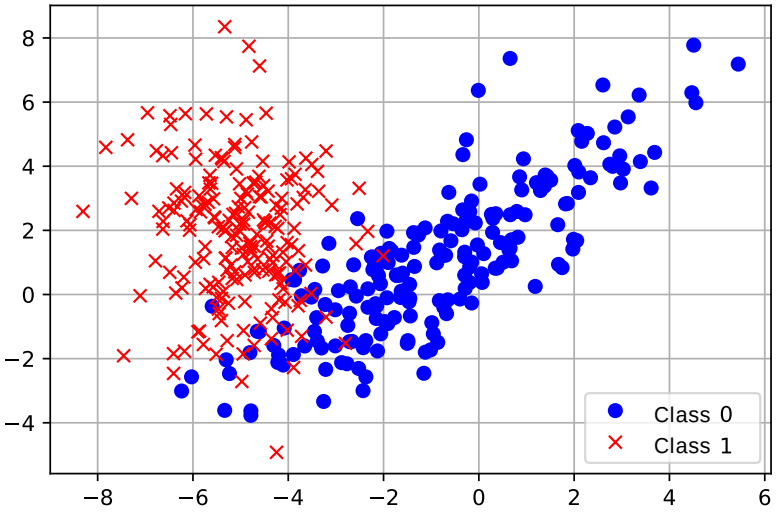
COMP.SGN.100 Introduction to Signal Processing,
Exercise 12, 13.-15.10.2021

Pen & paper task solutions should be submitted to Moodle at least one hour before your exercise session. Matlab tasks are done during the exercise session.

Task 1. *(Pen & paper)* When designing a linear classifier for two-dimensional data (figure below), we obtain from the training data the following means and covariance matrices for the two classes:

$$\boldsymbol{\mu}_0 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \boldsymbol{\mu}_1 = \begin{pmatrix} -5 \\ 2 \end{pmatrix}$$
$$\mathbf{C}_0 = \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix} \quad \mathbf{C}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

Calculate the vector $\mathbf{w}$ that determines the projection line and draw it to the figure.



Task 2. *(Pen & paper)* We continue the previous task. In addition to the direction of the projection line, we need to determine the point where the boundary between the classes is drawn. The simplest way is to place it to the midpoint between the mass centers of the classes. In practice, this is done by projecting the data on the projection line and comparing the result with the threshold $c \in \mathbf{R}$:

$$\begin{cases} \text{The sample } \mathbf{x} \text{ belongs to the class 0, if } \mathbf{w} \cdot \mathbf{x} \geq c. \\ \text{The sample } \mathbf{x} \text{ belongs to the class 1, if } \mathbf{w} \cdot \mathbf{x} < c. \end{cases}$$

Calculate the threshold c as follows:

- Calculate the value where $\boldsymbol{\mu}_0$ is projected to.
- Calculate the value where $\boldsymbol{\mu}_1$ is projected to.
- Threshold c is the mean of these two values.

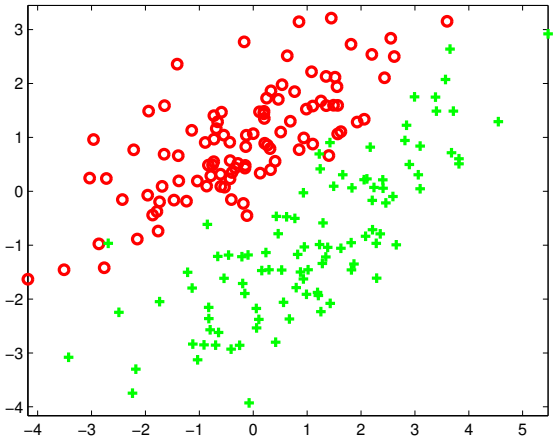
Task 3. *(Matlab)* We implement the LDA in this and the next task for two-dimensional synthetic data. In Task 5 we apply the method to image processing.

(a) Download the file `testdata_fisher.mat` from the course Moodle (`Ex_12.zip`). The file contains two matrices with two-dimensional data, each matrix representing one class. Load the file with the command

```
load testdata_fisher.mat
```

after which two new variables are in Matlab: X1 $\in \mathbf{R}^{100\times 2}$ contains 100 two-dimensional data points from class 1 and X2 $\in \mathbf{R}^{100\times 2}$ the same amount of data points from class 2. For both, the first column contains the x-coordinates of the data points and the second column the y-coordinates.

