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

Spring 2022
Project-Phase 1

CMPE 462 Project - Phase 1

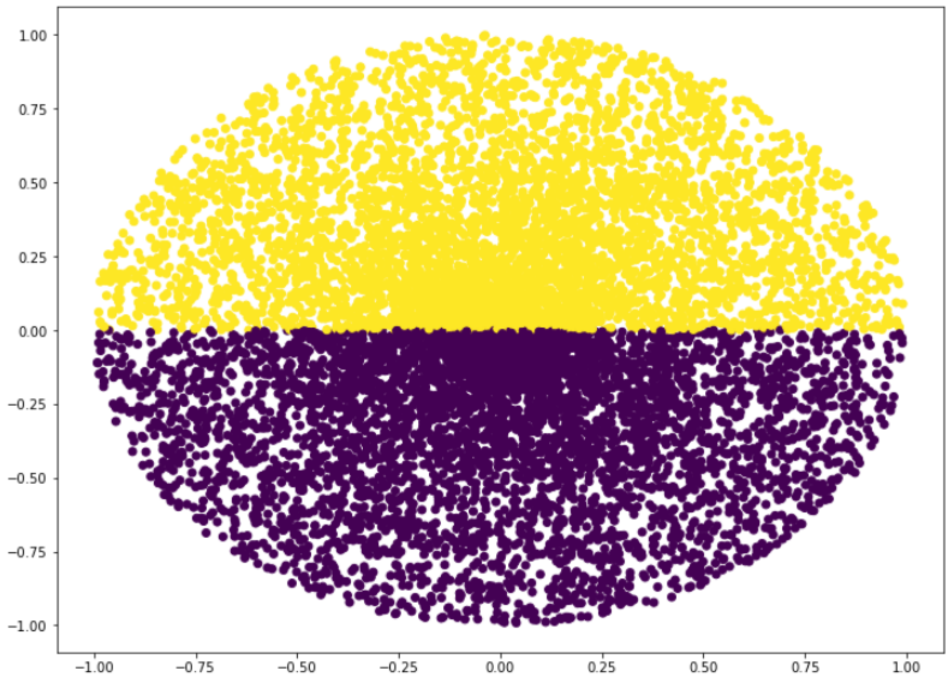
This assignment consists mostly of programming questions. Familiarity with Python, Jupyter Notebooks and relevant libraries are assumed, namely **numpy** and **matplotlib**.

For theoretical questions, we expect the answers in LaTeX. One option is to merge together your theoretical answers and code into a single Jupyter Notebook where use 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 how to reproduce the results for each question.

Question 1. *Perceptron Learning Algorithm.*

In this question you will implement the perceptron learning algorithm for a simple synthetic data set and inspect it's convergence rate. Our data set is a set of 2D-points uniformly sampled from the unit sphere where a point is labeled **+1** if it belongs to the upper hemisphere and **-1** if it's in the lower hemisphere.

a) Sample 1000 points from the data set as described. You can use the *create dataset* function in the **Helper Functions - Q1.ipynb** notebook provided. Notice that the function creates the data set in the format $x = [1, x_1, x_2]$ to account for the bias. Visualize the data set using **matplotlib.** Your output should be similar to the following:



b) Implement the Perceptron Learning Algorithm and run the algorithm for a single pass over the data set, i.e. do not consider any point more than once. Report the final w vector, and prediction accuracy over the entire data set. Additionally, visualize the decision boundary. You may use **numpy** for implementation and **matplotlib** for visualization.

c) For this question, you will theoretically explore the importance of bias with a simple example. Consider again the classification problem with the perceptron but without bias. Construct a simple sample that is linearly separable but is not separable with a perceptron without a bias. Argue formally that this is the case for your example. What geometric property (restriction) of the *no-bias* perceptron limits it? (Imagine a line in 2D or a plane in 3D without bias)

Question 2. *Linear Regression.*

This question will allow you to explore linear regression for a simple image dataset. We will use the **3D Shapes** dataset from Google's DeepMind which you will also encounter in future assignments so it is a good idea to take a look at it here.

In the dataset file you are provided, you will find 10000 images under the **3dshapes-train** folder and the **orientations-train.npy** file which contains the orientation values of the images in order. You need to use the numpy library to load this file which is simply an array of size 10000. Each image contains a shape with a particular orientation, orientation can take 15 different values, linearly spaced between [-30, 30]. These will serve as our target labels.

a) Convert the training images into gray scale, flatten each image into a vector of length 4096(64x64) and load them into a numpy array. You will obtain a matrix of size 10000x4096. You might need to use an additional library for this, e.g. **Pillow.**

b) Treating your data matrix as X and the orientations as y, we can pose this as a linear regression problem, looking for the minimizer of $\|Xw - y\|_2^2$. Apply the linear regression algorithm and compute the solution $w^\star$.

c) You are also provided with a test data set consisting of 1000 images under the **3dshapes-test** folder. Use the solution vector you obtained to make predictions on the test data. Notice that while our labels take discrete values, linear regression predicts continuous values. Report the Mean Square Error(MSE) between your predictions and target values in the **orientations-test.npy** file.

d) Using the entire image as the feature vector requires a very large data matrix which can make linear regression computationally expensive for larger data sets. Research and apply a technique of your choice from feature extraction/dimensionality reduction to obtain more compact feature vectors per image and use your new features for linear regression. Report the size of your new features and the MSE on the test data. Do not forget to cite your references.