Laboratory Test

DEDP 2019-2020 Final

Information

- There are 6 subjects in all, shown below according to the laboratory they were done in.
- The test will last for up to 110 minutes (~a full 2h normal lab session)

Subjects

Lab 1

1. Create another Matlab function `myPDF()` that directly 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, `x`, `epsilon` and `p` 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 * 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. Simulate threshold-based detection with a single sample, as follows:
    - Generate a vector of 100000 values 0 or A, with equal probability (hint: use `rand()` and compare to 0.5). Use $A = 5$.

- Add over it a random noise with normal distribution $\mathcal{N}(0, \sigma^2 = 1)$
- 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 rejections, false alarms, misses and correct detections have happened;
- Estimate probability of correct rejection, false alarm, miss and correct detection (by dividing the above counters to the size of the vector).

Lab 3

1. Simulate the BPSK sender
   - Generate a vector **`data`** of 1000 values 0 or 1, with equal probability (hint: use `rand()` and compare to 0.5).
   - Generate a vector **`signal`** of 100000 values as follows: for each bit in **`data`**, put a 100-long sine $\pm A \sin(2\pi f n)$ in **`signal`**. Use $A = 1$, $f = 1/100$. 0 corresponds to $+A$, 1 to $-A$.
   - Plot the resulting vector `signal`.
   - Generate a vector of white gaussian noise with distribution $\mathcal{N}(0, \sigma^2)$, the same length as `signal`, and $\sigma^2 = A/10$.
   - Add the noise to the signal, store result as **`r`**.
   - Plot the resulting signal `r` (do not overwrite the previous figure of `s`, we want to see both)

Lab 4

1. Implement a function `[class] = myKNN(signal, k, trainset)` for performing k-NN classification of a signal:
   - the function takes as input an unclassified signal `signal`, the parameter value `k`, and the training set matrix `trainset`
   - the function computes the Euclidean distance between `signal` and each vector from the training set
   - the output `class` is defined by the majority of the k nearest neighbours of the signal
2. Call the function `myKNN` for the first signal from the testing set and determine its class. Use different values for k: $k = 1$, then $k = 5$, then $k = 15$.

Note: the training set matrix can be loaded from the file `ECG_train.mat`, and the test set from `ECG_test.mat`

Lab 5

1. Load the color image 'Peppers.tiff' using `imread()`. Convert the image to `double` and display it (don't convert to grayscale, leave the colors).

2. Use Matlab's k-Means algorithm to cluster all the pixel values (each pixel = a group of three values R, G, B) into 4 groups.

3. Replace each pixel of the image with the *centroid* of its class. Display the image.

   Hint:

   - You can reshape a matrix A to a column vector using `A(:)`
   - You can reshape a column vector `v` to matrix of size `m,n` using `reshape(v, m, n)`
   - Alternatively, you can use the `reshape()` function to resize a $M \times N \times 3$ tensor `I` into a $(M * N) \times 3$ matrix `P`, as follows:

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

   - Use the `kmeans()` Matlab function to do the clustering. You can read the documentation for more details.

Lab 6

1. Generate a 500-samples long sinusoidal signal $s_\Theta = \sin(2\pi f n)$ with frequency $f = 0.01$, 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 1000 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 $\hat{f}_M L$
   - Plot a sinusoidal with the estimated frequency $\hat{f}_{ML}$, and the original vector `r`, on the same figure