NEURAL NETWORKS

The Neuron

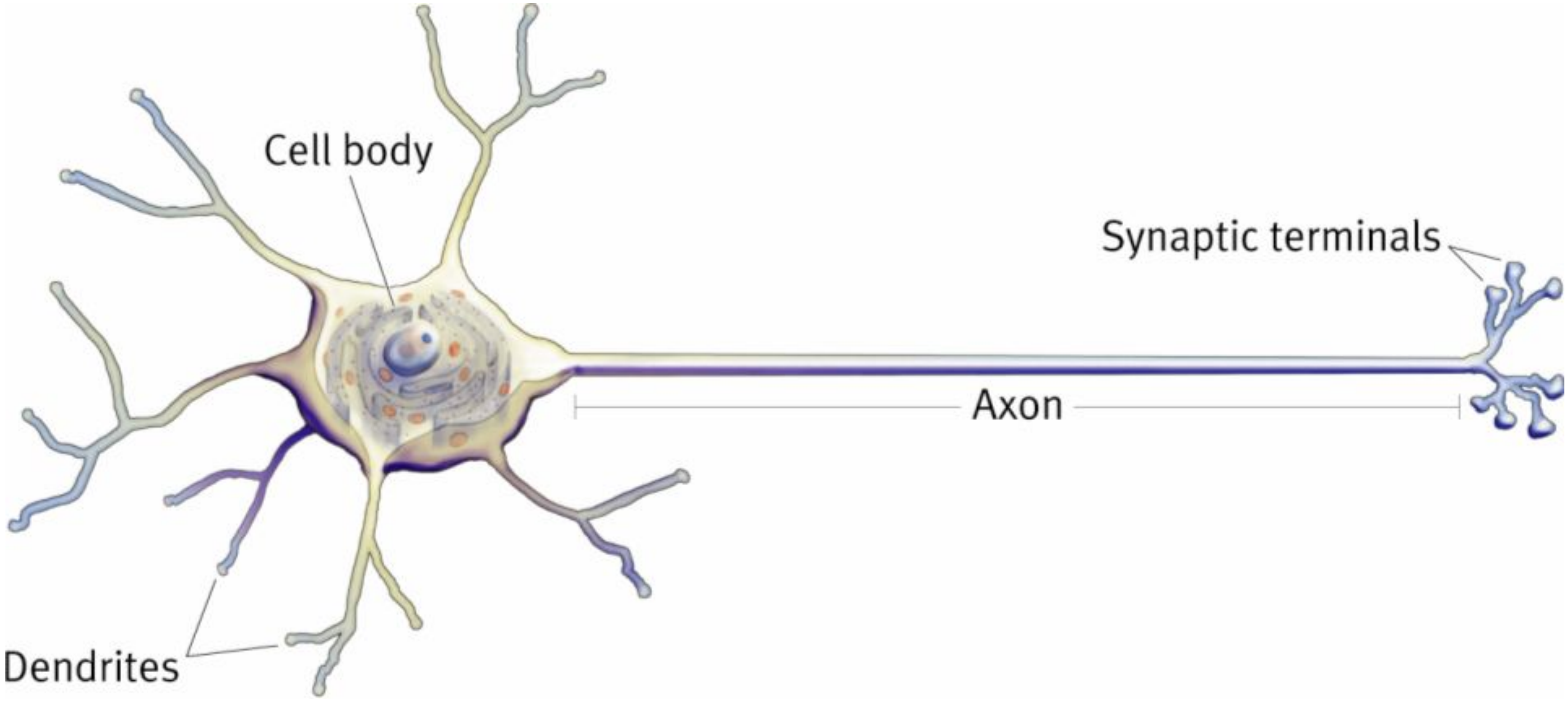


Image courtesy: `http://bio3520.nicerweb.com/Locked/chap/ch03/neuron.html`

