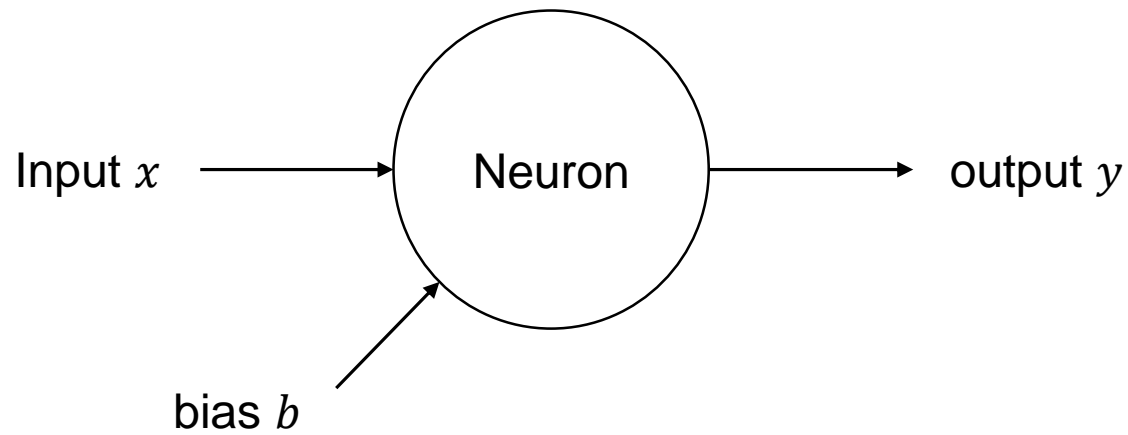


# Single Neuron Training

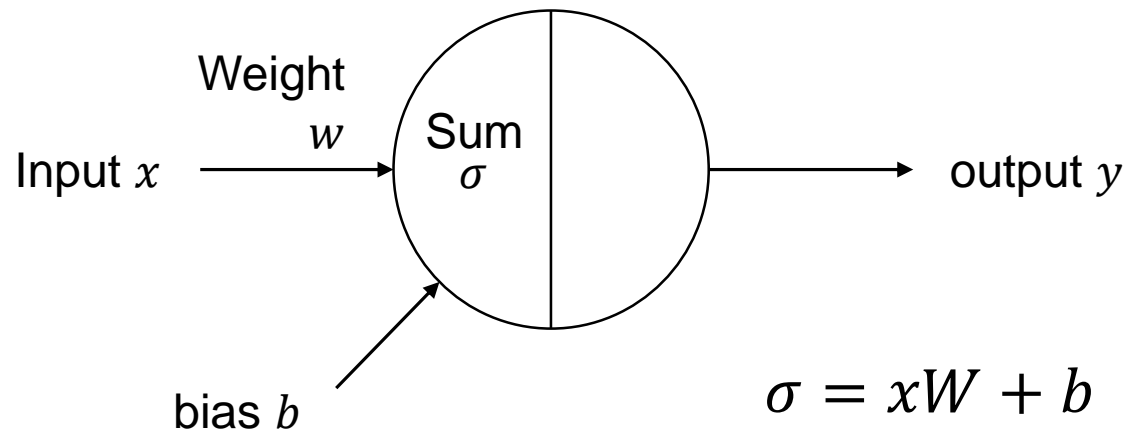
---

Youngtaek Hong, PhD

