, ..., \mathcal{D}_K = \{(\boldsymbol{x}_n^{(K)}, y_n^{(K)})\}_{n=1}^{N_K}$ be the training datasets for the K classes. Then:

$$\widehat{\boldsymbol{\mu}}_j = \frac{1}{N_j}\sum_{n=1}^{N_j} \boldsymbol{x}_n^{(j)} \tag{7}$$

for the sample mean of class j,

$$\widehat{\boldsymbol{\Sigma}}_j = \frac{1}{N_j - 1}\sum_{n=1}^{N_j} (\boldsymbol{x}_n^{(j)} - \widehat{\boldsymbol{\mu}}_j)(\boldsymbol{x}_n^{(j)} - \widehat{\boldsymbol{\mu}}_j)^T \tag{8}$$

for the unbiased in-class sample covariance matrix of class j, and for simplicity we can choose

$$\widehat{\pi}_j = \frac{N_j}{\sum_{k=1}^{K} N_k} \tag{9}$$

for the prior probability of class j.

Exercise 1: Maximum Likelihood Estimation of Covariance Matrix

In lecture, we proved that the ML estimate of the mean vector is the empirical average of the training samples. In this exercise we want to show that the ML estimate of the covariance matrix is

$$\widehat{\Sigma}_{\text{MLE}} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T \tag{10}$$

where $\{\boldsymbol{x}_n\}_{n=1}^N \subset \mathbb{R}^d$ is a set of iid data sampled from a random variable $\boldsymbol{X}$, and the mean $\boldsymbol{\mu}$ is assumed to be known. We focus on the multi-dimensional Gaussian so that the likelihood function is

$$p(\boldsymbol{x} \,|\, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big\} \tag{11}$$

To make things simpler we assume that $\boldsymbol{\Sigma}$ and $\widehat{\boldsymbol{\Sigma}}_{\text{MLE}}$ are invertible in this exercise.

Tasks:

(a) We show that (10) is the maximum likelihood estimate of the likelihood function defined in (11).

(i) Prove the matrix identity

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = \text{tr}[\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T],$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, and the trace operator is defined as $\text{tr}[\boldsymbol{A}] = \sum_{i=1}^d [\boldsymbol{A}]_{i,i}$.

(ii) Show the likelihood function can be written in the form:

$$p(\boldsymbol{x}_1,...,\boldsymbol{x}_N|\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{Nd/2}}|\boldsymbol{\Sigma}^{-1}|^{N/2}\exp\Big\{ -\frac{1}{2}\text{tr}\Big[\boldsymbol{\Sigma}^{-1}\sum_{n=1}^N(\boldsymbol{x}_n-\boldsymbol{\mu})(\boldsymbol{x}_n-\boldsymbol{\mu})^T\Big]\Big\} \tag{12}$$

(iii) Let $\boldsymbol{A} = \boldsymbol{\Sigma}^{-1}\widehat{\boldsymbol{\Sigma}}_{\text{MLE}}$, and $\lambda_1,...,\lambda_d$ be the eigenvalues of $\boldsymbol{A}$. Show that the result from the previous part leads to:

$$p(\boldsymbol{x}_1,...,\boldsymbol{x}_N|\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{Nd/2}|\widehat{\boldsymbol{\Sigma}}_{\text{MLE}}|^{N/2}}\Big(\prod_{i=1}^d\lambda_i\Big)^{N/2}\exp\Big\{-\frac{N}{2}\sum_{i=1}^d\lambda_i\Big\} \tag{13}$$

Hint: For matrix $\boldsymbol{A}$ with eigenvalues $\lambda_1,...,\lambda_d$, $\text{tr}[\boldsymbol{A}] = \sum_{i=1}^d\lambda_i$.

(iv) Show that the likelihood function is maximized by the choice $\lambda_i = 1$ for all i. Therefore, (10) maximizes (11).

(b) Alternatively, directly compute the derivative of the likelihood function (11) with respect to $\boldsymbol{\Sigma}$, set it to (matrix-valued) zero, and verify that (10) still solves your equation.