The Neuron

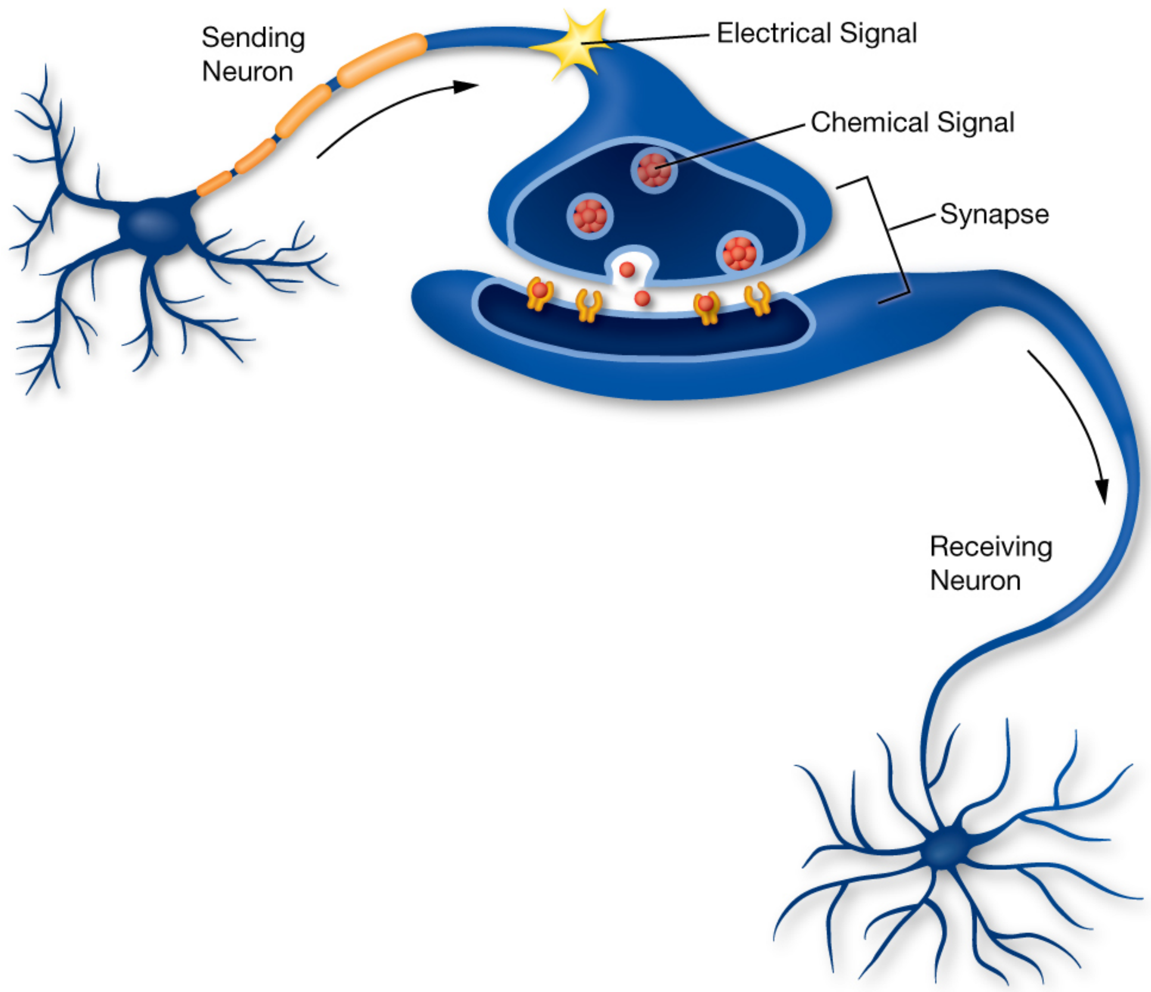


Image courtesy:
`http://learn.genetics.utah.edu/content/addiction/neurons/`