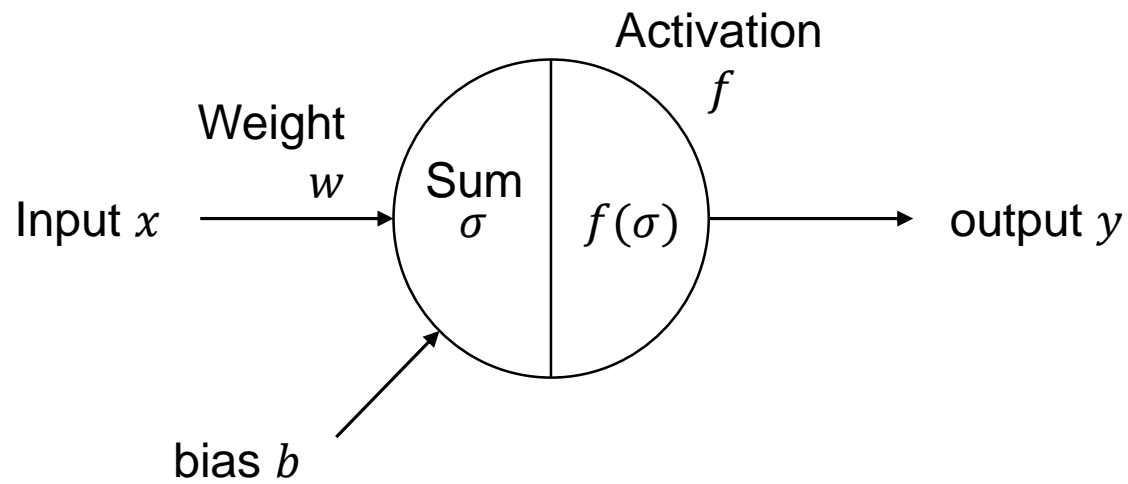
# Artificial Neuron



# sigma

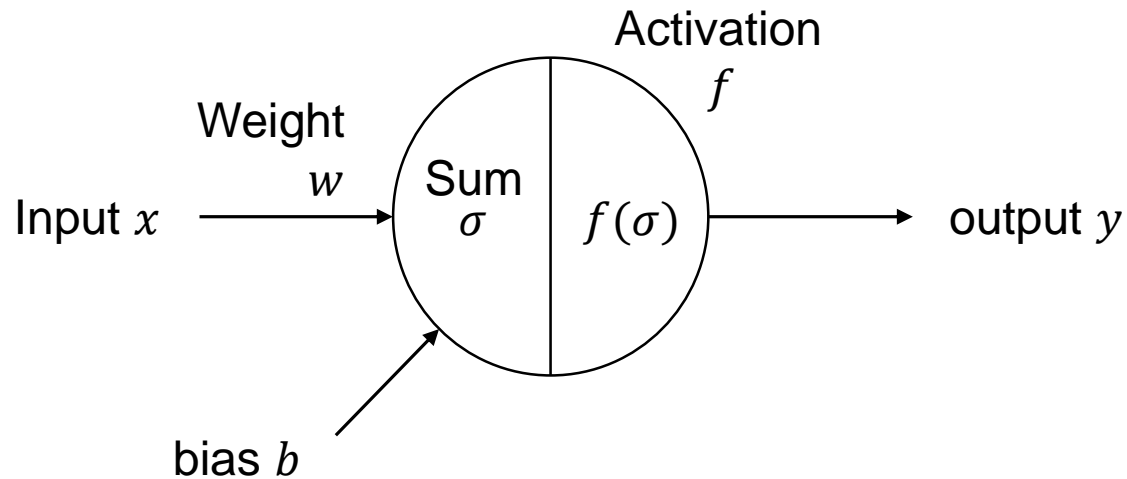
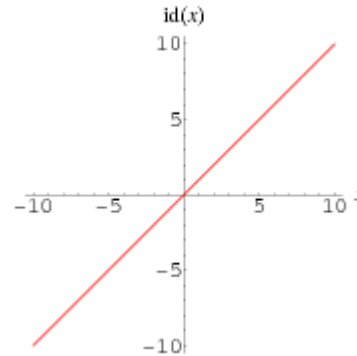


