BACKPROPAGATION

Sung Kim <hunkim+ml@gmail.com>
http://hunkim.github.io/ml

Neural Network(NN)

$W_1 = \begin{bmatrix} 5, -7 \\ 5, -7 \end{bmatrix}$, $B_1 = [-8, 3]$

k

$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix}$, $b_2 = 6$

X S S $\bar{Y}$

[0,0]
[0,1]
[1,0]
[1,1]
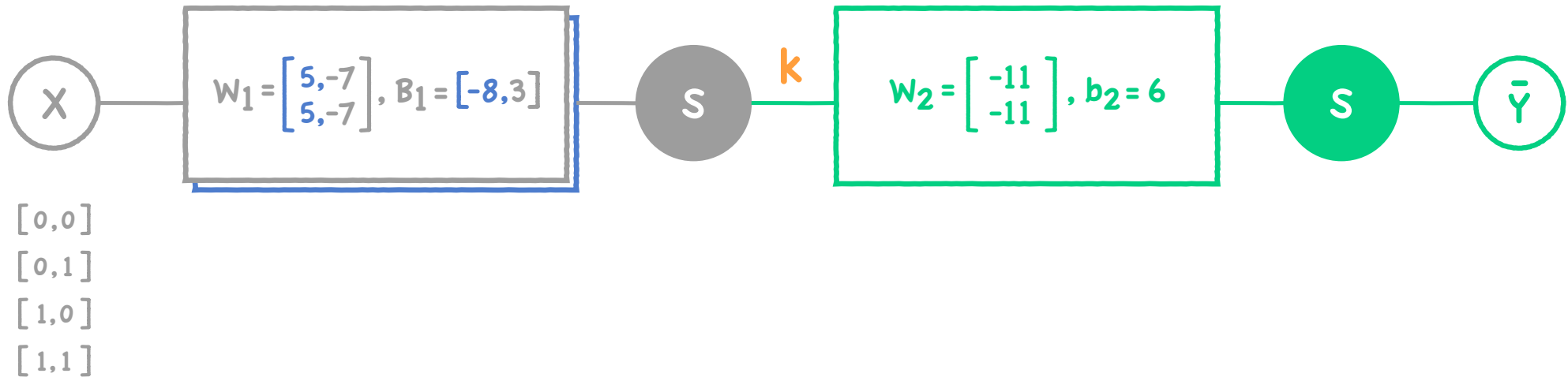
How can we learn W1, W2, B1, b2 from training data?

Neural Network(NN)

$$W_1 = \begin{bmatrix} 5,-7 \\ 5,-7 \end{bmatrix}, B_1 = [-8,3]$$

S

k

$$W_2 = \begin{bmatrix} -11 \\ -11 \end{bmatrix}, b_2 = 6$$

S

X

Ȳ

[0,0]
[0,1]
[1,0]
[1,1]

G?
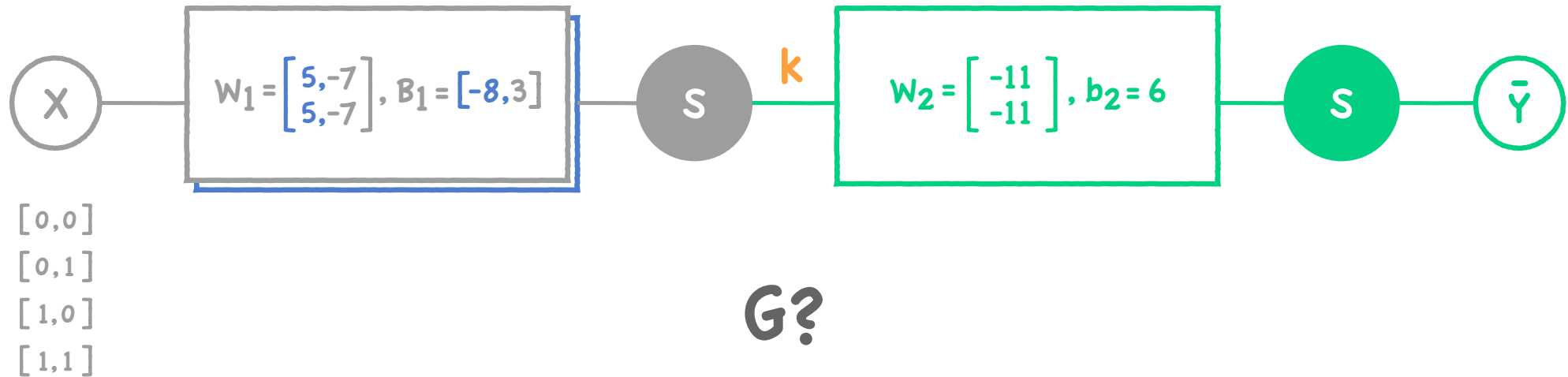
How can we learn W1, W2, B1, b2 from training data?