“Real” Neuron

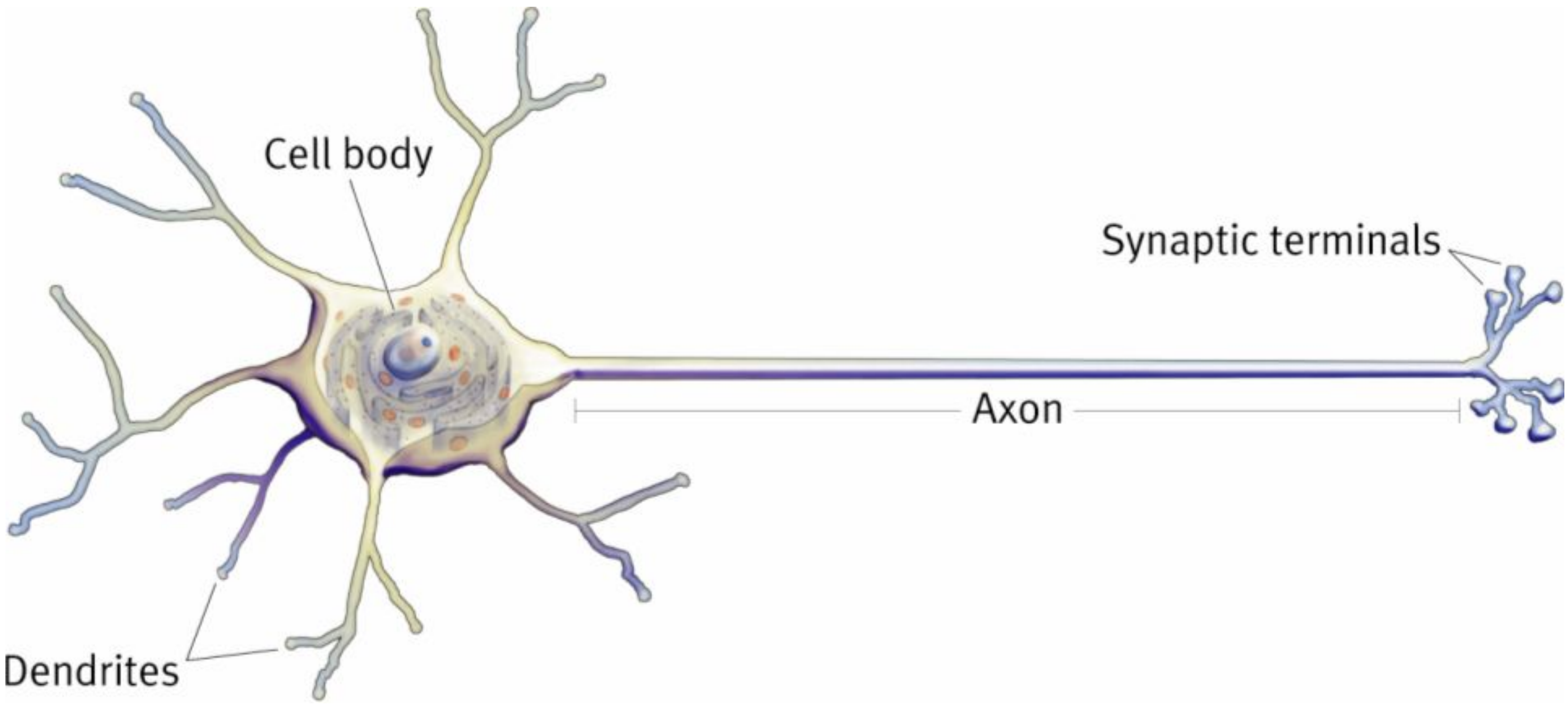
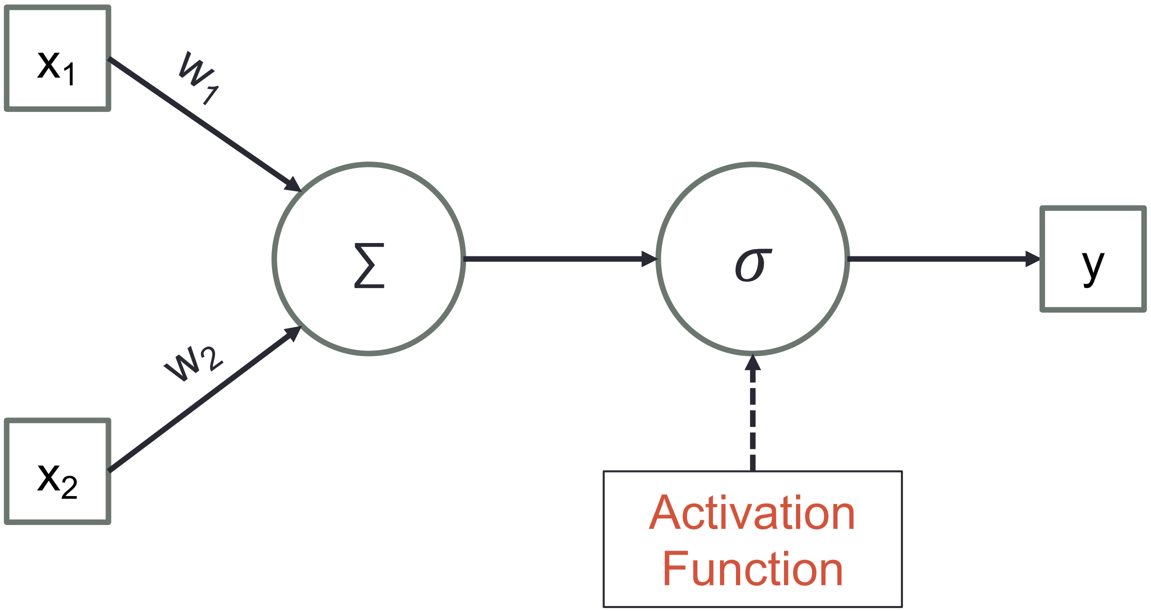
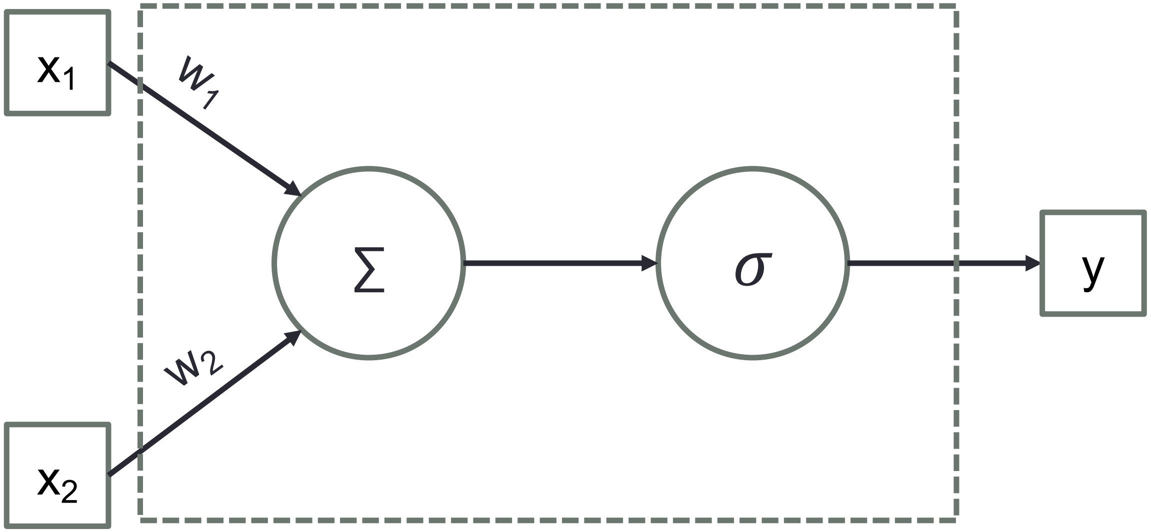


