ECEN 5712 (M. Varanasi) **Homework 3** **Due Feb. 24, 2022**

Problems of this homework require you to work with the data sets provided. For details, see the guide given here following the problems.

1. (Hard SVM for linearly separable data) In this problem, we are revisiting the data sets Classify-2D-wLabels-1.txt and Classify-2D-wLabels-2.txt. Each contains labeled two-dimensional inputs with input space $\mathcal{X} = [-1, 1]^2$ with the binary label set $\mathcal{Y} = \{0, 1\}$. Convert the labels to the $\mathcal{Y} = \{-1, 1\}$ form for this problem.

   a. Write a computer program that implements the dual formulation of the *hard*-margin SVM algorithm from scratch (that works with d-dimensional binary linearly separable labeled data). You can call a convex programming solver (or a Quadratic Programming (QP) solver) as part of your program (i.e., you don't have to implement a QP solver from scratch).

   b. Split the datasets into training and validation data. You will use the first 200 samples for training and the last 50 for validation. Train your hard-margin SVM algorithm to classify the two datasets. Report the final separating hyperplane, the empirical validation loss, and the maximum geometric margin achieved for each data set. Plot the data with the separating hyperplane (the decision boundary) and the two marginal hyperplanes.

   c. Train the Perceptron algorithm you implemented in Homework 1 on these datasets (with 200 points) and report the measurements specified in Part b. Compare the performance of these algorithms on the given classification task via the geometric margin and the empirical validation loss.

2. (Hard SVM for non-linearly separable data) Consider Classify-2D-wLabels-3.txt for this problem, which follows the same format as the datasets from the previous question. This problem focuses on using higher dimensional (polynomial) feature spaces with hard-margin SVM.

   a. In this problem, we will use the Polynomial(3) and Polynomial(4) feature spaces as defined in Homework 2. Modify your hard-margin SVM implementation from Problem 1 to one that uses these two Polynomial feature spaces. In other words, the algorithm should take the 2-dimensional data points, map them into the higher-dimensional Polynomial(3) and Polynomial(4) feature spaces and classify in those spaces.

   b. Train your hard-margin SVM algorithm using the Polynomial(3) and Polynomial(4) mappings with the first 200 training data. Report the weights you obtain and the resultant training loss. Compare the two classifiers. For the given dataset, pick the smallest of the two hypothesis classes that works (i.e., it perfectly classifies the data). Do you observe anything notable about the weights in the Polynomial(3) and Polynomial(4) classes? Use the 50 validation data points to predict labels and compare with the given labels for these data points and report validation loss (the fraction of such data points that are mis-classified), if any.

3. (Soft-margin SVM) In this problem, you will work with Classify-3DwLabels-2.txt and a new dataset Classify-3DwLabels-3.txt, which follows the same format. You are to use soft-margin SVM to linearly classify these datasets.

   a. Write a computer program that implements the *soft*-margin SVM algorithm that works with d-dimensional binary labeled data and allows the user to specify the C coefficient to tune the algorithm. Your program should solve the dual form of the soft-SVM optimization problem. Again, use an available convex programming solver (or QP solver) to accomplish this task as in Problem 1.

   b. Train your soft-margin SVM on the two datasets with parameters

   $$C \in \{0.01, 0.25, 0.5, 0.75, 1\}$$

   using all 250 data points. Report the empirical training loss (the fraction of mis-classified training data points) and the geometric margin achieved in a table comparing the five classifiers (for the five values of C) for each dataset. This information gives you a sense of how the soft-SVM algorithm trades-off training loss and geometric margin.

   c. Now train your soft SVM algorithm with just the first 200 points (training data) for each value of C in Part b. Use the last 50 input data points and predict their labels using your trained soft-SVM classifier for each of the five C values. Compute the validation loss in each case (the fraction of validation data points in which your classifier mis-classifies the data). Which value of C yields the lowest validation loss for each of the two data sets?

Programming Guide

All datasets in this homework are given as comma-separated values, where each row corresponds to a datapoint.

For this homework, you can use any programming language and linear algebra, optimization, and visualization library you would like. As easy-to-use options, we recommend Python with NumPy, SciPy and matplotlib. Below are some links for this setup that might be helpful. You are not allowed to use scikit-learn or any equivalent high-level machine learning library. If you are unsure a library you want to use might fall in this category, please ask about the library and the function(s) you are planning to use on Piazza.

1. As a hint for Question 2.b, you can revisit how scikit-learn's Pipeline mechanism handles inserting polynomial features into the learning algorithm. You can't use the scikit-learn functions directly, but you can use the idea presented here in your own implementation.

2. The following libraries may or may not be useful for implementing your SVM algorithms depending on your design.

   (a) NumPy is your go-to for linear algebra related functions.

   (b) SciPy.optimize has easy-to-use optimization functions. Alternatively, you can use PuLP which is slightly harder to use, but offers more flexibility and control.

3. For plotting decision boundaries and marginal planes, you can use matplotlib's contour method.

Please make sure the code you submit runs stand-alone. If you are submitting multiple files that contain dependencies to each other, make sure the file/folder structure in your submission is the same as your setup. We should be able to run and evaluate your code without needing any reverse engineering. If you think it is not obvious, please include comments on how you expect your code to be executed or any specific versions of languages/libraries if that matters. For any concerns about submission format or any programming-related issues, feel free post your questions on Piazza.