Boğaziçi University
Department of Computer Engineering
Instructor: İnci M. Baytaş
Volunteering Teaching Assistant: Ege Onur Taga

Spring 2023
Assignment 2

CMPE 462 Assigment 2

This assignment consists mainly of programming questions. Familiarity with Python, Jupyter Notebooks, and relevant libraries are assumed, namely **numpy** and **matplotlib**.

For theoretical questions, we expect the answers in LaTeX. One option is to merge your theoretical answers and code into a single Jupyter Notebook where you can use markdown cells to insert LaTeX into the notebook. This is our preferred method of delivery, and we believe it would be easier for you as well. If you want to submit your code and solutions separately, you need to provide .py files with instructions on reproducing the results for each question.

In this assignment, you will continue working with the **3D Shapes** dataset provided to you in the first assignment.

Question 1. *Support Vector Machine*

(a) Train an SVM classifier for the 3D Shapes dataset. Please use the features you extracted in the first assignment. You may also extract new features if you want. There are 15 unique orientations that will be considered as 15 classes. You may use scikit-learn's SVM function in this question.
   i. Train an SVM classifier for a binary classification problem by choosing two orientations as the classes.
   ii. Train an SVM classifier for 15 class classification problem.
   iii. Compare the performances of 2 and 15 class classification with the same hyperparameters. Please discuss the reasons why their performances differ.

(b) Please repeat the steps above for hard-margin linear SVM, soft-margin linear SVM, hard-margin non-linear SVM with various kernel functions, and soft-margin non-linear SVM with various kernel functions. Please describe your strategy to tune the hyperparameters.

(c) Do you observe any improvement in computation time or classification performance if you apply PCA on your dataset before training the SVM classifier?

Question 2. *k-Means Clustering and Gaussian Mixture Models*

A *Gaussian Mixture* is a function that is a convex combination of multiple Gaussians, each of which is identified by $k \in \{1, 2, ..., K\}$ for a mixture of K Gaussians. The density is given explicitly as

$$p(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mu_k, \sigma_k^2)$$

where $\pi_1 + \pi_2 + ... + \pi_K = 1$ and $\pi_k \geq 0$ for all k. $\mathcal{N}(\cdot, \cdot)$ is the normal density, and μ_k and σ_k^2 are the mean and the variance of the kth Gaussian in the mixture respectively.

(a) Consider a 2-dimensional Gaussian Mixture of four Gaussians with $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0.25$, $\mu_1 = (0, 1), \mu_2 = (1, 0), \mu_3 = (-1, 0), \mu_4 = (0, -1)$ and $\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \mathcal{I}_2$ where Σ_k is the covariance matrix of the kth Gaussian and $\mathcal{I}_2$ is the 2×2 identity matrix. Sample 500 points from this distribution. Investigate how to sample from a Gaussian Mixture. Describe how the sampling strategy you will use work. To sample vectors you can use libraries like **numpy**. Plot the points with a scatter plot.

(b) Implement the k-means algorithm from scratch to cluster the dataset you created in (a). Plot the cluster assignments. Is the k-means able to correctly cluster the different components of the mixture? Note that you cannot use any direct implementation of the k-means algorithm, but you can use linear algebra libraries like NumPy.