Image courtesy: `http://bio3520.nicerweb.com/Locked/chap/ch03/neuron.html`

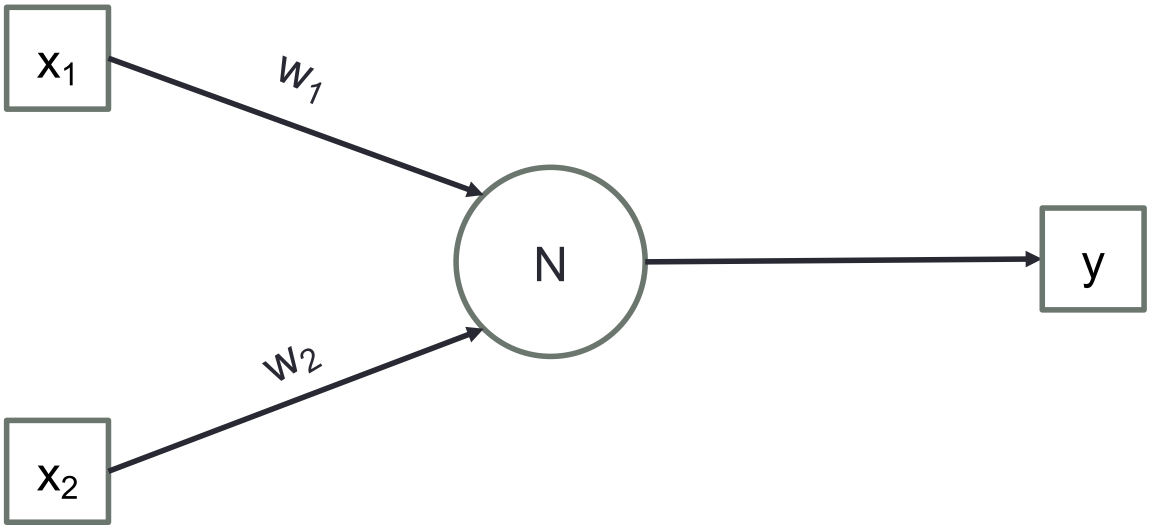
Artificial Neurons



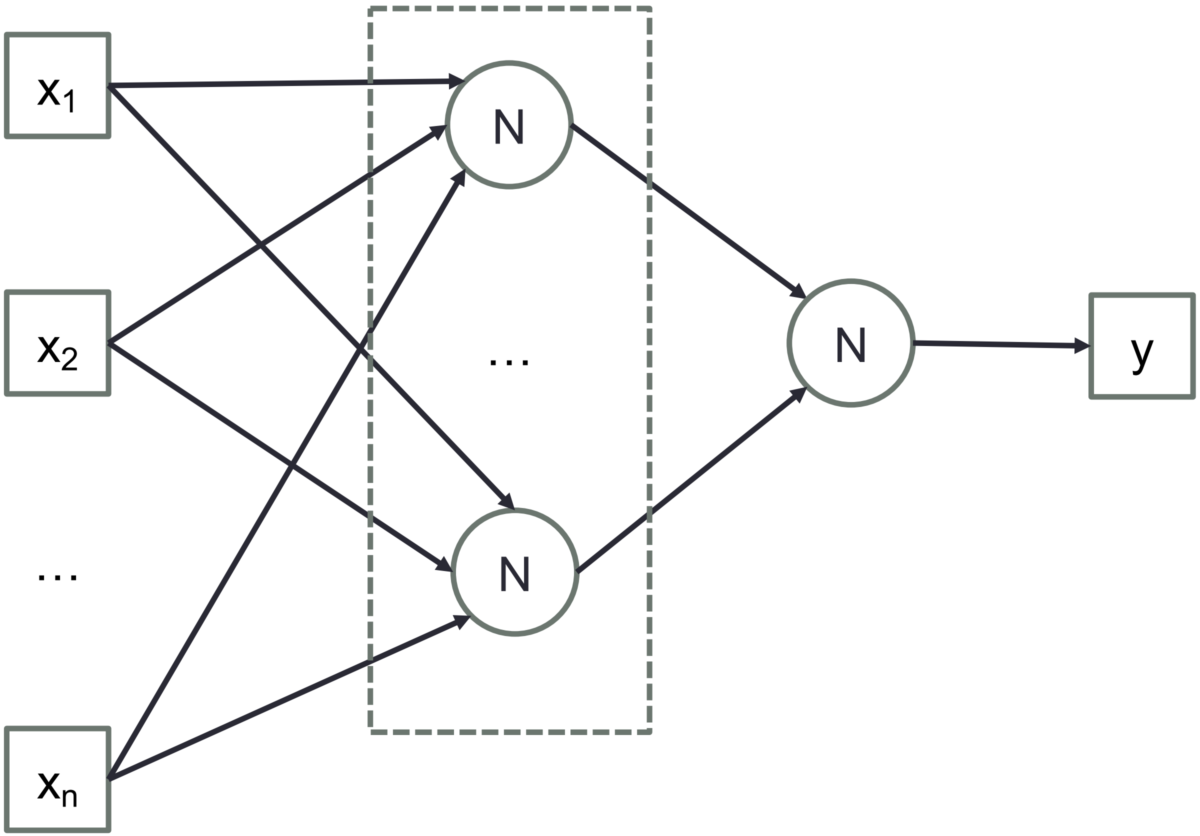
Artificial Neurons



Artificial Neurons



Fully-Connected Neural Network



Input Layer

Hidden Layer

Output Layer

Backpropagation:

Initialize all weights in the network to small, random numbers.

loop

  for each training example $(\mathbf{x}, y)$ **do**

    ForwardProp:

      For each hidden unit h, $a_h = \sigma(net_h) = \sigma(\sum_i w_{ih} x_i)$

      $\hat{y} = a_k = \sigma(net_k) = \sigma(\sum_h w_h a_h)$

    BackProp:

      $\delta_k = \frac{\partial J}{\partial net_k} = (y - \hat{y})\hat{y}(1 - \hat{y})$

      For each weight w_h, $w_h \leftarrow w_h - \eta \delta_k a_h$

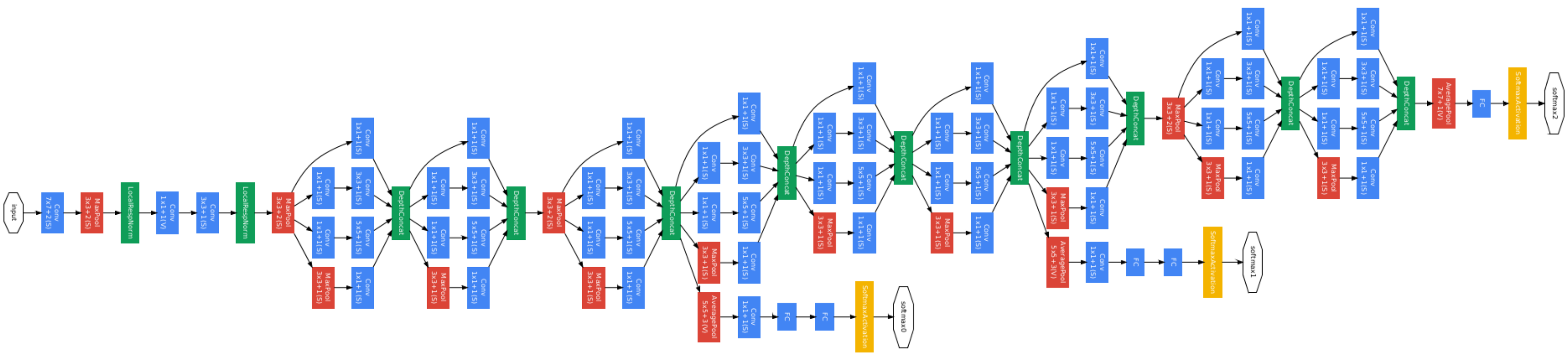
      For each hidden unit h, $\delta_h = \delta_k w_h a_h (1 - a_h)$

