Exercise 2: Perceptron Questions (5 pts)

1. (3 pts) The perceptron algorithm makes an update every time it witnesses a mistake. What if it makes an update on every point, regardless of whether or not that point is correctly classified? For simplicity, consider a setting where where b is fixed to 0. Give an example of an infinite sequence of points (x_i, y_i) with the following properties:
   - I The sequence is strictly linearly separable with $b = 0$ (i.e., the margin is some constant $\gamma > 0$) ,
   - II The sequence has max $\|x_i\|_2 \leq 1$,
   - III The modified perceptron algorithm makes an infinite number of mistakes on this sequence .

   Prove that it has these properties. Note that the perceptron convergence theorem and the first two conditions imply that, at some point, the unmodified perceptron algorithm would only make a finite number of mistakes.

   Ans: **Answer 2.1**
2. (1 pt) Give examples of where the perceptron algorithm converges to a 0 margin halfspace, and a separate example where it converges to a maximum margin halfspace. As pointed out by some on Piazza, tech- nically, if a halfspace has 0 margin, then it would misclassify anything that still lies on the hyperplane, and the perceptron algorithm would not yet have halted. If you came up with an example that ignores these cases and halts with a point on the hyperplane, that's fine. However, the intent was more like the following, so consider solving the problem where it converges to an arbitrarily small margin halfspace. More precisely: for any $0 < \varepsilon < 1/2$, give a dataset (with margin at least 1) and a order in which to process the points such that the perceptron algorithm halts providing a halfspace with margin $\leq \varepsilon$. This problem has to do with the original perceptron, not the modified perceptron from part 1.

   Ans: **Answer 2.2**
3. (1 pt) Suppose that in each iteration of the perceptron algorithm, a point is chosen uniformly at random from the dataset. Show how the perceptron algorithm can be viewed as an instantiation of stochastic gradient descent (SGD). In particular, you must provide a loss function and a learning rate such that SGD with these parameters and perceptron are identical.

   Ans: **Answer 2.3**

Answer 2.3: SGD

The goal for Stochastic Gradient Descent (SGD) is to learn a model to minimize the loss $\ell(\mathbf{w};\mathbf{x}_i,y_i)$ over a training set $\mathscr{D} = (\mathbf{x}_i,y_i)$. Similarly, we can consider the minimization is related to a linear mode with form of $f_w(x) = \langle \mathbf{w},x\rangle$. Hence:

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w};\mathbf{x}_i,y_i) = \min_w \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i) \tag{32}$$

Then SGD basically updates the weight vector $\mathbf{w}_t$ by performing an approximate gradient descent:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w};\mathbf{x}_i,y_i) \tag{33}$$

Recall the perceptron update:

$$\text{if} \quad y_i(\langle \mathbf{w}_i,\mathbf{x}_i\rangle + b_i) \le 0 \Rightarrow (y_i\langle \mathbf{w}_i,\mathbf{x}_i\rangle + y_i b_i) \le 0 \tag{34}$$

$$\text{then} \quad \mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + y_i\mathbf{x}_i \tag{35}$$

$$b_{i+1} \leftarrow b_i + y_i \tag{36}$$

As Equation (34) stated, when $y(\langle \mathbf{w},\mathbf{x}\rangle + b)$ is larger than zero, the system does not update. It updates only when $y(\langle \mathbf{w},\mathbf{x}\rangle + b)$ is less than zero (when there is a mistake).

Hence, it is equivalent to update when the $-y(\langle \mathbf{w},\mathbf{x}\rangle + b)$ is larger than zero. This can be the loss function we try to minimize for when the term is larger than zero.

Hence, we may assume the loss function as:

$$\ell(\mathbf{w};\mathbf{x}_i,y_i) = \max(0,-y_i\langle \mathbf{w},\mathbf{x}_i\rangle - y_i b) \tag{37}$$

Conveniently, this result a gradient:

$$\nabla_{\mathbf{w},b}\ell(\mathbf{w};\mathbf{x}_i,b;y_i) = \frac{\partial \max(0,-y_i\langle \mathbf{w},\mathbf{x}_i\rangle - y_i b)}{\partial \mathbf{w}} \tag{38}$$

We may also separate them into to two distinct components for w and b:

$$\nabla_{\mathbf{w}}\ell(\mathbf{w};\mathbf{x}_i,y_i) = \max(0,-y_i\mathbf{x}_i) \tag{39}$$

$$\nabla_b\ell(b;y_i) = \max(0,-y_i) \tag{40}$$

Since the data at each iteration 't' of SGD is randomly selected from the dataset $\mathscr{D}$, hence we may replace $(y_i,\mathbf{x}_i)$ with sampled dataset $(y_t,\mathbf{x}_t)$, which is characterized in a Stochastic GD instead of the GD method.

As a result, the SGD update function (Equation (33)) becomes:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}}\ell(\mathbf{w};\mathbf{x}_i,y_i) = \mathbf{w}_t - \eta_t \max(0,-y_i\mathbf{x}_i) = \max(\mathbf{w}_t,\mathbf{w}_t + \eta_t y_i\mathbf{x}_i) \tag{41}$$

$$b_{t+1} \leftarrow b_t - \eta_t \nabla_b\ell(b;y_i) = b_t - \eta_t \max(0,-y_i) = \max(b_t,b_t + \eta_t y_i) \tag{42}$$

As Equation (41) and stated, the SGD formulation stated above returns exact result as the update function for the weight in the perceptual algorithm (Equation (35) and Equation (36) respectively), when the learning rate is the step size $\eta_t = 1$. It either maintain the current $\mathbf{w}_t$ and b_t, or update towards a better location to minimize the loss function (in another word, when it makes a mistake).

In conclusion, we may conclude the perceptron algorithm can be viewed as an instantiation of stochastic gradient descent (SGD), with:

- a loss function: $\ell(\mathbf{w}; \mathbf{x}_i, y_i) = \max(0, -y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i b)$
- a learning rate: $\eta_t = 1$