Hint: You might find it more convenient to work with the precision matrix $\boldsymbol{P} = \boldsymbol{\Sigma}^{-1}$, and consider the following matrix calculus identities:

$$\frac{\partial\text{tr}[\boldsymbol{A}\boldsymbol{X}]}{\partial\boldsymbol{X}} = \frac{\partial\text{tr}[\boldsymbol{X}\boldsymbol{A}]}{\partial\boldsymbol{X}} = \boldsymbol{A}, \text{ and } \frac{\partial|\boldsymbol{X}|}{\boldsymbol{X}} = |\boldsymbol{X}|\boldsymbol{X}^{-1} \tag{14}$$

Exercise 2: Cat Classification

Image classification is an important problem in computer vision and (probably) the most widely used test bed problem in artificial intelligence. In this exercise, we are going to study a simplified version of a common image classification problem by classifying foreground and background regions in an image, using a Gaussian classifier.

The image that we are going to classify consists of a cat and some grass [1]. The size of this image is 500×375 pixels. The left hand side of Figure 1 shows the image, and the right hand side of Figure 1 shows a manually labeled "ground truth". Your task is to do as much as you can to extract the cat from the grass, and compare your result with the "ground truth". However, before proceeding on to tackling the task, let us establish some conventions, and understand further how our classifier is going to work.

We first need to understand what a patch is in an image. By patch we mean a small neighborhood surrounding the pixel. For example, when we say extract an 8×8 patch at pixel (i, j), we mean that we want to extract `Y[i:i+8, j:j+8]` from the image. **In this exercise, we assume the size of our patches to be** 8×8.

[1] Image Source: `http://www.robots.ox.ac.uk/vgg/data/pets/`

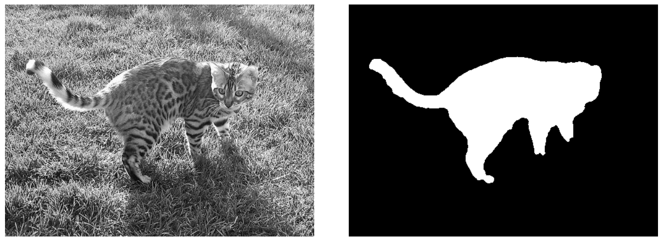

Figure 1: The "Cat and Grass" image.

Our classifier will be based on the MAP method discussed before, using the discriminant functions (6), i.e., we assume the posterior distrbutions to be Gaussian. More specifically, the MAP procedure is as follows: for any patch $\boldsymbol{x} \in \mathbb{R}^{64=8\times 8}$ in the image,

$$g_{\text{cat}}(\boldsymbol{x}) \ \underset{<\text{grass}}{\overset{\geq\text{cat}}{}} \ g_{\text{grass}}(\boldsymbol{x}) \tag{15}$$

For purposes of testing, we adopt the convention of assigning a value 1 to the patch $\boldsymbol{x}$ if it is classified as "cat", and 0 otherwise (this is the convention the "ground truth" image is constructed on). Furthermore, you are given training data files `train_cat.txt` and `train_grass.txt`. The sizes of the arrays in these files are $64 \times N$, where N corresponds to the number of training samples and 64 corresponds to the size of the block 8×8. You will need to use these data to compute the necessary parameters for your Gaussian classifier with formulas (7), (8) and (9).

Now, let us start training and testing the classifier.

Tasks:

(a) We want to train the classifier first.

(i) Load the data using

```
train_cat = np.matrix(np.loadtxt('train_cat.txt', delimiter = ','))
train_grass = np.matrix(np.loadtxt('train_grass.txt', delimiter = ','))
```

The data need to be stored as matrices for later use.

(ii) Now compute the sample mean vectors $\widehat{\boldsymbol{\mu}}_{(\text{cat})}$, $\widehat{\boldsymbol{\mu}}_{(\text{grass})}$, sample covariance matrices $\widehat{\boldsymbol{\Sigma}}_{(\text{cat})}$, $\widehat{\boldsymbol{\Sigma}}_{(\text{grass})}$, and sample priors $\widehat{\pi}_{(\text{cat})}$, $\widehat{\pi}_{(\text{grass})}$ using the data arrays. Organize your code into a function for cleaner presentation.