      For each weight w_{ih}, $w_{ih} \leftarrow w_{ih} - \eta \delta_h x_i$

  end for

end loop

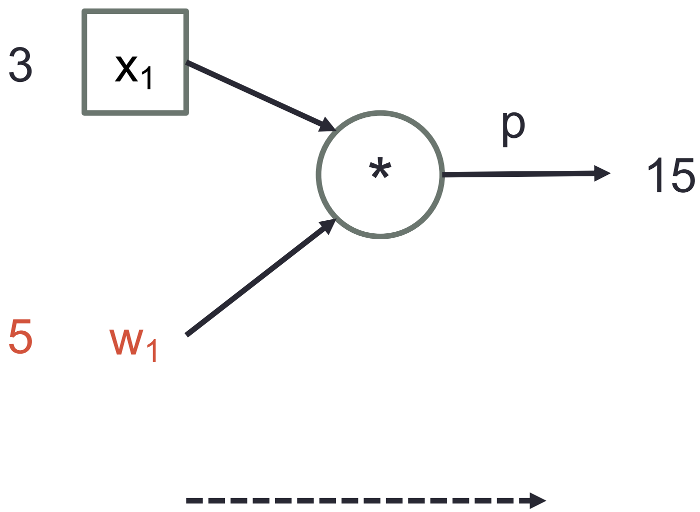
Modern Neural Networks



Automatic Differentiation

- Use the abstraction of a computational graph
- Define your computation and let engine worry about optimization

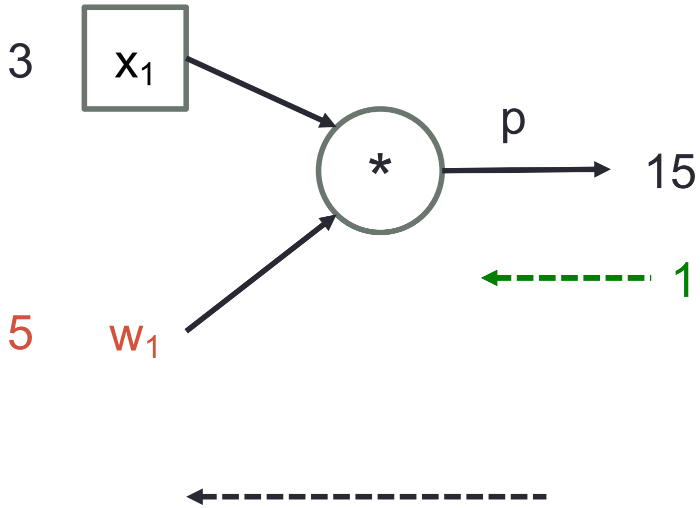
Computational Graph



Forward Pass

- Apply the operator

Computational Graph

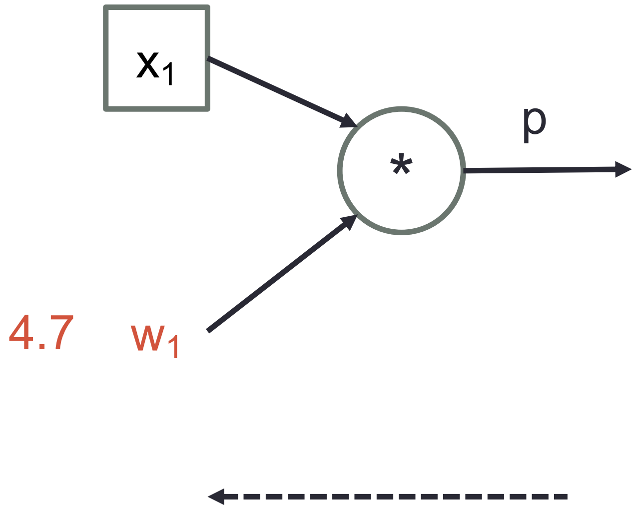


$$\frac{\partial p}{\partial w_1} = x_1$$

Backward Pass

- Adjust parameter using local gradient 3 (scaled by a learning rate)

