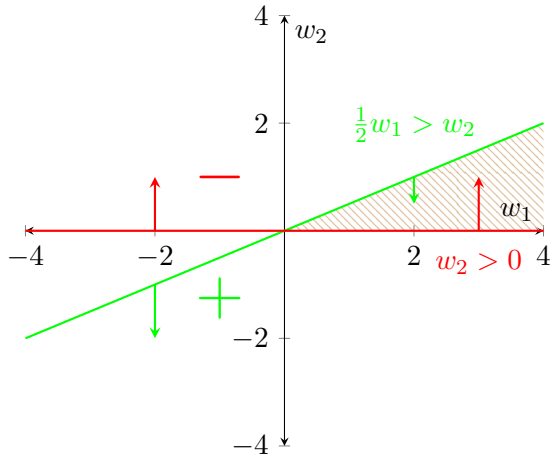
CSC 321 Homework 2

Woosung Hwang
Student ID : 1003709481

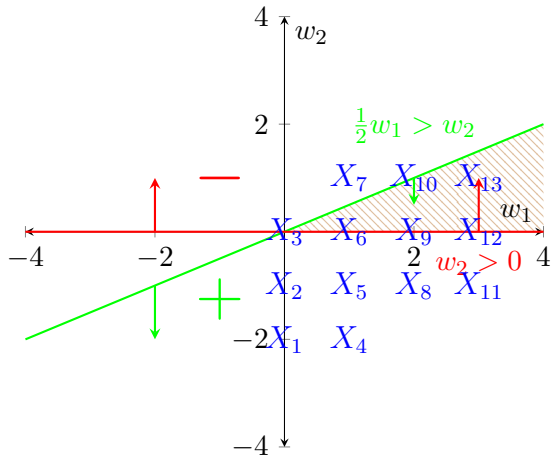
1. Perceptron Algorithm

a. Sketch out the problem in weight space. In particular: draw and label the axes, draw the half-space corresponding to each of the two training examples, and shade the feasible region. (Remember that there is no bias term.)



- The feasible region is highlighted with brown line shade.

b. Simulate the perceptron algorithm by hand, starting from the initial weights $w_1 = 0$, w_2 = -2. On the plot from part (a), mark an X on every point in weight space visited by the algorithm until it stops. You do not need to provide anything for this part other than the X's on the plot.



i. $w_1 = 0, w_2 = -2$
ii. $w_1 = 0, w_2 = -1$
iii. $w_1 = 0, w_2 = 0$
iv. $w_1 = 1, w_2 = -2$

v. $w_1 = 1, w_2 = -1$
vi. $w_1 = 1, w_2 = 0$
vii. $w_1 = 1, w_2 = 1$
viii. $w_1 = 2, w_2 = -1$
ix. $w_1 = 2, w_2 = 0$
x. $w_1 = 2, w_2 = 1$
xi. $w_1 = 3, w_2 = -1$
xii. $w_1 = 3, w_2 = 0$
xiii. $w_1 = 3, w_2 = 1$

2. Feature Maps

x	t
-1	1
1	0
3	1

a. Argue briefly (at most a few sentences) that this dataset is not linearly separable.

- Since the positive region is convex, if we draw the line segment connecting the two positive examples (-1) and (3), this entire line segment must be classified as positive. Similarly, if we point the dot negative examples (1), the entire line segment (It can be the line segment by connecting two same exmaples (i.e. (1), (1))) must be classified as negative. But this (1) point must be classified as both positive and negative, which is impossible. *Therefore, this dataset is not linearly separable.*

b. Write down the constraint on w_1 and w_2 corresponding to each training example, and then find a pair of values (w_1,w_2) that correctly classifies all the examples. Remember that there is no bias term.

-The constraints on w_1 and w_2.

i. $-w_1 + w_2 > 0$
ii. $w_1 + w_2 < 0$
iii. $3w_1 + 9w_2 > 0$

The pair of values (w_1,w_2) that correctly classifies all the examples.

$$w_1 = -1 \quad w_2 = 0.5$$

3. Loss Functions

Suppose we have a prediction problem where the target t corresponds to an angle, measured in radians. A reasonable loss function we might use is

$$\mathcal{L}(y, t) = 1 - cos(y - t).$$

Suppose we make predictions using a linear model,

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b.$$

Derive a sequence of vectorized mathematical expressions for the gradients of the cost with respect to $\mathbf{w}$ and b. As usual, the inputs are organized into a design matrix $\mathbf{X}$ with one row per training example. The expressions should be something you can translate into a Python program without requiring a for-loop.

$$
\begin{aligned}
\mathbf{y} &= \mathbf{X}\mathbf{w} + b\mathbf{1} \\
\frac{\partial \mathcal{E}}{\partial \mathbf{y}} &= \frac{1}{N} sin(\mathbf{y} - \mathbf{t}) \\
\frac{\partial \mathcal{E}}{\partial \mathbf{w}} &= \frac{1}{N} \mathbf{X}^{\mathrm{T}} sin(\mathbf{X}\mathbf{w} + b\mathbf{1} - \mathbf{t}) \\
\frac{\partial \mathcal{E}}{\partial b} &= \frac{1}{N} \mathbf{1}^{\mathrm{T}} sin(\mathbf{X}\mathbf{w} + b\mathbf{1} - \mathbf{t})
\end{aligned}
$$

- The input $\mathbf{X}$ is N $\times$ D matrix having N training examples, each D-dimensional. The weights are represented as a D-dimensional vector $\mathbf{w}$ (D $\times$ 1), and the targets are represented as a N-dimensional vector $\mathbf{t}$ (N $\times$ 1). Finally $\mathbf{1}$ denotes a vector of all ones (N $\times$ 1).