SGN-11007 Introduction to Signal Processing,
Exercise 6, Fall 2020

Task 1. *(Pen & paper)* The signal $x(n)$ with the sampling rate 12 kHz should be converted to a signal with the sampling rate 15 kHz. Determine the steps of the conversion as a block diagram using resampling ($\boxed{\uparrow L}$ and $\boxed{\downarrow M}$) and low-pass filtering ($\boxed{H(z)}$). Specify the passband and stopband intervals of the required low-pass filters, when the frequencies on the interval $0 - 5.5$ kHz are to be preserved.

Task 2. *(Pen & paper)* The goal is to reduce the sampling rate of a 12 kHz signal to one kilohertz in several stages. The most important information in the signal is on the frequency interval $0 - 450$ Hz, which will therefore be strictly preserved. Find out the total amount of the coefficients of the required filters in each multistage implementation (4 cases are enough to investigate, i.e. only the cases where the decimation coefficients are in descending order) when the low-pass filters are designed with the Hamming window ($N = 3.3/\Delta f$).

Task 3. *(Pen & paper)* How many multiplications per second (MPS) are needed in each of the implementations in Task 2? This is calculated using the formula

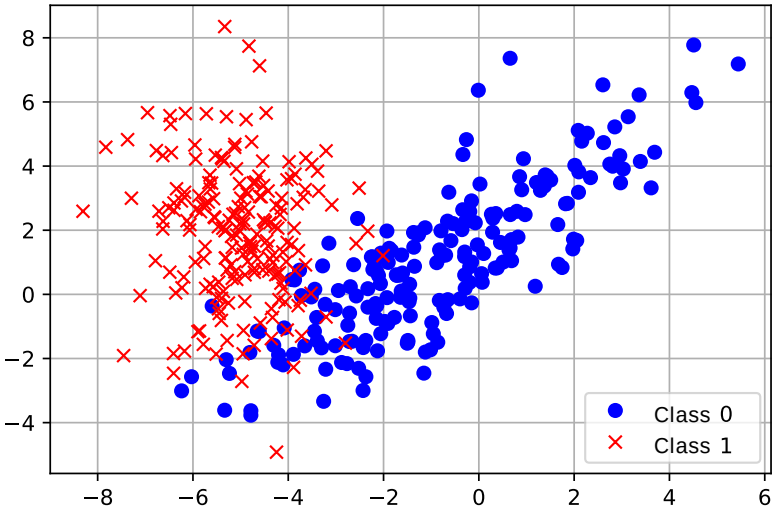
$$\text{MPS} = \sum_{i=1}^{I} N_i F_i,$$

where I is the number of decimation blocks, N_i is number of coefficients in the i^{th} block, and F_i is the sampling rate of the input signal of the i^{th} block.

Task 4. *(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 5. *(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 6. *(Matlab)* We will implement in this and the next task the sampling rate conversion in Matlab with factor $\frac{2}{3}$. First load Matlab's test signal `laughter` (with `load laughter` to the variable `y`). The sampling frequency of the signal is 8192 Hz.

In this task we double the sampling rate of the signal. This requires the following steps:

- Create a zero signal `z` with length twice the one of the original signal.
- Use the command `z(1:2:end) = y;` to assing values of the original signal to every second element of the zero signal. Listen to the result. Note that the command `soundsc` needs also the sampling rate as an input. Otherwise, the result sounds incorrect.
- Design e.g. an FIR filter of order 100 (command `fir1`) and filter the generated interference.
- Listen to the filtering result and check that it sounds the same as the original signal.
- Plot the spectrograms of all three signals with the command `spectrogram`.

Task 7. *(Matlab)* Now we reduce the sampling rate of the output signal of Task 6 to one third as follows:

- Design an antialiasing filter of order 100.
- Filter the signal `z` by the filter.
- Take every third sample of the result.
- Listen to the result at the sampling rate of 5461 Hz, which is about 2/3 of the original sampling rate.
- Plot the spectrograms of all three signals with the command `spectrogram`.

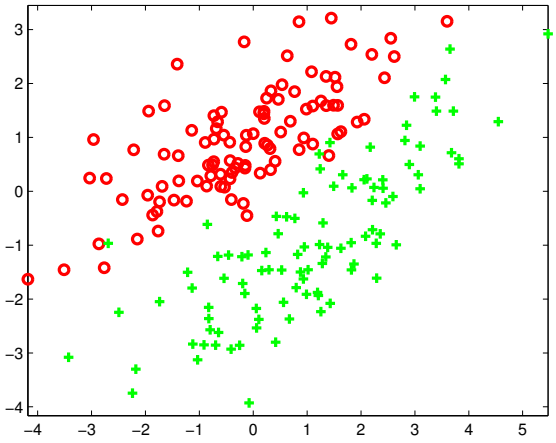
Task 8. *(Matlab)* We implement the LDA in this and the next task for two-dimensional synthetic data. In Task 10 we apply the method to image processing.

(a) Download the file `testdata_fisher.mat` from the course Moodle (`Ex_6.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 9. *(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 10. *(Matlab)* We now apply the LDA to real data. Download the file `canoe.jpg` from the course Moodle (`Ex_6.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 8 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?