Computational Graph

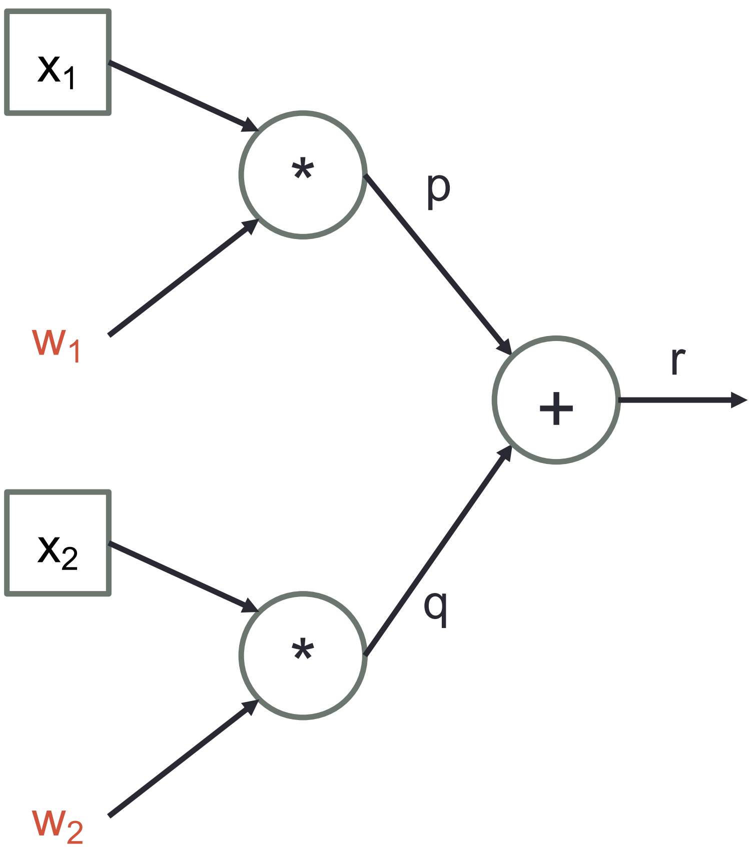


$$\frac{\partial p}{\partial w_1} = x_1$$

Backward Pass

- Adjust parameter using local gradient 3 (scaled by a learning rate)