# Activation

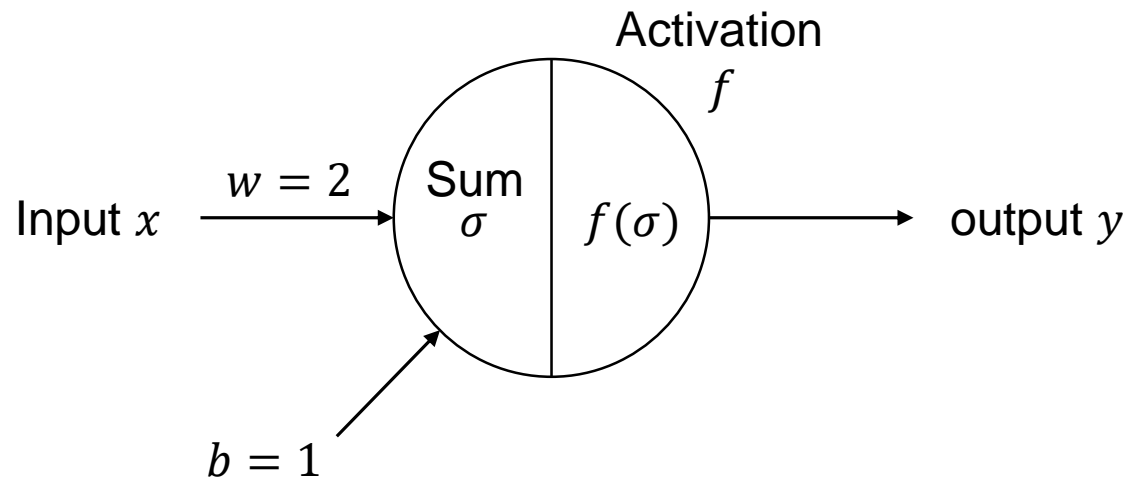


# Feed forward

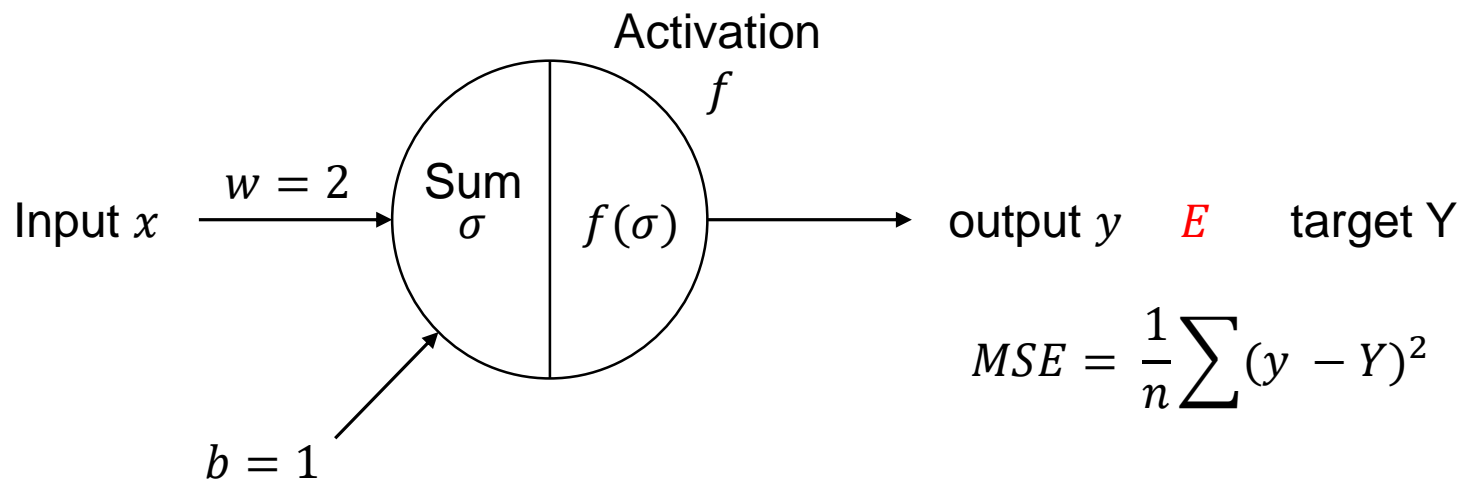
Activation: identity(= linear)



