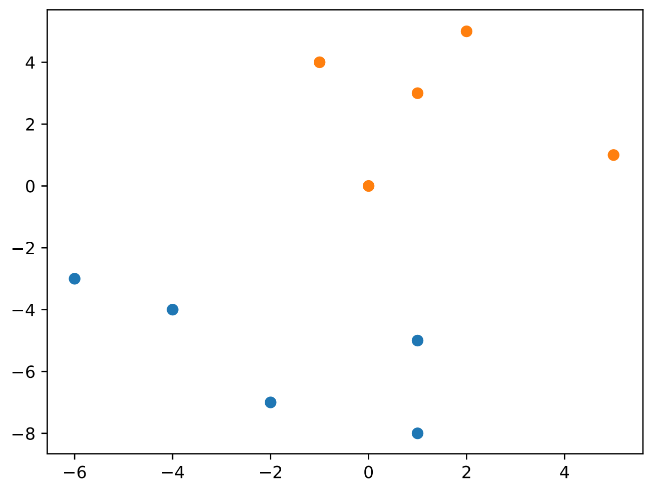
Homework 4 – Machine Learning (CS4342, Whitehill, Spring 2022)

You may do this homework either by yourself or in a team of 2 people.

1. **Separating hyperplanes and margins [15 points]**: Consider a set of data $\{\mathbf{x}^{(i)}\}_{i=1}^{10}$ (of which 5 are negatively and 5 are positively labeled), as shown in the scatter plot below.



(The examples $\mathbf{X}$ and associated labels $\mathbf{y}$ are available on Canvas as `hw4_X.npy` and `hw4_y.npy`. Note that these data points all happen to have integer-valued coordinates; this is just for convenience and is generally not the case!) There are infinitely many separating hyperplanes H that will perfectly separate these data (i.e., all of the negatively labeled points are on one side, and all the positively labeled points are on the other side).

For a hyperplane H that perfectly separates the data, we can form two hyperplanes H^- and H^+ that are both parallel to H such that H^- contains all the *negatively* labeled data closest to H, and H^+ contains all the *positively* labeled data closest to H. (Note that the set of points "closest to H" will depend on what H is.) We can then define the **margin** as the distance between H^- and H^+ (see Lecture #13, slide #35). Let us consider only those H that are equidistant to (i.e., halfway between) both H^- and H^+; in that case, there is a unique hyperplane H with maximum margin.

Consider the three different hyperplanes described below:

- The hyperplane "normal to" (perpendicular to) the vector $[0, 1]^\top$ (i.e., H is a horizontal line) that perfectly separates the data.
- The hyperplane normal to $[-0.3, 1]^\top$ that perfectly separates the data.
- The hyperplane with maximum margin (which you should obtain by training an SVM on the dataset above).

For each hyperplane H above, (a) find $\mathbf{w}$ and b that describe the hyperplane so that $H = \{\mathbf{x} : \mathbf{x}^\top\mathbf{w} + b = 0\}$, $H^+ = \{\mathbf{x} : \mathbf{x}^\top\mathbf{w} + b = +1\}$, and $H^- = \{\mathbf{x} : \mathbf{x}^\top\mathbf{w} + b = -1\}$. Next, (b) plot H, H^+, and H^- on top of a scatter plot like the one shown above (combine all three choices of H onto a single scatter plot); and (c) calculate the margin of H. Put all your answers to (a), (b), and (c) for each H into the PDF file. Make sure that you show (and submit) your work, i.e., include Python code and/or manual calculations of how you obtained your answers. See the starter code `homework4_geometry_template.py`.

Hint: on this problem, to determine the set of points closest to each H, it is fine to plot a hyperplane with a specified normal vector (i.e., $\mathbf{w}$) on top of the scatter plot, try moving it up/down, and manually see which points it intersects. Once you know the points closest to H, you can then use algebra to solve for $\mathbf{w}$ and b.

2. **Support Vector Machines: Training** [**40 points**]: In this problem you will implement a class called `SVM4342` 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, subject to the constraints that each data point be on the correct side of the hyperplane (hard-margin SVM). To do so, you should harness an off-the-shelf quadratic programming (QP) solver from the `cvxopt` Python package (see `https://cvxopt.org/install/`; it is also available on Google Colab). 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, as described in lecture. 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.

   See the starter code `homework4_template.py`.

Combine both your Python file(s) and the PDF in a single Zip file, and then submit the Zip file on Canvas. If you are working as part of a team, then make sure you register as one of the pre-allocated teams on Canvas; only one person should then submit the Zip file (and you will both receive credit).

[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.