Hint: If you want to convert a patch `z` into a 64×1 column vector, you can type

```
z_vector = z.flatten('F');
```

To convert `z_vector` back to a patch, you can type

```
z = reshape(z_vector, (8, 8), order = 'F');
```

(b) Let us now test the classifier on `cat_grass.jpg`.

(i) Read the testing image `cat_grass.jpg`. The suggested way to read it is

```
Y = plt.imread('cat_grass.jpg') / 255
```

We divide the whole array by 255 to convert the into a floating point array within range (0,1). Such conversion is necessary because the training was done in such range.

(ii) Write a function that outputs your classifier's classification result on every **overlapping** patch in the testing image `cat_grass.jpg`. Your function should be in the following form:

```
Output = np.zeros((M-8,N-8))
for i in range(M-8):
    for j in range(N-8):
        z = Y(i:i+8, j:j+8)
        % intermediate steps
        % if patch z classified as cat, then set Output(i,j) = 1
```

You can plot the output using

```
plt.imshow(output*255, cmap = 'gray')
```

Remark 1: Since the double for loop run over every $i \in \{1, ..., M-8\}$ and $j \in \{1, ..., N-8\}$, this way of classification indeed have the patches overlapping.

Remark 2: By performing the above procedure, we left out the boundary pixels. Of course, this can be fixed (how?) but we will not drill into them.

(iii) Carry out the same task as in (ii), except now classify the patches **independently/ non-overlappingly** with each other; in other words, your patches should now have indices $(i, j) \in \{(0,0), (0,8), (8,0), ...\}$. Be cautious about the boundary pixels, we will still ignore those pixels in our classifier, i.e., for the pixels belonging to the set

$$(\{\texttt{Y[i,j]}|i > M-8\} \cup \{\texttt{Y[i,j]}|j > N-8\}) \cap (\{\texttt{Y[i,j]}|i \mod 8 \neq 0\} \cup \{\texttt{Y[i,j]}|j \mod 8 \neq 0\}) \tag{16}$$

leave your `Output(i,j)` matrix to be 0 for those boundary entries.

(iv) Let us examine the performance of our classifiers constructed in (ii) and (iii), by comparing the classification results to the "ground truth" image `truth.png`. To do so, we use mean absolute error as the comparison metric. The mean absolute error between an estimate $\boldsymbol{y}$ and the ground truth $\boldsymbol{y}^*$ is defined as

$$\text{MAE}(\boldsymbol{y}, \boldsymbol{y}^*) = \frac{1}{N}\sum_{n=1}^{N} |y_n - y_n^*|.$$

Report on the accuracy of the two versions of the Gaussian classifier.

Hint: Remember to read `truth.png` the same way you did in (b)(i).

(v) Go to the internet and download an image with similar content: a cheetah on grass, a cat on grass, or something like that. Apply your classifier to the image, and submit your resulting mask. Does it perform well? If yes, why? If no, what could go wrong? Write one to two bullet points to explain your findings. Please be brief; one sentence for each bullet point.

Exercise 3: Connection to Linear Least-Squares

In this exercise, let us take a step back, and study an important theoretical property of the Gaussian classifier: its connection to the linear least-squares (LLS) classifier we studied in homework 2 in the 2-class case. This will shed light on interesting geometric properties of the two classifiers. Before any derivations, we state some assumptions first, and recap on some theory.

Suppose that we have two classes of data, with datasets $\mathcal{D}_1 = \{(\boldsymbol{x}_i^{(1)}, y_i^{(1)})\}_{i=1}^{N_1}$ and $\mathcal{D}_2 = \{(\boldsymbol{x}_i^{(2)}, y_i^{(2)})\}_{i=1}^{N_2}$, with $\boldsymbol{x}_i^{(j)} \in \mathbb{R}^d$, and $y_i^{(j)} \in \mathbb{R}$. Recall from lecture that **unless** the two classes share the same **between-class** sample covariance matrix

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N_1 + N_2 - 2}\sum_{j=1}^{2}\sum_{i=1}^{N_j}(\boldsymbol{x}_i^{(j)} - \widehat{\boldsymbol{\mu}}_j)(\boldsymbol{x}_i^{(j)} - \widehat{\boldsymbol{\mu}}_j)^T \tag{17}$$

the discriminant functions from (6) are not going to be linear. Recalling that the decision boundary of the LLS classifier is linear, **we will limit ourselves to the equal covariance case in this exercise**, because in our case, only linear discriminant functions produce linear decision boundaries. Additionally, we assume $\widehat{\boldsymbol{\Sigma}}$ is invertible.