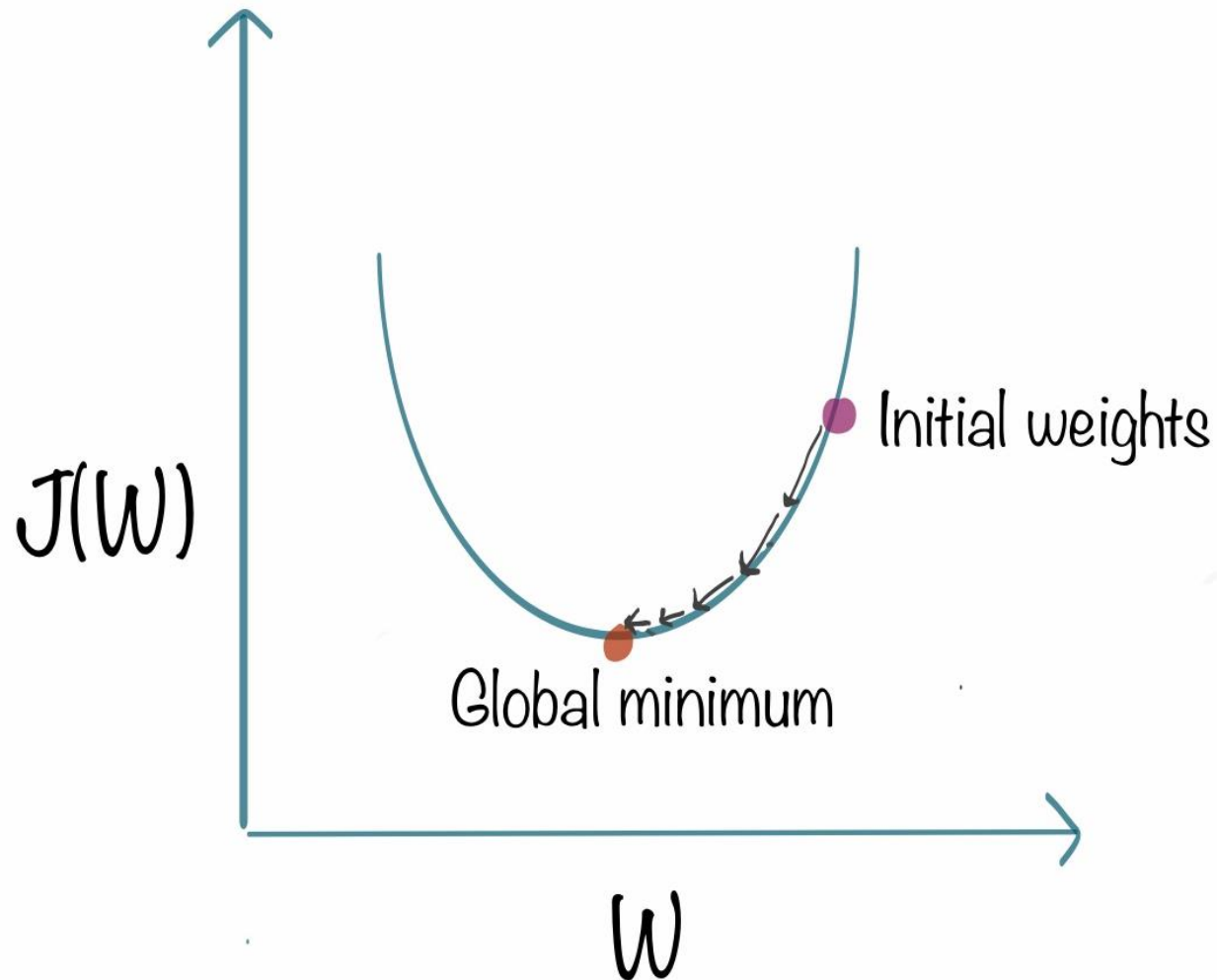
# Feed forward



# Feed forward

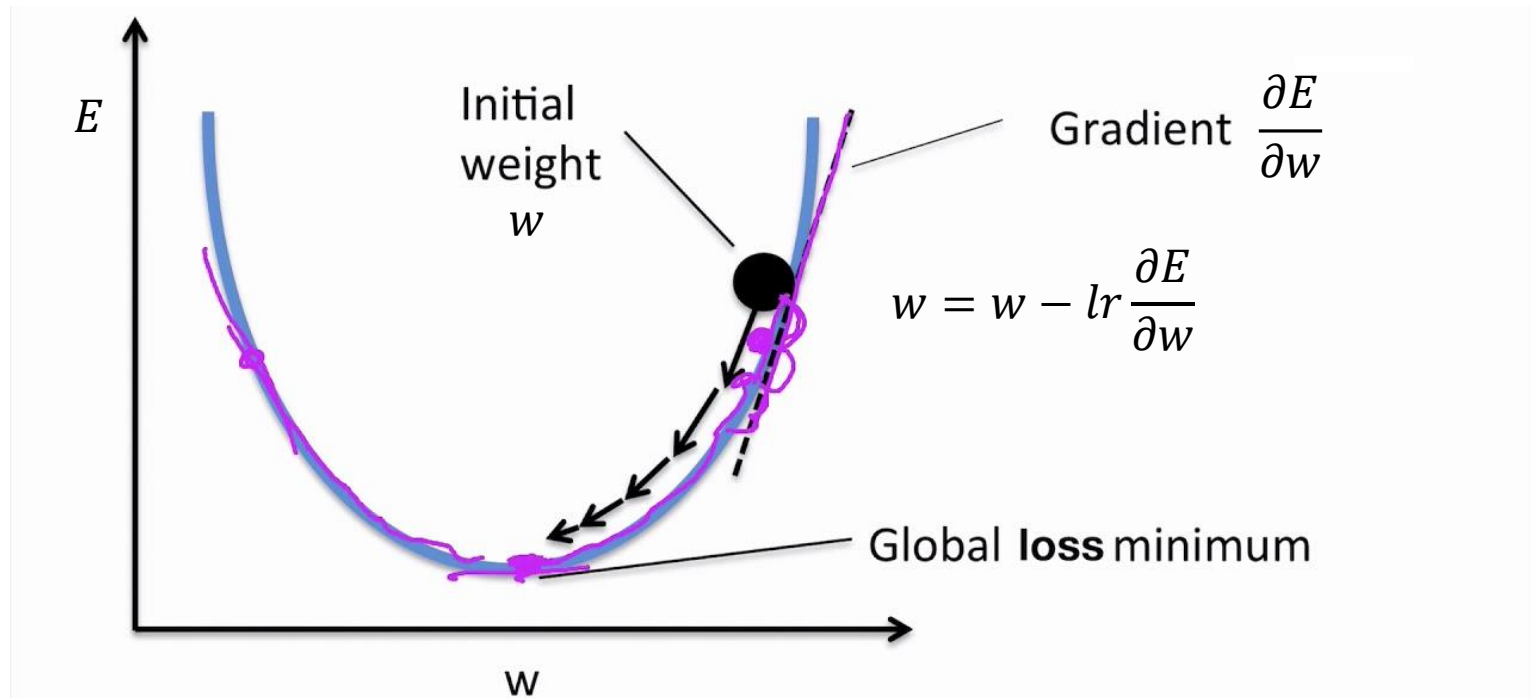


# Gradient Descent Method

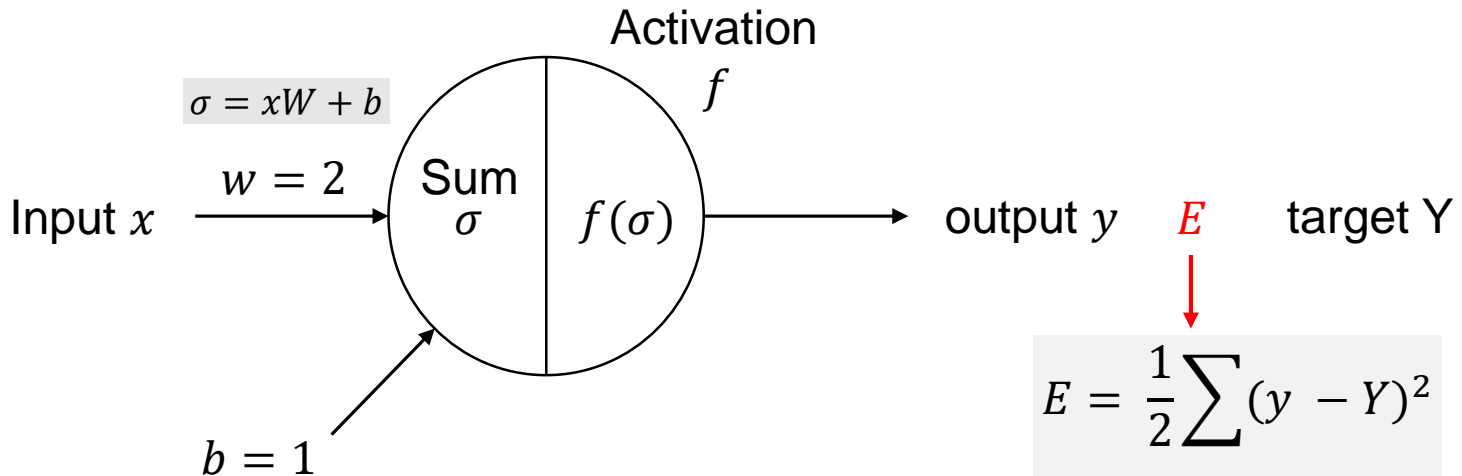




# Gradient Descent Method



# How to control E by adjusting w

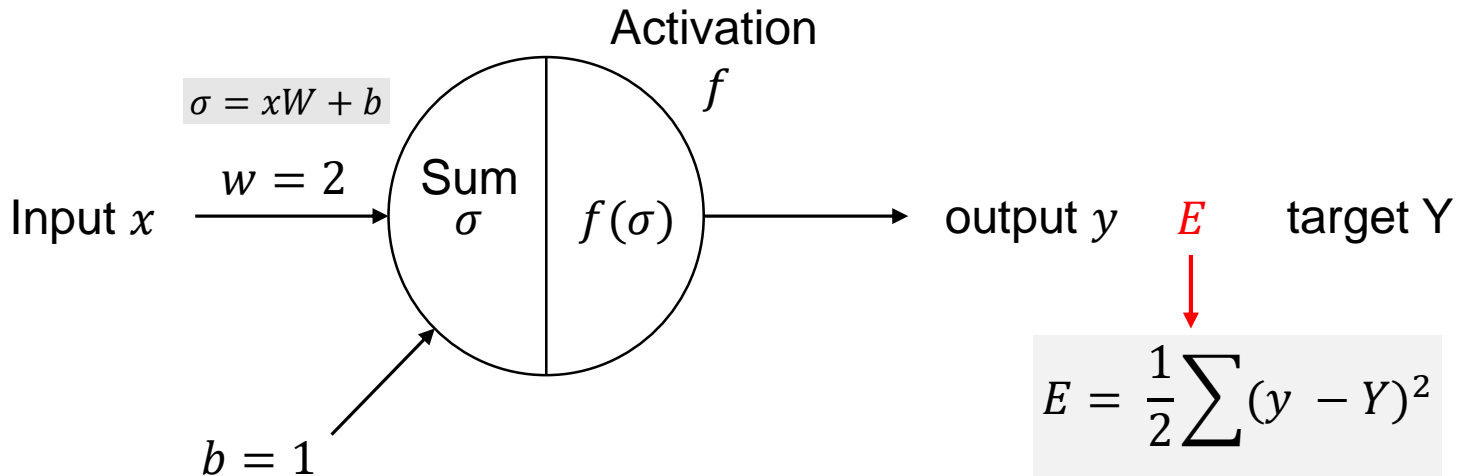


$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial f} \cdot \frac{\partial f}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial w}$$

$$\frac{\partial E}{\partial y} = (y - Y) \quad \frac{\partial y}{\partial f} = 1 \quad \frac{\partial f}{\partial \sigma} = 1 \quad \frac{\partial \sigma}{\partial w} = x$$

$$\frac{\partial E}{\partial w} = (y - Y) \times x$$