Derivation
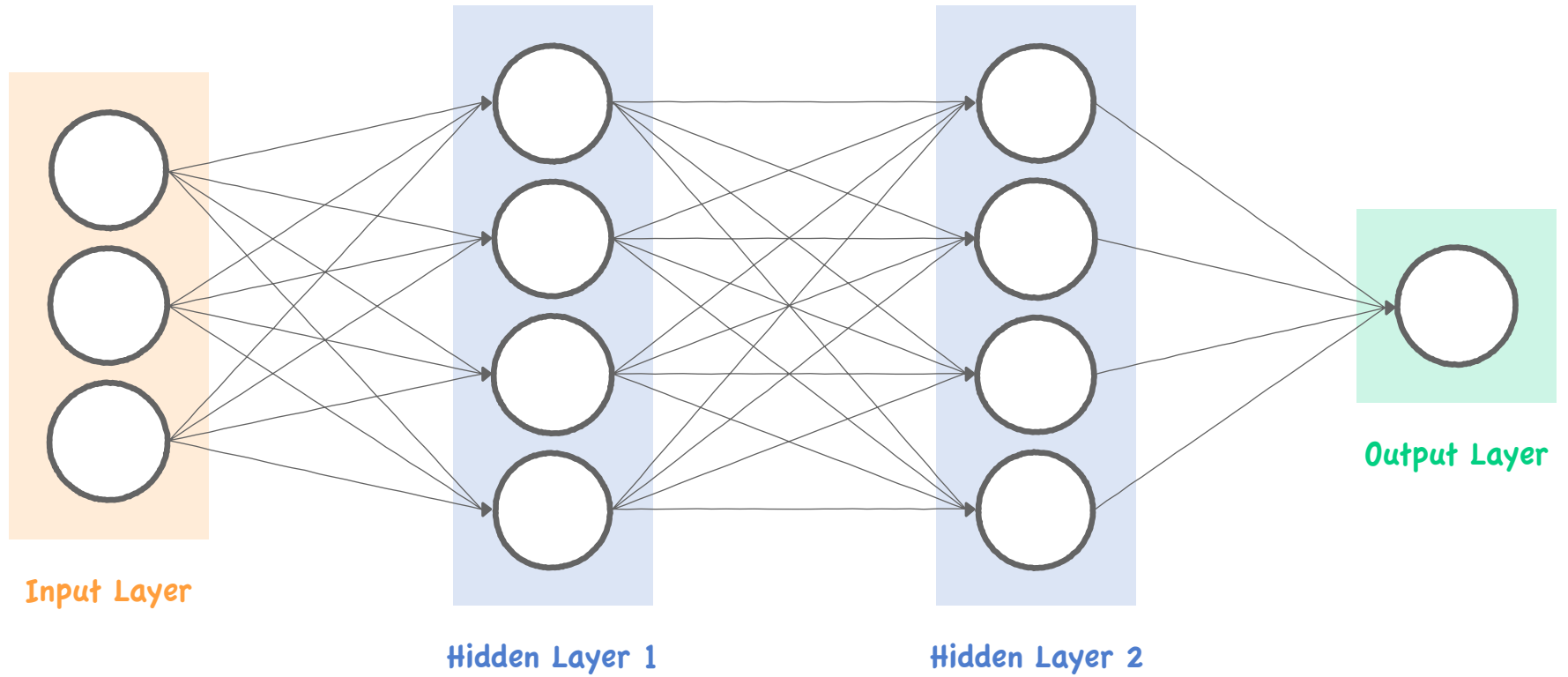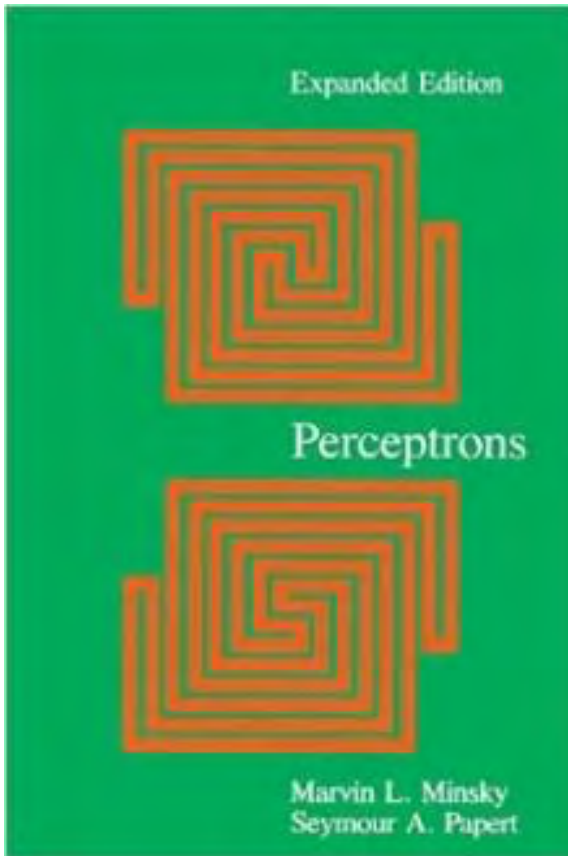
Input Layer

Hidden Layer 1

Hidden Layer 2

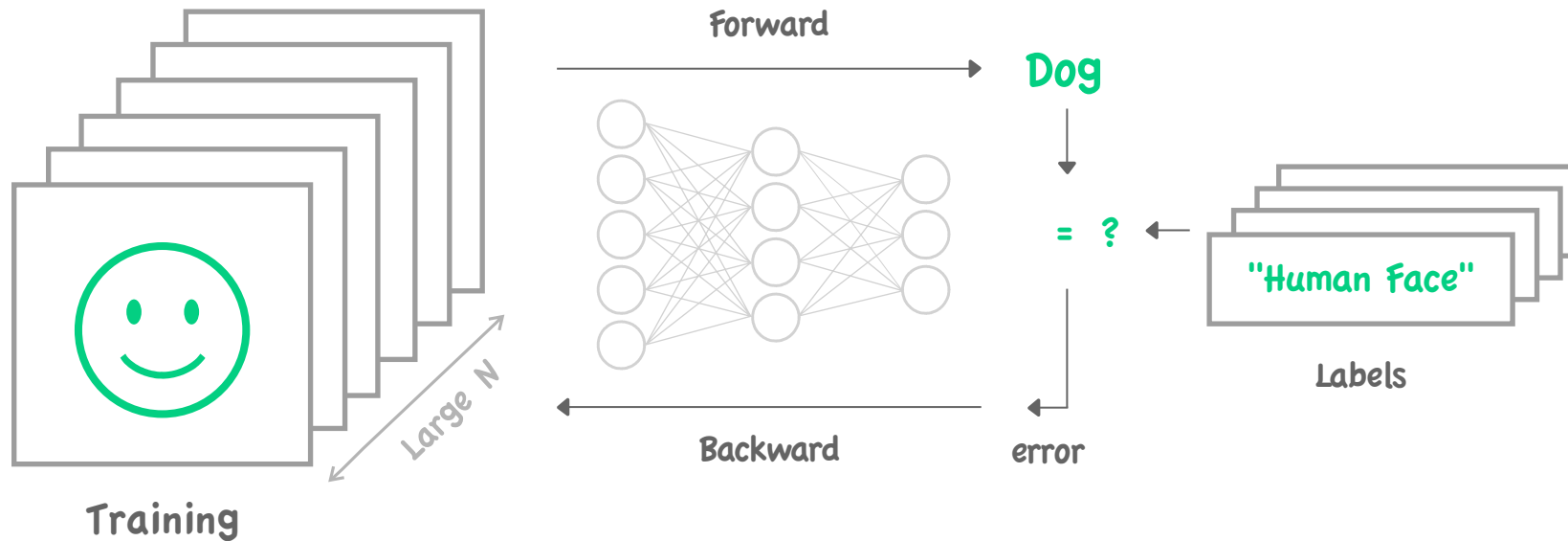Output Layer

Perceptrons (1969)

- We need to use MLP, multilayer perceptrons (multilayer neural nets)

- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

Perceptrons (1969)
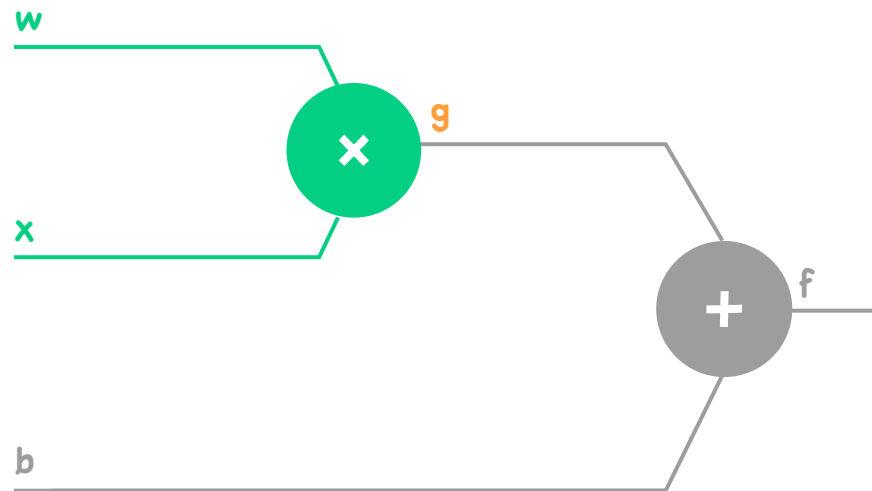by Marvin Minsky, founder of the MIT AI Lab

Backpropagation

Backpropagation (Chain Rule)

$$f = wx + b, \; g = wx, \; f = g + b$$

Backpropagation (Chain Rule)
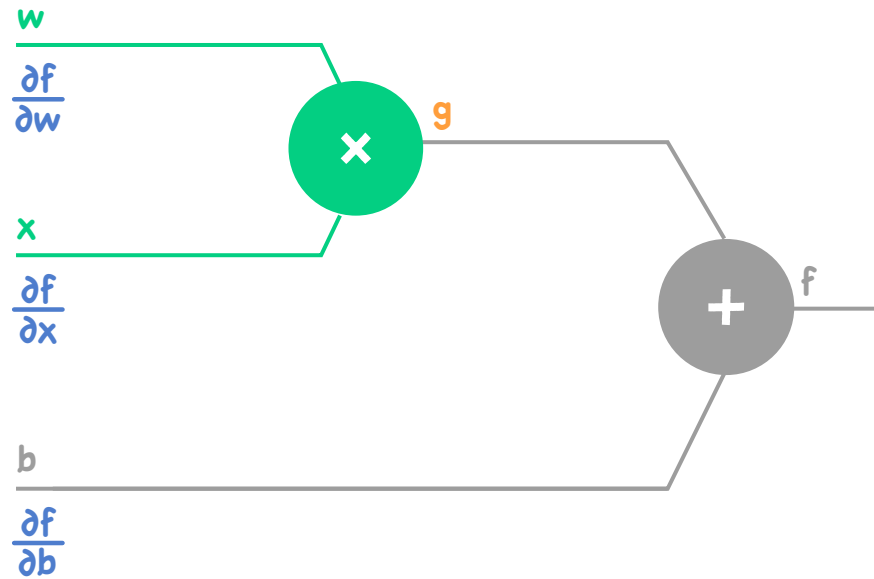
$$f = wx + b, \, g = wx, \, f = g + b$$

Backpropagation (Chain Rule)

$$f = wx + b, \quad g = wx, \quad f = g + b$$

Basic Derivative

$$\frac{d}{dx} f(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$f(x) = 3$

$f(x) = x$

$f(x) = 2x$

Partial Derivative

Consider other variables as constants

$$f(x) = 2x$$

$$f(x,y) = xy, \frac{\partial f}{\partial x}$$

$$f(x,y) = xy, \frac{\partial f}{\partial y}$$

Partial Derivative

Consider other variables as constants
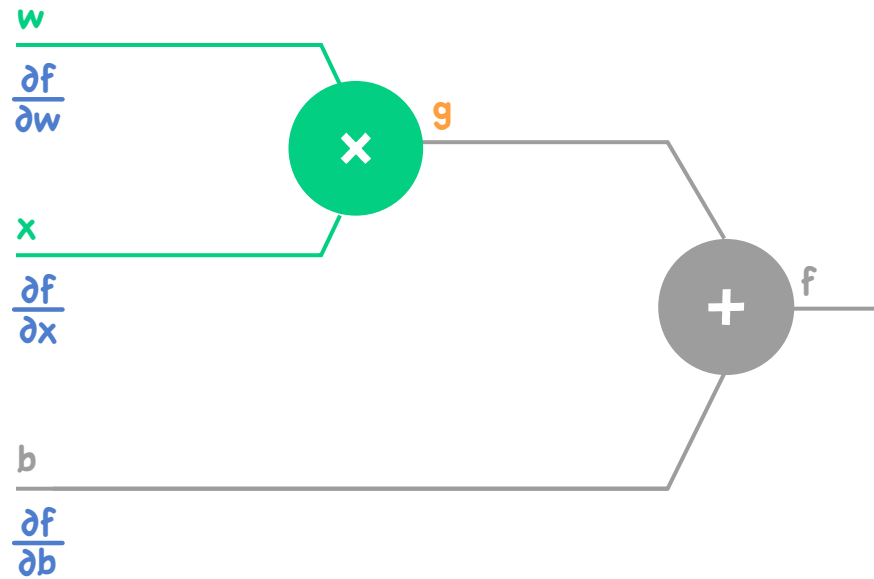
$f(x) = 3$

$f(x) = 2x$ $f(x) = x+x$

$f(x) = x+3$

$f(x,y) = x+y, \dfrac{\partial f}{\partial x}$

$f(x,y) = x+y, \dfrac{\partial f}{\partial y}$

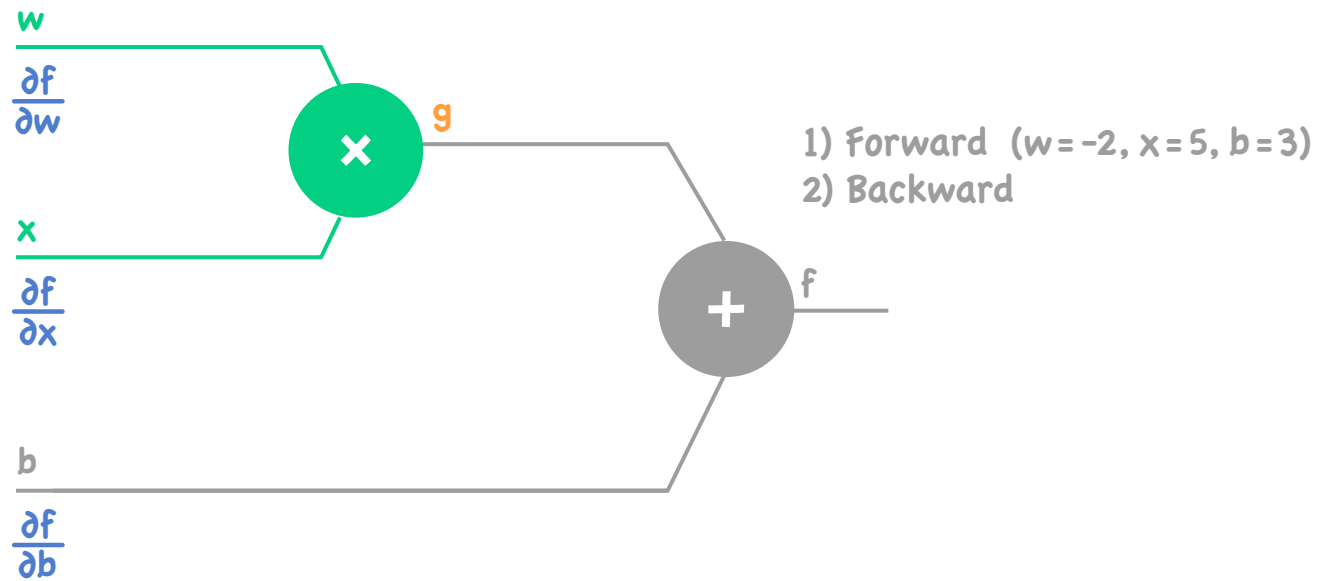# Backpropagation (Chain Rule)

