PERCEPTRON ALGORITHM

KEY POINT :

↳ The Perceptron Algorithm only works when your data is linearly seperable.

INTRODUCTION :

Consider n sample points $x_1, x_2, \ldots, x_n$
For each sample point, we have a label $y_i = \begin{cases} 1 & \text{in class } c \\ -1 & \text{not in class } c \end{cases}$

We want a weight vector such that :

$$x_i \cdot w > 0 \quad \text{when } y_i = 1$$
$$x_i \cdot w < 0 \quad \text{when } y_i = -1$$

Not accounting for bias term here.

LOSS FUNCTION :

$$L(x) = \begin{cases} 0 & \text{if } y_i(x_i \cdot w) > 0 \\ -y_i(x_i \cdot w) & \text{otherwise} \end{cases}$$

this is a positive term

RISK FUNCTION :

average of the loss function

$$R(x) = \frac{1}{n} \sum_{i \in V} -y_i (x_i \cdot w)$$

↳ set of all misclassified points

→ Now our goal is to find a weight vector w, that minimizes the Risk Function. We solve this using gradient descent

↳ Covered in detail under optimization

Once you find that weight vector you can create a hyperplane (decision boundary) that is orthogonal to the weight vector.

Now let us look at the case where we have a bias term :

$$f(x) = w \cdot x + \alpha = \begin{bmatrix} w_1 & w_2 & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

this produces a hyperplane in d+1 dim that passes through origin

Run perceptron Algorithm in (d+1) dim space.

Now we have sample points in $\mathbb{R}^d$, all lying on hyperplane $x_{d+1} = 1$