Let us also recall some details of the LLS classifier. We encoded class 1 with class label $+1$, and class 2 with class label -1. The decision boundary of the classifier was described by the set $\{\boldsymbol{x} \in \mathbb{R}^d | \ \boldsymbol{w}^T\boldsymbol{x} + w_0 = 0\}$, where $\boldsymbol{w}$ and w_0 are characterized by the equation

$$\boldsymbol{A}^T\boldsymbol{A}\begin{bmatrix}\boldsymbol{w}\\ w_0\end{bmatrix} = \boldsymbol{A}^T\boldsymbol{b} \tag{18}$$

where $\boldsymbol{A} \in \mathbb{R}^{(N_1+N_2)\times(d+1)}$ and $\boldsymbol{b} \in \mathbb{R}^{N_1+N_2}$ were formed from the $\boldsymbol{x}_i^{(j)}$'s and $y_i^{(j)}$'s.

In this exercise, we are going to show that when $N_1 = N_2$, the Gaussian classifier produces the same decision boundary between the two classes as does the linear least square classifier.

Tasks:

(a) To warm up, write the expression for the decision boundary of the linear Gaussian classifier in the form $\{\boldsymbol{x} \in \mathbb{R}^d | \ \boldsymbol{\beta}^T\boldsymbol{x} + \beta_0 = 0\}$ (i.e. find $\boldsymbol{\beta}$ and β_0).

(b) Instead of adhering to the problem setup strictly, we prove a more general preliminary result first. But before that, we need to prove a technical lemma.

First, let us give a general encoding of $c_1 \in \mathbb{R}$ to class 1 (replacing $+1$), and encoding $c_2 \in \mathbb{R}$ to class 2 (replacing -1), where $c_1 \neq c_2$. Be careful that the expression of $\boldsymbol{b}$ depend on c_1 and c_2. Also, for convenience, denote $N = N_1 + N_2$.

We want to show that for $\boldsymbol{w}$ satisfying (18), the following equation holds:

$$[(N-2)\widehat{\boldsymbol{\Sigma}} + \frac{N_1N_2}{N}(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)^T]\boldsymbol{w} = \Big(\frac{c_2N_1N_2 - c_1N_1N_2}{N}\Big)(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1) \tag{19}$$

The derivation is somewhat complicated, so we will show the result in a few steps:

(i) Prove that

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix}(N-2)\widehat{\boldsymbol{\Sigma}} + N_1\widehat{\boldsymbol{\mu}}_1\widehat{\boldsymbol{\mu}}_1^T + N_2\widehat{\boldsymbol{\mu}}_2\widehat{\boldsymbol{\mu}}_2^T & N_1\widehat{\boldsymbol{\mu}}_1 + N_2\widehat{\boldsymbol{\mu}}_2\\ N_1\widehat{\boldsymbol{\mu}}_1^T + N_2\widehat{\boldsymbol{\mu}}_2^T & N\end{bmatrix} \tag{20}$$

Hint: Write $\widehat{\boldsymbol{\Sigma}}$ in terms of the $\boldsymbol{x}_i^{(j)}$'s and $\widehat{\boldsymbol{\mu}}_j$'s using its definition, and observe its relation to $[\boldsymbol{A}^T\boldsymbol{A}]_{1,1}$.