$$f = wx + b, \; g = wx, \; f = g + b$$

Backpropagation (Chain Rule)

$$f = wx + b, g = wx, f = g + b$$

1) Forward (w = -2, x = 5, b = 3)
2) Backward

Backpropagation (Chain Rule)

$$f = wx + b, \; g = wx, \; f = g + b$$

w = -2

$\frac{\partial f}{\partial w}$

g = -10

x = 5

$\frac{\partial f}{\partial x}$

f = -7

b = 3

$\frac{\partial f}{\partial b}$

1) Forward (w = -2, x = 5, b = 3)
2) Backward

Backpropagation (Chain Rule)

$$f = wx + b, \; g = wx, \; f = g + b$$

$$\frac{\partial g}{\partial w} = x, \; \frac{\partial g}{\partial x} = w$$

$$\frac{\partial f}{\partial g} = 1, \; \frac{\partial f}{\partial b} = 1$$

w = -2

$\dfrac{\partial f}{\partial w}$

g = -10

x = 5

$\dfrac{\partial f}{\partial x}$

f = -7

1) Forward (w = -2, x = 5, b = 3)
2) Backward

b = 3

$\dfrac{\partial f}{\partial b}$

Backpropagation (Chain Rule)

$$f = wx + b, \; g = wx, \; f = g + b$$

$$\frac{\partial g}{\partial w} = x, \; \frac{\partial g}{\partial x} = w$$

$$\frac{\partial f}{\partial g} = 1, \; \frac{\partial f}{\partial b} = 1$$

w = -2

$\frac{\partial f}{\partial w}$

g = -10

$\frac{\partial f}{\partial g} = 1$

x = 5

$\frac{\partial f}{\partial x}$

f = -7

b = 3

$\frac{\partial f}{\partial b}$

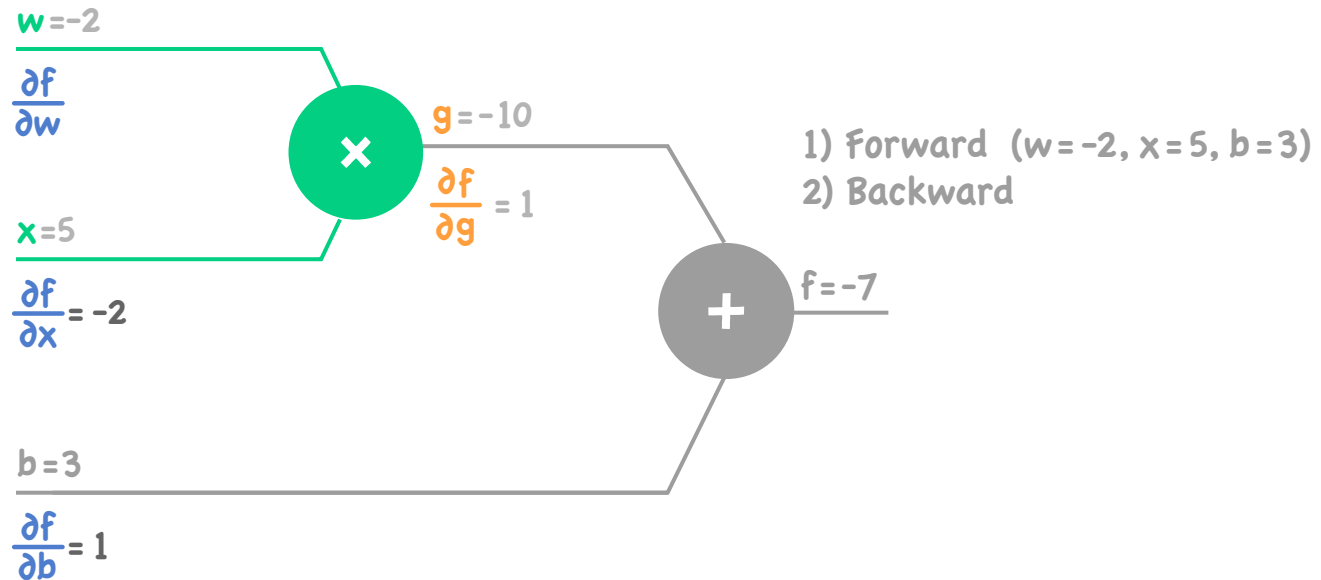1) Forward (w = -2, x = 5, b = 3)
2) Backward