Computational Graph

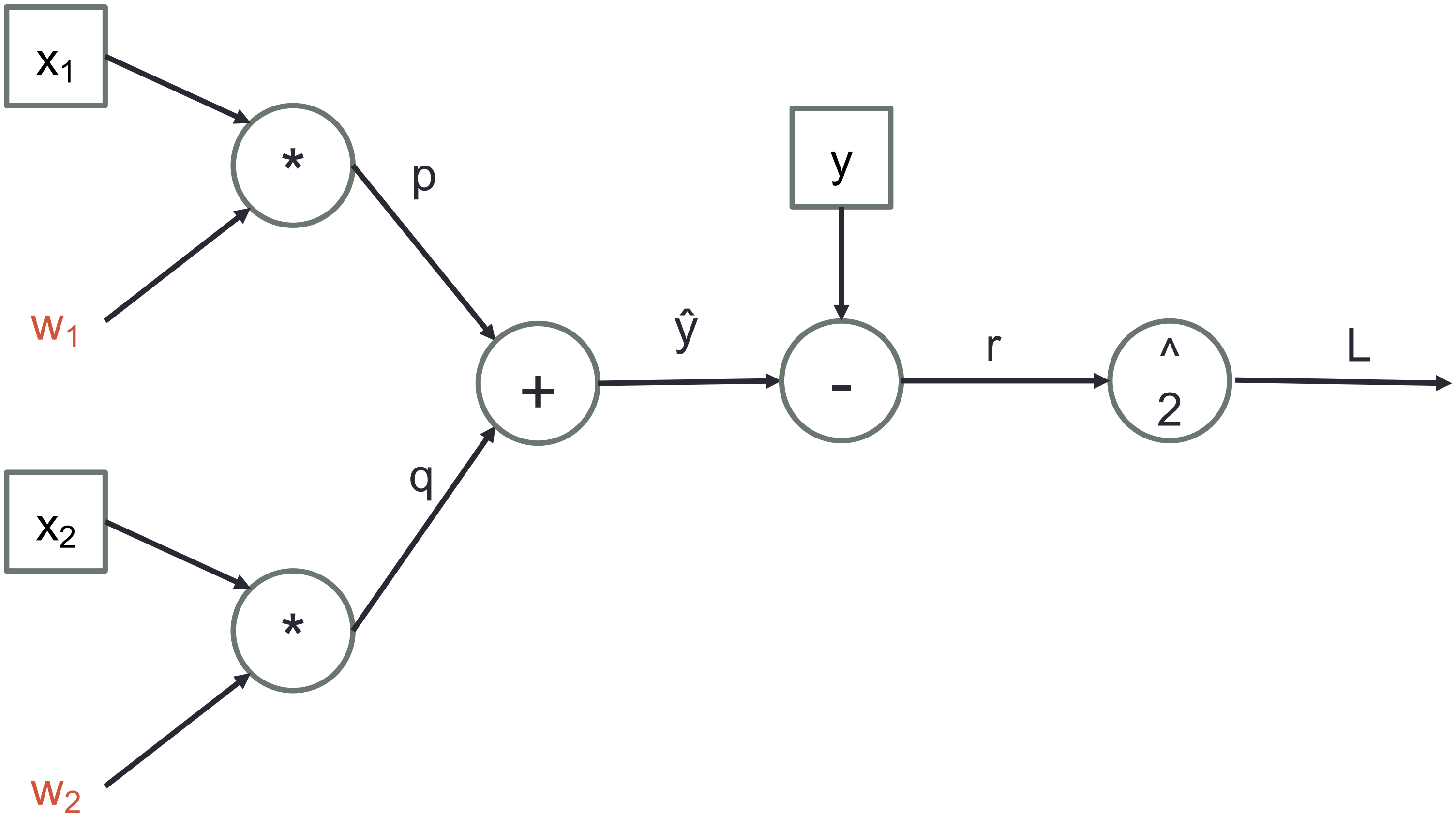


$$\frac{\partial r}{\partial p} = 1$$

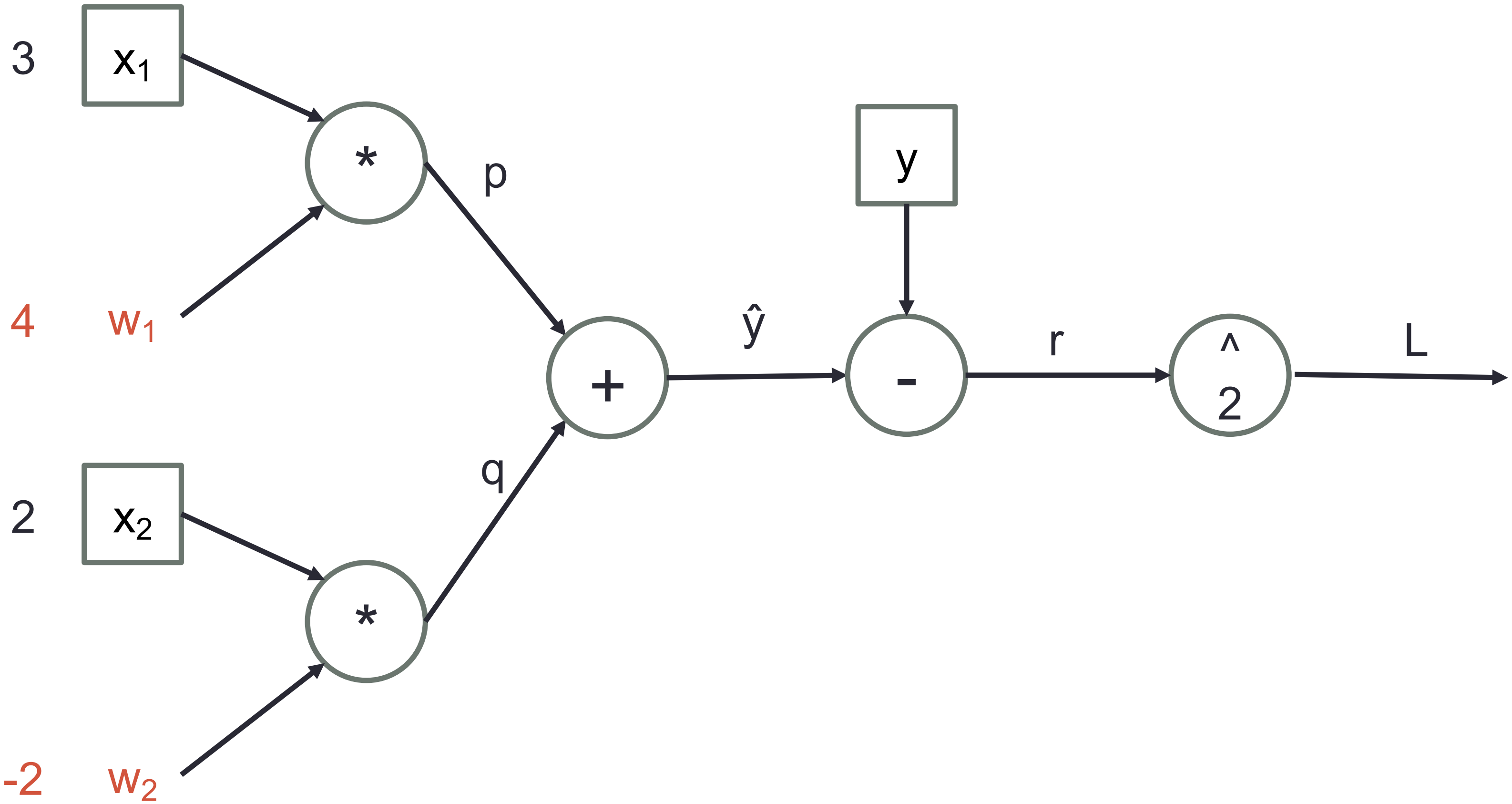
$$\frac{\partial p}{\partial w_1} = x_1$$

$$\frac{\partial r}{\partial w_1} = \frac{\partial r}{\partial p}\frac{\partial p}{\partial w_1}$$

