Gradient Descent Algorithm:

1. Start with random weights $w_1, \ldots, w_n, b$
2. For every point $(x_1, \ldots, x_n)$:
   
   For $i = 1 \ldots n$:
   
   update $\hat{w}_i \leftarrow w_i - \alpha (\hat{y} - y)x_i$
   
   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 $\text{Error} (y, \hat{y}) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y})$

Updating weights $\omega_i \rightarrow \omega_i + \alpha (y - \hat{y})x_i$

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