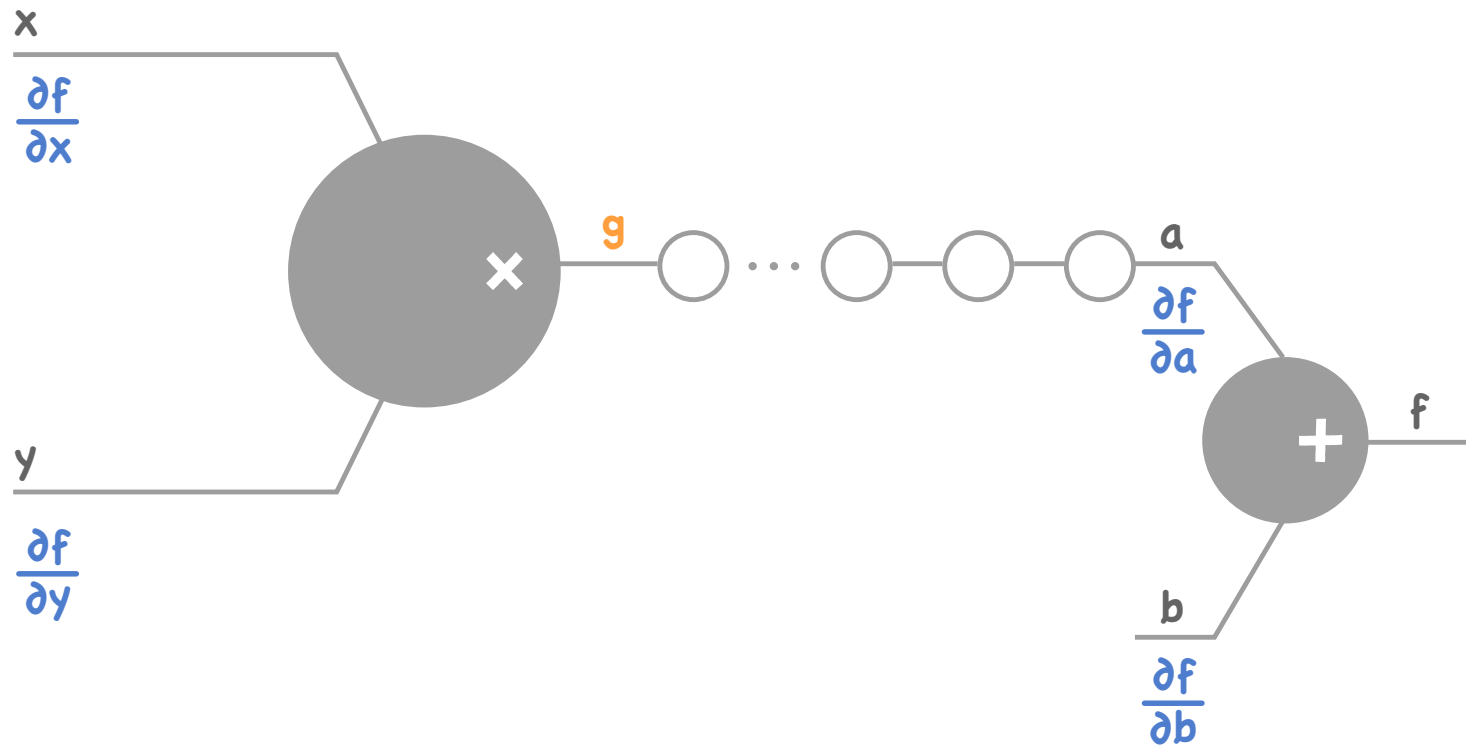# Backpropagation (Chain Rule)