# How to control E by adjusting b



$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial f} \cdot \frac{\partial f}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial b}$$

$$\frac{\partial E}{\partial y} = (y - Y) \quad \frac{\partial y}{\partial f} = 1 \quad \frac{\partial f}{\partial \sigma} = 1 \quad \frac{\partial \sigma}{\partial b} = 1$$

$$\frac{\partial E}{\partial b} = (y - Y)$$

# Neuron class

```
[1] import tensorflow as tf

[2] class Neuron(object):

    def __init__(self, w, b):
        self.w = tf.Variable(w, name='weight')
        self.b = tf.Variable(b, name='bias')
        self.output = 0

        self.input = tf.placeholder(tf.float32, shape=[1], name="input")
        self.target = tf.placeholder(tf.float32, shape=[1], name="output")

    # Linear function = identity function
    def getActivation(self, x):
        return x

    def getActivationGradient(self, x):
        return 1.0

    def propBackWard(self):
        lr = 0.01

        grad = (self.output - self.target) * 1.0 * self.getActivationGradient(self.output)
        self.w = self.w - (lr * grad * self.input)
        self.b = self.b - (lr * grad * 1.0)

        return self.feedforward()

    def feedforward(self):
        sigma = self.w * self.input + self.b
        self.output = self.getActivation(sigma)
        return self.output
```

