Homework 4 – Machine Learning (CS453X, Whitehill, Spring 2019)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **Support Vector Machines: Implementation [40 points]**: In this problem you will implement a class called `SVM453X` that supports both training and testing of a linear, hard-margin support vector machine (SVM). In particular, you should flesh out the two methods `fit` and `predict` that have the same API as the other machine learning tools in the `sklearn` package.

   (a) `fit`: Given a matrix $\mathbf{X}$ consisting of n rows (examples) by m columns (features)[1] as well as a vector $\mathbf{y} \in \{-1, 1\}^n$ consisting of labels, optimize the hyperplane normal vector $\mathbf{w}$ and bias term b to maximize the margin between the two classes. To do so, you should harness an off-the-shelf quadratic programming (QP) solver from the `cvxopt` Python package. The template file already includes code showing how to use it – just pass in the relevant matrices and vectors as `numpy` arrays. The trick is in setting up the variables to encode the SVM objective and constraints correctly. Then, store the optimized values in `self.w` and `self.b` because you'll need them later.

   (b) `predict`: Given a matrix $\mathbf{X}$ of test examples, and given the pre-trained `self.w` and `self.b`, compute and return the corresponding test labels.

   Your code will be evaluated based both on the accuracy of the predicted test labels in several tests (you can see the exact test code we'll be running in the template file), as well as the difference between the hyperplane parameters compared to their ideal values.

   Note: as shown in the template, you can calculate the ideal values using the off-the-shelf `sklearn` SVM solver. Note that, since the SVM in this package is, by default, a soft-margin SVM, we have to "force" it to be hard-margin by making the cost penalty C very large.

2. **Support Vector Machines: Application [30 points]**: Register for the following Kaggle competition about predicting which bank customers will make a transaction: `https://www.kaggle.com/c/santander-customer-transaction-prediction` Read the Overview section to learn what the competition is about. Download the training data and associated labels from the Data tab. Train SVMs using two different kernels: linear and polynomial (I suggest degree 3, but it's your choice). Use `sklearn`'s implementation of the `SVC` class. Use half the training data (randomly selected from the entire training set) for training, and use the remaining half to estimate the performance of your classifiers. You are **not** required to submit your guesses for the test examples; however, if you want to, you can earn a spot on the competition's leaderboard. **Important note**: You should compute the SVM's predictions using `svm.decision_function`, not `svm.predict`. The former gives $\mathbf{w}^\top \mathbf{x} + b$, i.e., a real number, whereas the latter gives $\text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$, i.e., a binary number. When using AUC as the accuracy metric, we typically use real-valued predictions because they have more information.

   Bootstrap aggregation (bagging): In order to train an SVM with a non-linear kernel on a dataset of this size (the whole training dataset contains 200K examples; hence, your training data will consist of 100K examples) on an ordinary desktop/laptop computer (i.e., without terabytes of RAM), you will need to use an approach such as **bagging**: (a) split the data into multiple subsets; (b) train an SVM on each subset; and (c) create an ensemble classifier that averages the predictions of the individual classifiers in the ensemble. Report your estimated accuracy (AUC) using both your SVMs; use the `sklearn.metrics.roc_auc_score` function. You can just include the accuracy as a comment at the top of your Python file. We might run your code to reproduce this result; hence, make sure you only use `sklearn`, `pandas`, and `numpy` – no other packages.

Submit your Python code in a file called `homework4_WPIUSERNAME1.py` (or `homework4_WPIUSERNAME1_WPIUSERNAME2.py` for teams).

[1] To be consistent with the notation I've used in the lecture slides, $\mathbf{X}$ in this assignment would actually need to be called $\mathbf{X}^\top$. The reason I call it $\mathbf{X}$ here is to be consistent with the `sklearn` package.