$$f = wx + b, \ g = wx, \ f = g + b$$

$$\frac{\partial g}{\partial w} = x, \ \frac{\partial g}{\partial x} = w$$

$$\frac{\partial f}{\partial g} = 1, \ \frac{\partial f}{\partial b} = 1$$

1) Forward (w = -2, x = 5, b = 3)
2) Backward
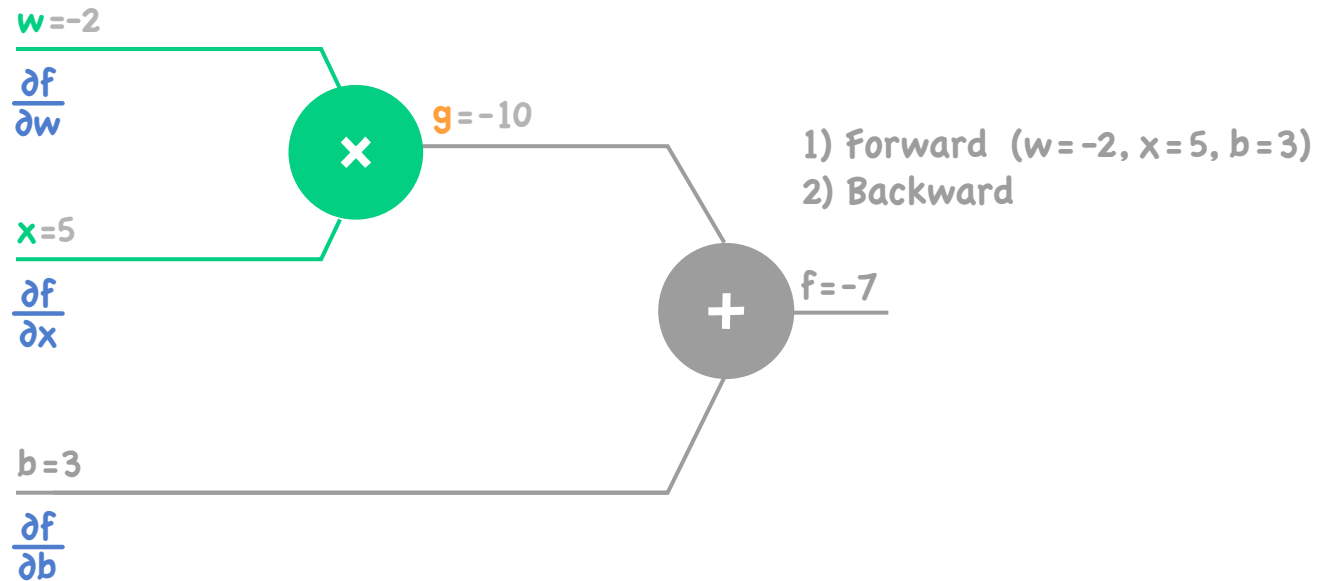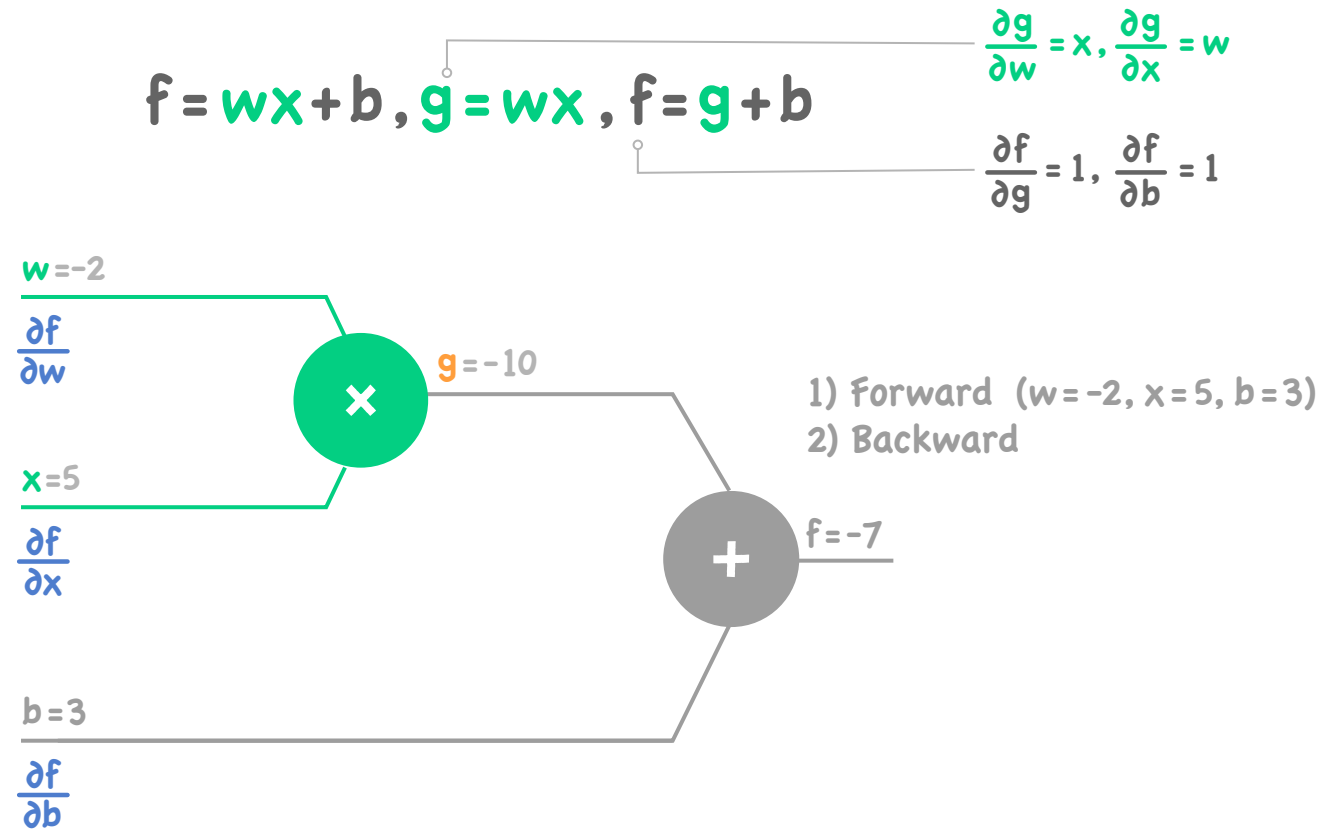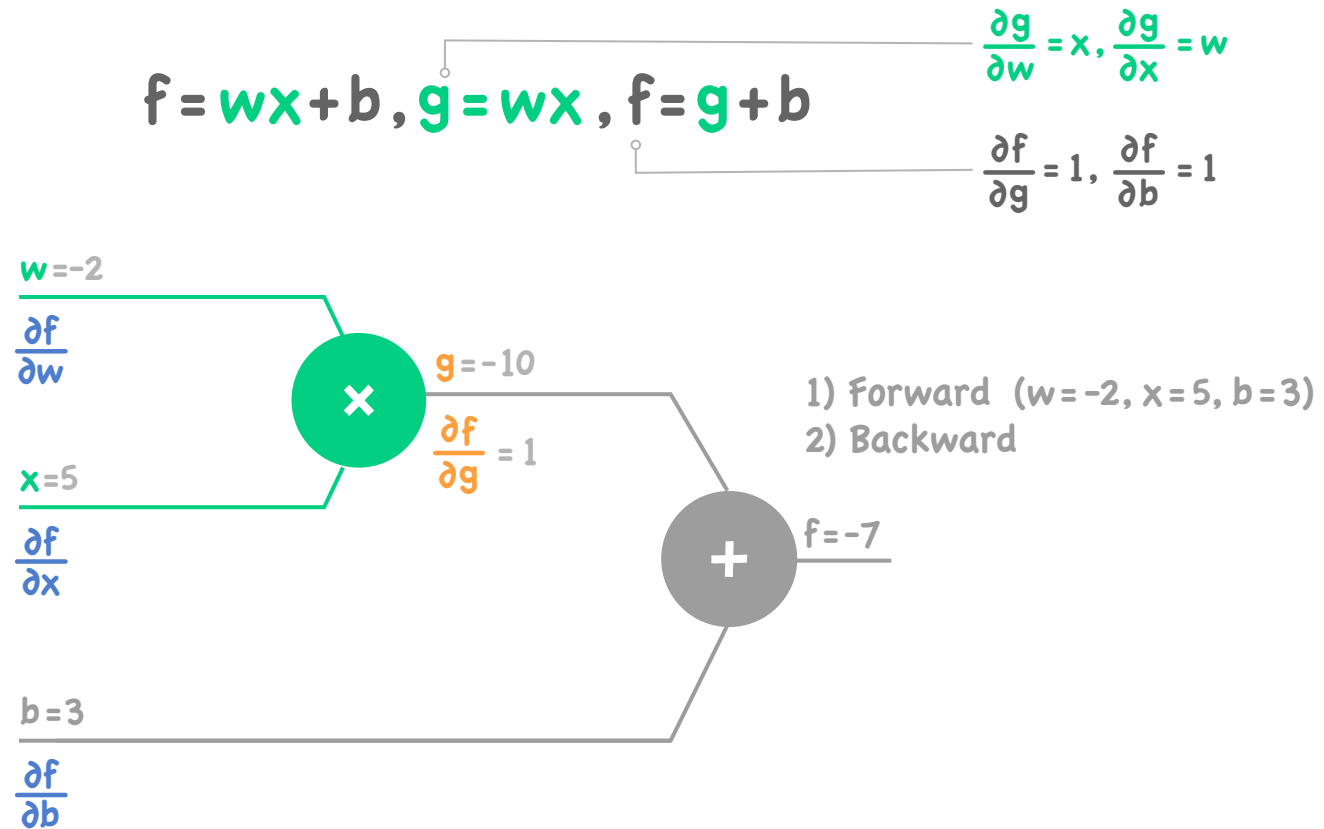
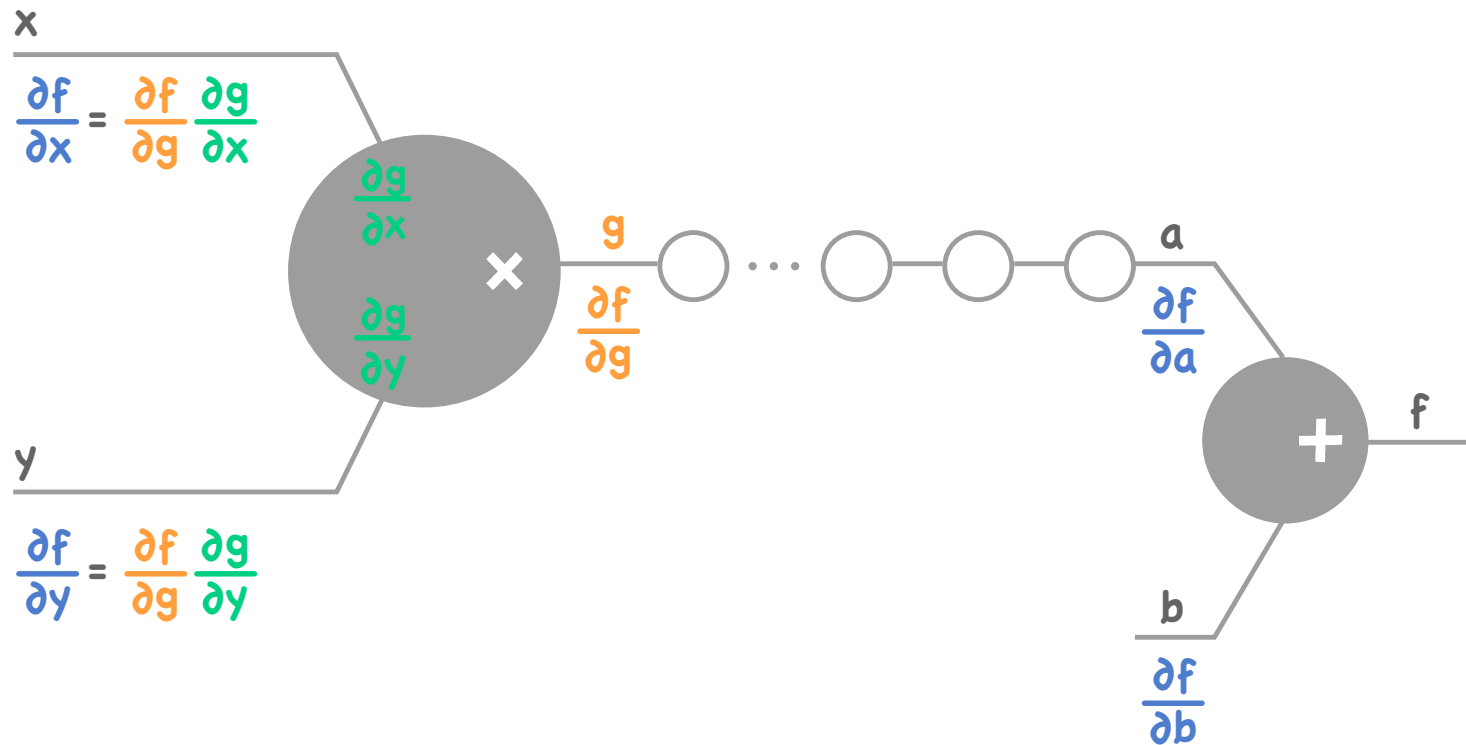$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 1 \times w = -2$$