# Neuron class

```
[3] my_neuron = Neuron(w = 2.0, b = 1.0)
```

```
x = [1.0]
```

```
y = [4.0]
```




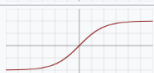
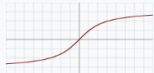





```
[4] with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print (sess.run(my_neuron.feedforward(),
                    feed_dict={my_neuron.input: x,
                               my_neuron.target:y}))

    for i in range(150):
        print (sess.run(my_neuron.propBackWard(),
                        feed_dict={my_neuron.input: x,
                                   my_neuron.target:y}))
    print (sess.run([my_neuron.w, my_neuron.b],
                    feed_dict={my_neuron.input: x,
                               my_neuron.target:y}))
```

```
[3.]  
[3.02]  
[array([2.01], dtype=float32), array([1.01], dtype=float32)]  
[3.0396]  
[array([2.0198], dtype=float32), array([1.0198], dtype=float32)]  
[3.0588078]  
[array([2.029404], dtype=float32), array([1.0294039], dtype=float32)]  
[3.0776315]  
[array([2.0388157], dtype=float32), array([1.0388159], dtype=float32)]  
[3.0960789]  
[array([2.0480394], dtype=float32), array([1.0480396], dtype=float32)]  
[3.1141572]  
[array([2.0570786], dtype=float32), array([1.0570787], dtype=float32)]  
[3.131874]  
[array([2.065937], dtype=float32), array([1.0659372], dtype=float32)]  
[3.1492367]  
[array([2.0746183], dtype=float32), array([1.0746185], dtype=float32)]  
[3.1662521]  
[array([2.083126], dtype=float32), array([1.0831261], dtype=float32)]
```

```
[3.9465704]  
[array([2.4732854], dtype=float32), array([1.4732851], dtype=float32)]  
[3.947639]  
[array([2.4738197], dtype=float32), array([1.4738194], dtype=float32)]  
[3.9486861]  
[array([2.4743433], dtype=float32), array([1.474343], dtype=float32)]  
[3.9497125]  
[array([2.4748564], dtype=float32), array([1.4748561], dtype=float32)]  
[3.9507182]  
[array([2.4753592], dtype=float32), array([1.475359], dtype=float32)]
```

# Activation functions

Name	Plot	Equation	Derivative (with respect to $x$ )	Range	Order of continuity
Identity					
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	$C^{-1}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ <sup>[1]</sup>	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	$C^\infty$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	$C^\infty$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	$C^\infty$
ASinH		$f(x) = \sinh^{-1}(x) = \ln(x + \sqrt{x^2 + 1})$	$f'(x) = \frac{1}{\sqrt{x^2 + 1}}$	$(-\infty, \infty)$	$C^\infty$
ElliotSig <sup>[8][9][10]</sup> Softsign <sup>[11][12]</sup>		$f(x) = \frac{x}{1 +  x }$	$f'(x) = \frac{1}{(1 +  x )^2}$	$(-1, 1)$	$C^1$
Inverse square root unit (ISRU) <sup>[13]</sup>		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left( \frac{1}{\sqrt{1 + \alpha x^2}} \right)^3$	$(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}})$	$C^\infty$
Inverse square root linear unit (ISRLU) <sup>[13]</sup>		$f(x) = \begin{cases} \frac{x}{\sqrt{1 + \alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left( \frac{1}{\sqrt{1 + \alpha x^2}} \right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\frac{1}{\sqrt{\alpha}}, \infty)$	$C^2$
Square Nonlinearity (SQNL) <sup>[10]</sup>		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$	$(-1, 1)$	$C^2$
Rectified linear unit (ReLU) <sup>[14]</sup>		$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$[0, \infty)$	$C^0$