Laboratory Test DEDP 2023-2024

Information

- The test will be taken during the lab, and will last for ~~1 hour~~ up to 2 hours
- General Matlab stuff you need to know is listed in the **Syllabus** section
- Template subjects (i.e. exercises extracted from the labs) are in **Template Subjects** section
- The test will be roughly based on these templates, with some modifications

Syllabus

General things to know in Matlab:

- Create and use vectors and matrices
- Create and use simple cell arrays (see Lab 2)
- General instructions: `if`, `for`, etc
- Create functions in Matlab and use them
- Load and save data from/to a `*.mat` file
- Generate random numbers with Gaussian and uniform distributions
  - single numbers, vectors or matrices with specified shapes
  - with specified parameters (mu and sigma for normal distribution, a, b for uniform distribution)
- Compute mean, average squared value, variance of vectors or columns
- Compute correlation and autocorrelation of vectors or columns (using `xcorr()`, as in Lab 2)

- Generate a random vector with 0's and 1's in variable proportions (or some value A instead of 1)
- Generate a sin or cos signal of a certain length, with a specified amplitude, frequency and initial phase
- Generate a vector with N values equally spaced between a start and a stop value (e.g. `linspace()`)
- Operate with columns (or rows) of a matrix:
  - extract one or more columns
  - arithmetic operations: add columns, divide element by element, etc
  - same with rows instead of columns
- Count how many values of 1 are in a vector (or maybe how many values equal to A)
- Count pair of values in two vectors (e.g. count when there is a 0 in a vector and 1 in another vector, like for false alarms, misses etc)
- Compute Euclidean distance between vectors
- Sort a vector and find the original positions of the smallest k values (use `sort()`)
- Find the maximum value in a vector and its position (using `max()`)
- Find the minimum value in a vector and its position (using `min()`)
- Plot a vector
- Plot a vector as a function of another vector (i.e. `plot(x,y)`)
- Plot a histogram plot (using `hist()`)
- Use the `fitcknn()` and `predict()` functions (for k-NN algorithm)
- Use the `kmeans()` function (for k-Means algorithm)
- Use the `fminsearch()` function for ML estimation
- Define an anonymous function (as in the examples with `fminsearch()`)

Sample (template) subjects from each lab

Lab 1

1. Create a Matlab function `myPDF()` that estimates the probability density function from a vector of data.

    - the function requires three arguments and returns one value: `p = myPDF(v,x,epsilon)`
    - `v` is a vector, and `x`, `epsilon` are scalar numbers
    - the function computes how many elements from `v` are in the interval $[x-\epsilon, x+\epsilon]$, divided to the total number of elements of `v`, and also divided to 2 times `epsilon`

2. Plot the probability density function estimated from a vector of data

    - generate a vector v with 100000 values from the normal distribution $\mathcal{N}(2,2)$ and plot the values
    - generate a vector n of 50 values uniformly spread between -5 to 15
    - apply `myPDF()` on v to estimate the probability density at every value from n (use `epsilon = 0.1`)
    - plot the results of the function against the values of n

Lab 2

1. Load the file `ElectionsData.mat`. It contains election data for the local elections in the city of Iasi held on 27.09.2020.

    - names: a cell array with the names of the voting centers
    - values: a matrix with the voting numbers for each center

    The structure of the values matrix is as follows:

    - first column: total number of registered voters on permanent lists
    - second column: total number of registered voters on complementary lists
    - third column: number of votes from permanent lists
    - fourth column: number of votes from complementary lists
    - fifth column: number of votes from supplementary lists
    - sixth column: number of votes with mobile urns

    a. Compute the **turnout** for every voting center, defined as: total number of votes (columns 1 + 2) / total number of registered voters on all lists (columns 3 + 4 + 5 + 6).
    b. Plot the turnout vector
    c. Compute the mean and the variance of the turnout vector

Lab 3

1. Simulate threshold-based detection with a single sample, as follows:

   - Generate a vector of 100000 values 0 or A, with equal probability (consider $A = 5$)
   - Add over it a random noise with normal distribution $\mathcal{N}(0, \sigma^2 = 1)$
   - Pick a value of $T = A/2$ and compare each element with T to decide which sample is logical 0 or logical 1 (A)
   - Compare the decision result with the true original vector, and count how many correct detections and how may false alarms have been
   - Estimate the four probabilities by dividing the above numbers to the size of the vector

Variant: Same exercise, but written as a function which accepts T as input and return the four values as outputs. Running the function for 100 values of T uniformly spaced between 0 and A, and plotting the resulting vector `pcd` against `pfa`.

Lab 4

1. Load two data matrices from `*.mat` files (e.g. `ECG_train.mat` and `ECG_test.mat`) and use the built-in functions `fitcknn()` and `predict()` to predict the class of some signals with the k-NN algorithm

   Use different values for k: $k = 1$, then $k = 5$, then $k = 15$.

Lab 5

1. Use `k-Means` to clusters colors in an image:

   - Load an image (`'Peppers.tiff'`) using `imread()` and convert to double.
   - Use the `reshape()` function to resize the $M \times N \times 3$ image `I` into a $(M * N) \times 3$ matrix `P`, as follows:

     ```
     P = reshape(I, [], 3);
     ```

   - Use the `kmeans()` Matlab function to cluster the pixels, with $k = 3$
   - Replace each pixel of the image with the *centroid* of its class, then display the resulting image.

Variant: Identify the largest cluster and replace each pixel of that class with a certain color (e.g. with black = [0, 0, 0]).

Lab 6

Variant A

1. Generate a 500-samples long sinusoidal signal $s_\Theta = A * \sin(2\pi f n)$ with frequency $f = 0.02$, and add over it normal noise with distribution $\mathcal{N}(0, \sigma^2 = 0.5)$. Name the resulting vector `r`. Plot the `r` vector.

2. Estimate the frequency $\hat{f}$ of the signal via Maximum Likelihood estimation from the `r` vector:

   - Generate 10000 candidate frequencies f_k equally spaced from 0 to 0.5
   - Compute the Euclidean distance between `r` and the sine signal with each candidate frequency
   - Maximum Likelihood: choose $\hat{f}_{ML}$ as the candidate frequency which minimizes the Euclidean distance
   - Display the estimate value $\hat{f}_M L$

Variant: estimate amplitude A instead of frequency f, in the same way

Variant B

Use `fminsearch()` to find the ML estimate of some parameter(s) in a signal expression, by minimizing the Euclidean distance between `r` and that expression.

1. Use `fminsearch()` to fit a linear curve $y = ax + b$ through the following points:

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
y = [0.11, 0.21, 0.29, 0.43, 0.5, 0.58, 0.69, 0.81, 0.91, 0.99];
```

   a. Estimate the values of `a` and `b` using `fminsearch()`
   b. Generate the vector `y_est` with the estimated parameters.
   c. Plot `y_est` and `y`

or:

1. Generate a 500-samples long sinusoidal signal $s_\Theta = A * \sin(2\pi f n)$ with frequency $f = 0.02$, and add over it normal noise with distribution $\mathcal{N}(0, \sigma^2 = 0.5)$. Name the resulting vector `r`. Plot the `r` vector.

2. Use `fminsearch()` to estimate the value of `f` (or of `A`, or of both `A` and `f`) with Maximum Likelihood estimation from the `r` vector.