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 1 \times x = 5$$
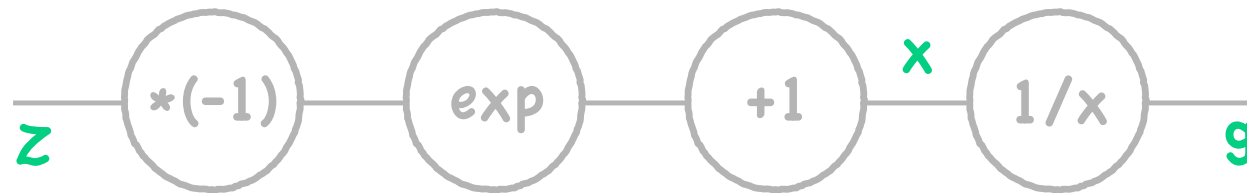
Backpropagation (Chain Rule)

http://cs231n.stanford.edu/

Backpropagation (Chain Rule)

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial y}$$

x

$\frac{\partial g}{\partial x}$

$\frac{\partial g}{\partial y}$

×

y

g

$\frac{\partial f}{\partial g}$

a

$\frac{\partial f}{\partial a}$

+

f

b

$\frac{\partial f}{\partial b}$
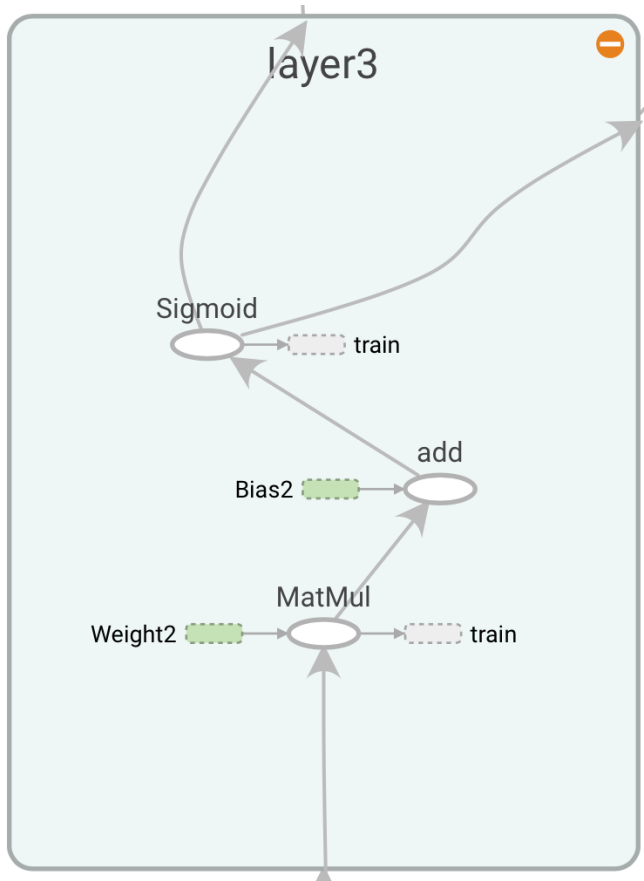
http://cs231n.stanford.edu/

