Assignment 3

Prof.	Panos Trahanias	trahania@csd.uoc.gr
T.A.	Despina - Ekaterini Argiropoulos	despargy@csd.uoc.gr
T.A.	Myrto Villia	mvillia@ics.forth.gr
T.A.	Michalis Savorianakis	csdp1387@csd.uoc.gr
Deadline	**Tuesday 23:59, 29th of April 2025**	

Note: This is a personal assignment and should be pursued individually and without use of any computerised AI facilities. Note that this will be automatically checked for every delivered assignment and for cases of relevant detections the whole assignment will be dropped. The assignment should be implemented entirely in Google Colaboratory following the delivered instructions below. **Comments regarding the submitted code are mandatory.**

Part A: (20/100)

Maximum Likelihood Estimation for Parameter Estimation

The scope of this exercise is to get familiar with the Maximum Likelihood Estimation (MLE) technique in order to calculate the parameters of some given distributions. You are given 3 different normal distributions representing 3 different classes. Each distribution is about a different class of 2-dimensional instances. You are asked to find the parameters describing the distribution of each class.

Data – Part A: For this exercise, you will use the dataset included in *dataA_MLE.csv* file. The dataset consists of 900 samples by row, 300 for each different class (ex. first 300 rows are about class 0, next 300 rows are about class 1 and the last 300 rows are about class 2). Notice that there are 3 columns. The first 2 columns represent the features of the data and the last column represents the class label 0, 1, 2.

!Notice: You are not allowed to use library functions, everything should be developed from scratch.

Question:

1. Implement a function which takes as input an array of samples and returns the mean and the covariance matrix of those samples.
2. Print the mean and the covariance matrix for each one of the 3 classes, using the function from Question 1.
3. Considering the function of Question 1, plot each class distribution in one single 3D plot.

Part B: (40/100)

Parzen Windows

The scope of this exercise is to get familiar with the Parzen Windows in order to estimate the probability density function of the distribution of some given data.

Data – Part B: For this exercise, you will use the dataset included in *dataB_Parzen.csv* file. The dataset consists of 200 samples by row, each sample is 1-dimensional (1 column). We do not have any knowledge or assumption about their distribution.

!Notice: You are not allowed to use library functions, everything should be developed from scratch.

Question:

1. Implement the window function $\phi(u)$, when window is a hypercube.
2. Implement the window function $\phi(u)$, when window is a Gaussian kernel.
3. Implement a function which takes as input a single point x_i, the center of the window as a single point c, the width h of the window and the $kernel$ type (*'hypercube'* or *'Gaussian'*) of the window. The function calls one of the above implemented functions (hypercube window or Gaussian window), with the appropriate input, and returns the result.
4. In this question, you are asked to develop the Density Function of Parzen Window. Implement a function which takes as input an array of 1-d points $data$, a single point x which represents the center, the width h of the window and the kernel type of the window. The function should return the likelihood of the center x, given the other inputs.
5. What's the best value for the width of the window h? To find this, assume that the dataset you have comes from the normal distribution N(1, 1) (this is a univariate normal distribution). Find the most suitable value for h based on that knowledge.
   (a) Create a histogram of the data to convince yourselves that they come from the aforementioned distribution.
   (b) For every h in the range [0.05, 5] with step = 0.1 calculate 1) the predicted likelihood for every point in the data, 2) their true likelihood (you can use the function norm.pdf(data, loc=1, scale=1)), and 3) the Mean Square Error of the two likelihoods (predicted and true). Repeat this process for both kernels (hypercube and Gaussian). What's the most suitable value for h for each kernel? Print your answer and create a plot which shows the values of h on the x-axis and their MSE on the y-axis (for both kernels).

Part C: (40/100)

K-Nearest Neighbors (KNN) Classification

The scope of this exercise is to get familiar with the K-Nearest Neighbors (KNN) Classifier.

Data – Part C: For this exercise, you will use the dataset included in *dataC_KNNtrain.csv* and *dataC_KNNtest.csv* files. Notice that each sample has 3 columns. The first 2 columns represent the 2-dimensional data and the last column represents their label (0,1).

Question:

1. Implement a function which has as input a 2D point x and a NumPy array *train_data* of 2D points, and it computes the Euclidean distances of that point x to all points in the given array. The function should return that NumPy array of the Euclidean distances.
2. Implement a function which has as input a 2D point x, a NumPy array *train_data* of 2D points and a number k. The function returns the k closer neighbors of x. As neighbors, we call all the points in *train_data*. **Hint:** Use the function from Question 1.
3. In this step, you are asked to develop the k-NN algorithm. Implement a function which has as input the train data, the test data and a number k of neighbors that will be considered during k-NN. The function should return two probabilities for each sample x_i of the test data, the probability of x_i sample belong to class 0 and the probability of x_i sample belong to class 1, respectively. These probabilities should add to 1.
4. In this question, you are asked to search for the best k number, meaning to select a number k that maximizes the accuracy of the k-NN classifier. Compute the accuracy of the classifier from question 3, for each k in the set of {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15} (e.g. acc = ? when k = 3, ..., acc = ? when k = 11, ...). Plot with point markers the above results, print and explain which k you would choose.
5. **Bonus 10%** Show the decision boundaries of your classifier in a 2D plot, using the value for k from the previous task.

Deliverable

This assignment should be implemented entirely in Google Colaboratory. Google's notebook allows you to combine executable Python scripts with rich text in a single document. Your deliverable should be a single '.ipynb' file along with its corresponding .py file (both can be easily exported from Google Colaboratory). Every single question should be implemented in a single code block. Code blocks should be clearly and shortly explained (you may use the text boxes for that goal). Use **only** library functions for matrix operations and plots.