Boğaziçi University
Department of Computer Engineering
Instructor: İnci Baytaş
Teaching Assistants: Berrenur Saylam, Tolga Kerimoğlu

Spring 2022
Assignment 3

CMPE 462 Project - Phase 3

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

Question 1. *Support Vector Machine*

(a) Show that the matrix Q described in the linear hard margin SVM on the lecture slides is a positive semi-definite matrix (that is $\mathbf{u}^T Q \mathbf{u} \geq 0 \quad$ for any $\quad \mathbf{u}$). The positive semi-definiteness of Q means that the QP-problem is convex. Convexity makes it easy to find an optimal solution. In fact, standard QP-solvers can solve our convex QP-problem in $O\left((N+d)^3\right)$.

(b) Consider a data set with three data points in $\mathbb{R}^2$:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -2 & 0 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix}$$

Manually solve linear hard-margin SVM (primal formulation) to get the optimal hyperplane $(b^*, \mathbf{w}^*)$ and its margin. Plot the decision boundary and denote the support vectors.

(c) Implement the hard margin SVM in (b) using Python's cvxopt library [1]. Do not use a built-in SVM function. Verify your result in (b).

(d) Train an SVM classifier for the 3D Shapes dataset. Please consider a binary classification task. You may consider two of the classes. Use your implementation in (c) with the features you obtained as the most significant in phase 2. What are the challenges you face when you want to use the cvxopt library?

(e) Train an SVM classifier using scikit-learn. Consider the same binary classification task in (d). Is it more efficient than implementing SVM using the cvxopt library? Please investigate and discuss the implementation details of the SVM function in scikit-learn.

(f) Train an SVM classifier for the multi-class classification problem you tackled in Phase 2. You may again use scikit-learn. Did you observe any performance improvement when you used the kernel trick?

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

(h) Please compare the classifiers you have trained for the 3D Shapes dataset. Which one was the best according to your experiments? Please discuss your observations.

[1] `https://cvxopt.org/`

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.