Plot in the same figure the data points of the matrix X1 with red circles and of X2 with green plus signs as shown in the figure below.



(b) We will now classify the two classes in (a) by the LDA. The method projects the two-dimensional data on the line that best separates the classes. After that it is easy to separate the classes by simply finding a suitable threshold from the line.

The best possible line is defined as a vector as follows:

$$\mathbf{w} = (\mathbf{C}_1 + \mathbf{C}_2)^{-1}(\mu_1 - \mu_2),$$

where $\mathbf{C}_1$ and $\mathbf{C}_2$ are of size 2×2 covariance matrices of the classes and μ_1 and μ_2 are the means of the classes as 2×1 vectors. Calculate the vector $\mathbf{w}$. The covariance is obtained by the function `cov` and the mean by the function `mean`. Plot the vector in the figure using the command `line`. The command `hold on` given before the plotting command holds the previous figure for comparison. Also, use the command `axis equal` to make the coordinate system more square and the comparison simpler.

Task 4. *(Matlab)* This continues the previous task. In addition to the direction of the projection line, we need to find the threshold between the classes on the line. The simplest way is to select it to be the midpoint between the means of the classes. In practice, this is done by projecting the data on the line and comparing the result with the threshold value $c \in \mathbf{R}$:

$$\begin{cases} \text{The sample } \mathbf{x} \text{ belongs to the class 1, if } \mathbf{w} \cdot \mathbf{x} \geq c. \\ \text{The sample } \mathbf{x} \text{ belongs to the class 2, if } \mathbf{w} \cdot \mathbf{x} < c. \end{cases}$$

(a) Calculate the threshold value c as follows:

- Calculate the value where μ_1 is projected.
- Calculate the value where μ_2 is projected.
- Threshold c is the mean of these two values.

(b) Calculate what percentage of the samples are classified correctly. [1]

Task 5. *(Matlab)* We now apply the LDA to real data. Download the file `canoe.jpg` from the course Moodle (`Ex_12.zip`). Read it to Matlab and show the image:

```
imOrig = imread('canoe.jpg');
imshow (imOrig, []);
```

The canoe of the image is very distinct for its red color. We will project it with LDA so that the separation of red from the other colors is as good as possible. To do this, you need a training set that can be obtained by selecting points from the image. Command `[x,y] = ginput(1);` returns the coordinates of the point in the image clicked by the mouse. Do this twice so you get two training sets: one from the canoe and the other one, for example, from the surface of the water. Let the coordinates thus obtained be in vectors `x1`, `y1` and `x2`, `y2`. Take 7×7 square neighborhoods of these points to obtain two training classes:

```
window1 = imOrig(y1-3:y1+3, x1-3:x1+3,:);
window2 = imOrig(y2-3:y2+3, x2-3:x2+3,:);
```

Note that the data is now three-dimensional: color information is presented as RGB values having dimension 3. In order to get this into the 3×49 matrix form required by the LDA, use the following commands:

```
X1 = double(reshape(window1, 49, 3))';
X2 = double(reshape(window2, 49, 3))';
```

Use the program code of Task 3 to obtain a three-dimensional vector that describes the optimal projection for separating the two colors. What is the vector in question?

[1] Note that in a real case the error should not be evaluated with the training data. For example, a 1-NN classifier gives perfect classification when tested with the training data, which does not mean that it would be good for any other data set.

The projection itself is done by multiplying the components of the vector **w** by the R, G and B components of the image, for example in this way:

```
imGray = w(1)*double(imOrig(:,:,1)) +...
         w(2)*double(imOrig(:,:,2)) +...
         w(3)*double(imOrig(:,:,3));
```

Show the resulting image. The red area from the canoe you clicked should glow as bright white and the other area as almost black.

Try to separate the red area from the others by thresholding the values above a certain threshold to white and others to black. As a Matlab command

```
imshow(imGray > threshold, []);
```

where the parameter `threshold` is the limit, above which the image points become white and the others become black. Are you able to separate the canoe from the background?