Sigmoid

$$g(z) = \frac{1}{1+e^{-2}}$$

Sigmoid

$$g(z) = \frac{1}{1+e^{-2}}$$

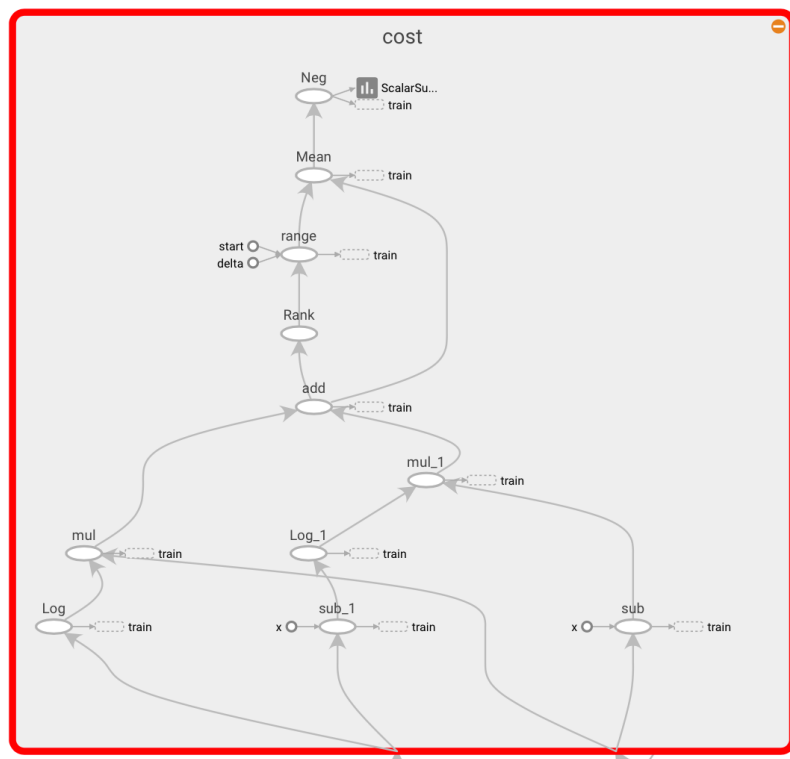# Back Propagation in TensorFlow

[TensorBoard]

```
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

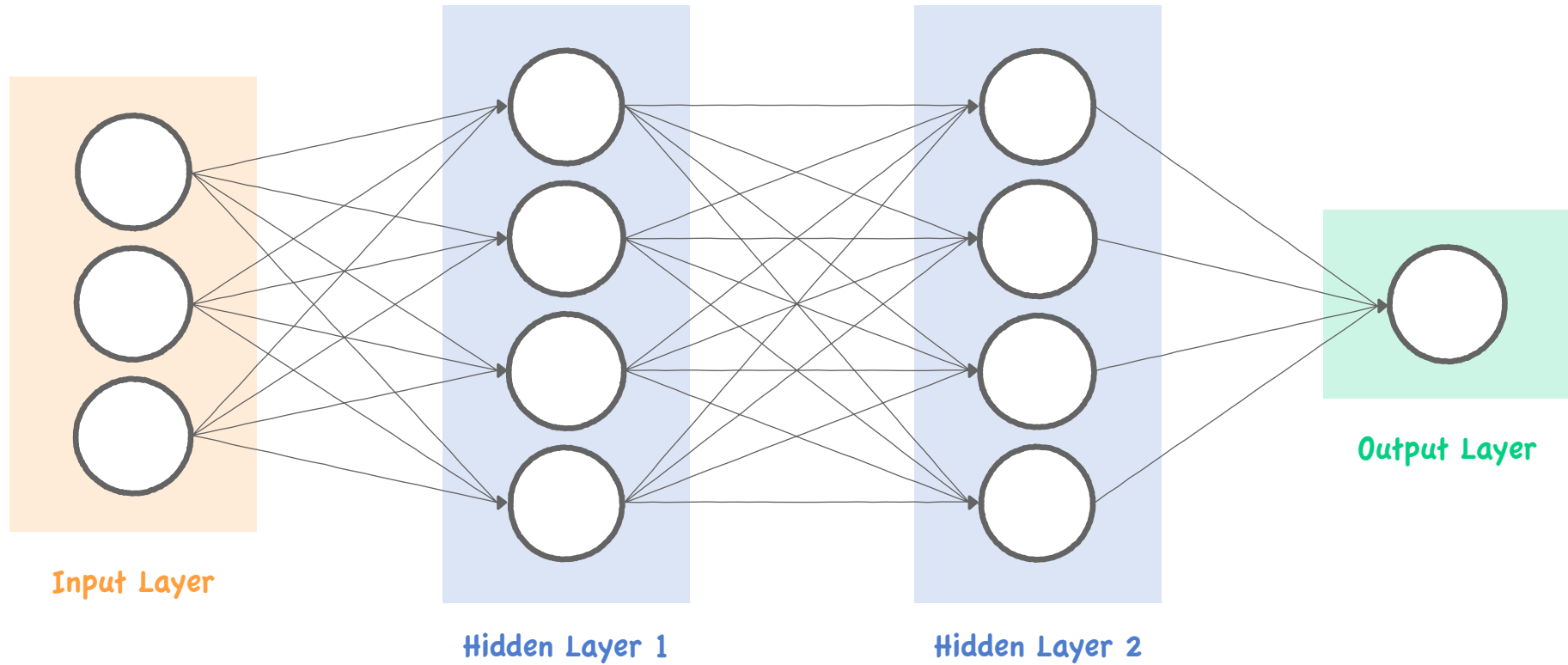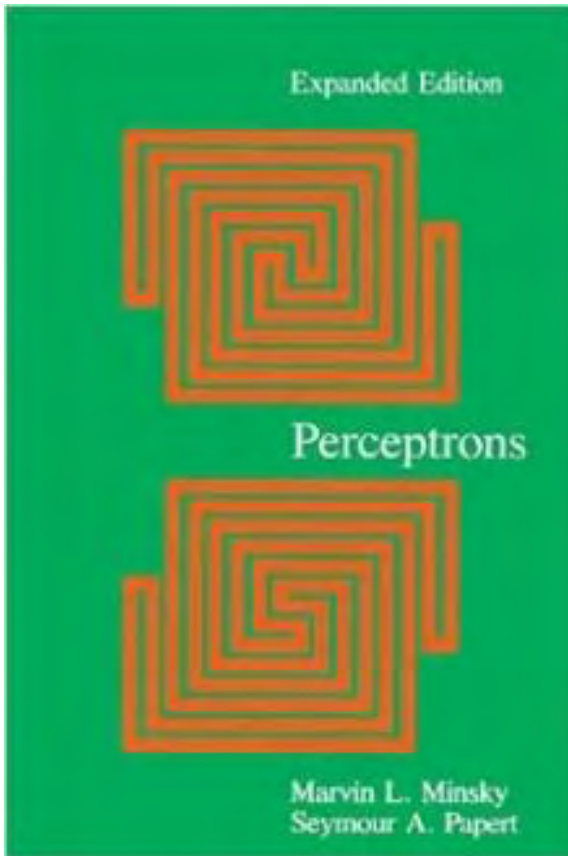# Back Propagation in TensorFlow

[TensorBoard]

```
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
```

Backpropagation

Input Layer

Hidden Layer 1

Hidden Layer 2

Output Layer

Perceptrons (1969)

- We need to use MLP, multilayer perceptrons (multilayer neural nets)

- No one on earth had found a viable way to train MLPs good enough to learn such simple functions.

Perceptrons (1969)
by Marvin Minsky, founder of the MIT AI Lab

ReLU