Gradient Descent Algorithm:

1. start with random weights $w_1, \dots, w_n, b$
2. For every point $(x_1, \dots, x_n)$:

   For $i = 1 \dots n$:

   update $\hat{\omega}_i \leftarrow w_i - \alpha(\hat{y} - y)x_i$

   update $b' \leftarrow b - \alpha(\hat{y} - y)$
3. Repeat until error is small → #epochs

« Summary »

sigmoid activation function $\sigma(x) = \frac{1}{1 + e^{-x}}$

output (prediction) formula $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

Error function $\text{Error}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log (y - \hat{y})$

Updating weights $w_i \to w_i + \alpha(y - \hat{y})x_i$

$b \to b + \alpha(y - \hat{y})$

Perceptron vs. Gradient Descent

in GDA: change w_i to $w_i + \alpha(y - \hat{y})x_i$ every time

in PA: only update if misclassified

In fact, they are basically the same!

in GDA: both of come closer & go further away

in PA: only come closer! or do nothing