(ii) Prove that

$$\boldsymbol{A}^T\boldsymbol{b} = \begin{bmatrix}c_1N_1\widehat{\boldsymbol{\mu}}_1 + c_2N_2\widehat{\boldsymbol{\mu}}_2\\ N_1c_1 + N_2c_2\end{bmatrix} \tag{21}$$

(iii) Using previous results and equation (18), show that

$$\begin{aligned}&\big[(N-2)\widehat{\boldsymbol{\Sigma}} + N_1\widehat{\boldsymbol{\mu}}_1\widehat{\boldsymbol{\mu}}_1^T + N_2\widehat{\boldsymbol{\mu}}_2\widehat{\boldsymbol{\mu}}_2^T\big]\boldsymbol{w} + (N_1\widehat{\boldsymbol{\mu}}_1 + N_2\widehat{\boldsymbol{\mu}}_2)\Big(\frac{N_1c_1 + N_2c_2}{N} - \frac{(N_1\widehat{\boldsymbol{\mu}}_1^T + N_2\widehat{\boldsymbol{\mu}}_2^T)\boldsymbol{w}}{N}\Big)\\ &= c_1N_1\widehat{\boldsymbol{\mu}}_1 + c_2N_2\widehat{\boldsymbol{\mu}}_2\end{aligned} \tag{22}$$

(iv) Prove that the left hand side of the equation from the last part can be simplified to the following:

$$\big[(N-2)\widehat{\boldsymbol{\Sigma}} + \frac{N_1N_2}{N}(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)^T\big]\boldsymbol{w} + (N_1\widehat{\boldsymbol{\mu}}_1 + N_2\widehat{\boldsymbol{\mu}}_2)\Big(\frac{N_1c_1 + N_2c_2}{N}\Big) \tag{23}$$

Hint: This problem requires some patience. Expand the expression, and prove that the outer product terms

$$\begin{aligned}&-\frac{N_1^2}{N}\widehat{\boldsymbol{\mu}}_1\widehat{\boldsymbol{\mu}}_1^T + N_1\widehat{\boldsymbol{\mu}}_1\widehat{\boldsymbol{\mu}}_1^T - \frac{N_2^2}{N}\widehat{\boldsymbol{\mu}}_2\widehat{\boldsymbol{\mu}}_2^T + N_2\widehat{\boldsymbol{\mu}}_2\widehat{\boldsymbol{\mu}}_2^T - \frac{2N_1N_2}{N}\widehat{\boldsymbol{\mu}}_1\widehat{\boldsymbol{\mu}}_2^T\\ &= \frac{N_1N_2}{N}(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)(\widehat{\boldsymbol{\mu}}_2 - \widehat{\boldsymbol{\mu}}_1)^T\end{aligned} \tag{24}$$

(v) Finally, prove that (19) holds.

(c) As a corollary of the lemma we showed in (b), show that $\boldsymbol{w}$ must be in the direction of $\boldsymbol{\beta}$ (i.e. $\boldsymbol{w}$ is a scalar multiple of $\boldsymbol{\beta}$), even though our encoding of the classes for the LLS classifier is completely arbitrary.

(d) Show that in the case of $c_1 = +1, c_2 = -1$ (i.e. the original encoding), if $N_1 = N_2$, then the decision boundary generated by the Gaussian classifier is the same as that generated by the LLS classifier. In other words, show that there exists some scalar a such that $a\boldsymbol{\beta} = \boldsymbol{w}$ and $a\beta_0 = \omega_0$.

(e) Now let us verify our theory with a simple experiment. We carry it out in a few steps:

(i) Generate two clusters of data, each with Gaussian distribution, equal covariance but different means, and each of size 1000. Choose whatever mean vectors you like, but try to make the two clusters at least visually separable. Make a scatter plot of the data.

(ii) Using the expression you obtained in (a), plot the decision boundary on top of the scatter plot of the two classes of data you generated in the previous part.

(iii) Now obtain the decision boundary by solving the linear least square problem in the same way you did in homework 2, i.e., solve for the optimal $\boldsymbol{w}$ and w_0 satisfying (18). Plot this decision boundary, and compare it to the previous one. What do you see?

(f) Having gone through this exercise, make some further comments on how you would interpret (18) geometrically, in the specific case of $N_1 = N_2 = N/2$ and $c_1 = +1, c_2 = -1$.