Perceptron:

\[ g(z) = \text{if } z \geq 0 \text{ then } 1 \text{ else } 0 \]

Logistic Regression:

\[ g(z) = \frac{1}{1+e^{-z}} \]
**Perceptron Learning Algorithm**

Initialize $w_0, ..., w_d$

Until <stopping condition>

Randomly order the data $<x^1, ..., x^N>$

For $j = 1$ to $N$ do

if $w \cdot x^j \geq 0$ then $h=1$ else $h=0$

For $i = 0$ to $d$ do

$w_i = w_i + \alpha \times (y^j - h) \times x^j_i$

Derived by Rosenblatt based on McCulloch/Pitts model of neurons, Hebbian model of learning (not covered in class)

---

**Logistic Regression Learning Algorithm**

Initialize $w_0, ..., w_d$

Until <stopping condition>

Randomly order the data $<x^1, ..., x^N>$

For $j = 1$ to $N$ do

$h = \frac{1}{1 + e^{-w \cdot x^j}}$

For $i = 0$ to $d$ do

$w_i = w_i + \alpha \times (y^j - h) \times h \times (1 - h) \times x^j_i$

Derived formula for gradient descent – minimizing

$$Loss_D(h_w) = \sum_{j=1}^{N} \left( y^j - h_w(x^j) \right)^2$$

and then doing updates example by example (Stochastic Gradient Descent)
Gradient Descent Learning Algorithm
(Not really used)

Initialize $w_0, \ldots, w_d$
Until <stopping condition>
Randomly order the data $<x^1, \ldots, x^N>$
$<\text{temp}_0, \ldots, \text{temp}_d> = <w_0, \ldots, w_d>$
For $i = 0$ to $d$ do
$h = \frac{1}{1 + e^{-w \cdot x^j}}$
For $j = 1$ to $N$ do
$\text{temp}_i = \text{temp}_i + \alpha \times (y^j - h) \times h \times (1 - h) \times x^j_i$
$<w_0, \ldots, w_d> = <\text{temp}_0, \ldots, \text{temp}_d>$

Logistic Regression Learning Algorithm
(Stochastic Gradient Descent)

Initialize $w_0, \ldots, w_d$
Until <stopping condition>
Randomly order the data $<x^1, \ldots, x^N>$
For $i = 0$ to $d$ do
$h = \frac{1}{1 + e^{-w \cdot x^j}}$
For $j = 1$ to $N$ do
$w_i = w_i + \alpha \times (y^j - h) \times h \times (1 - h) \times x^j_i$