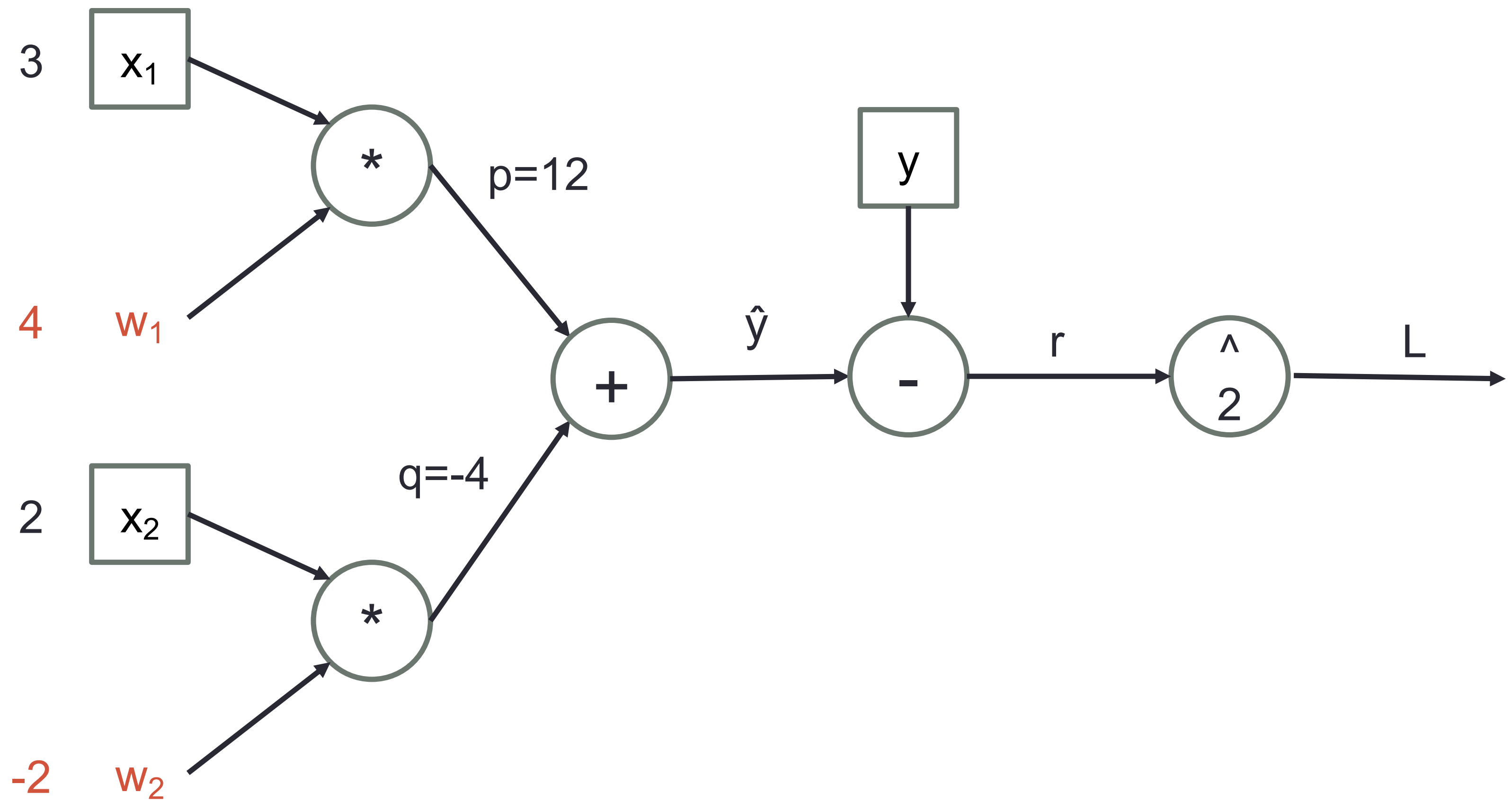
Computational Graph



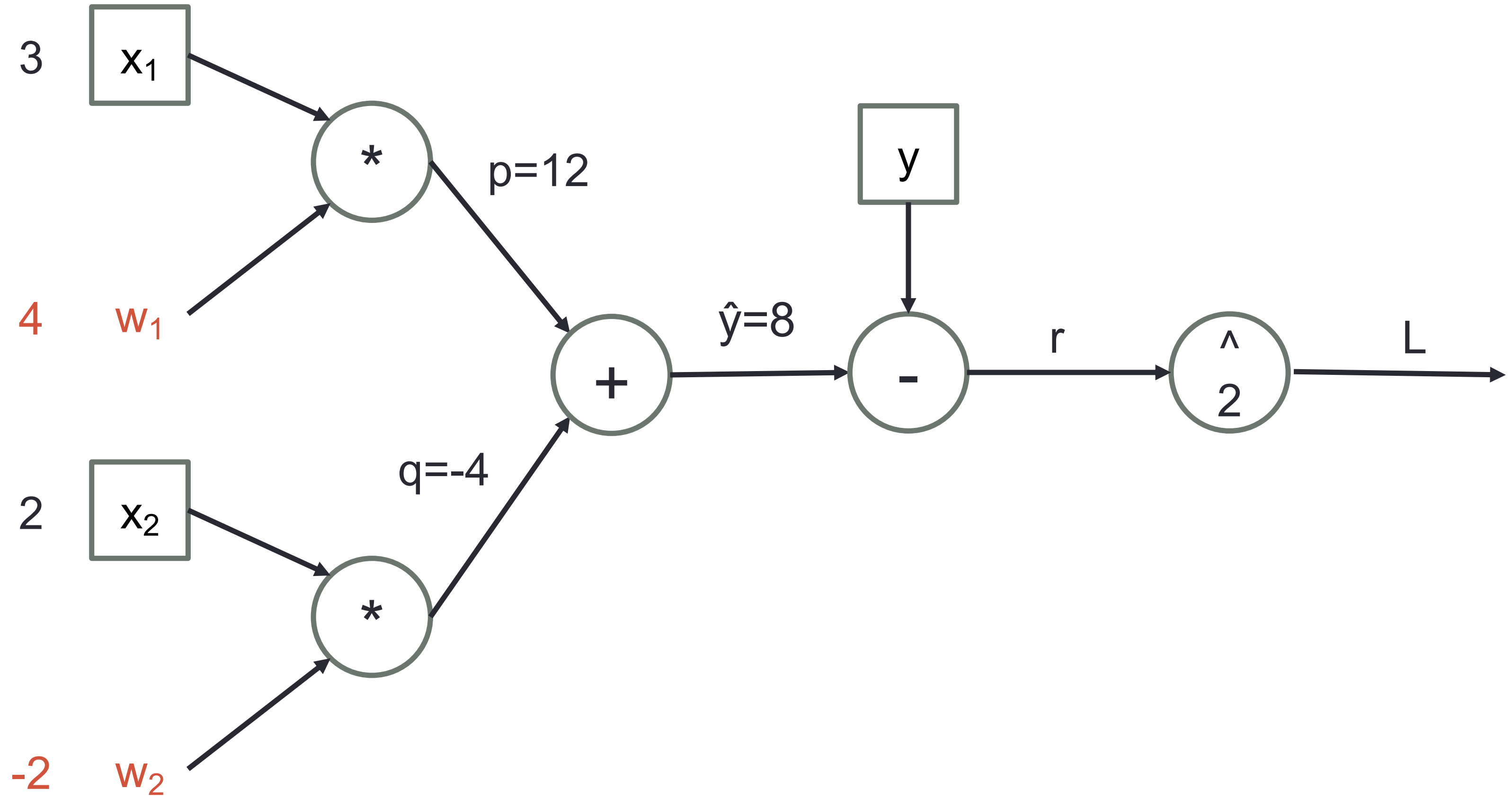
Computational Graph



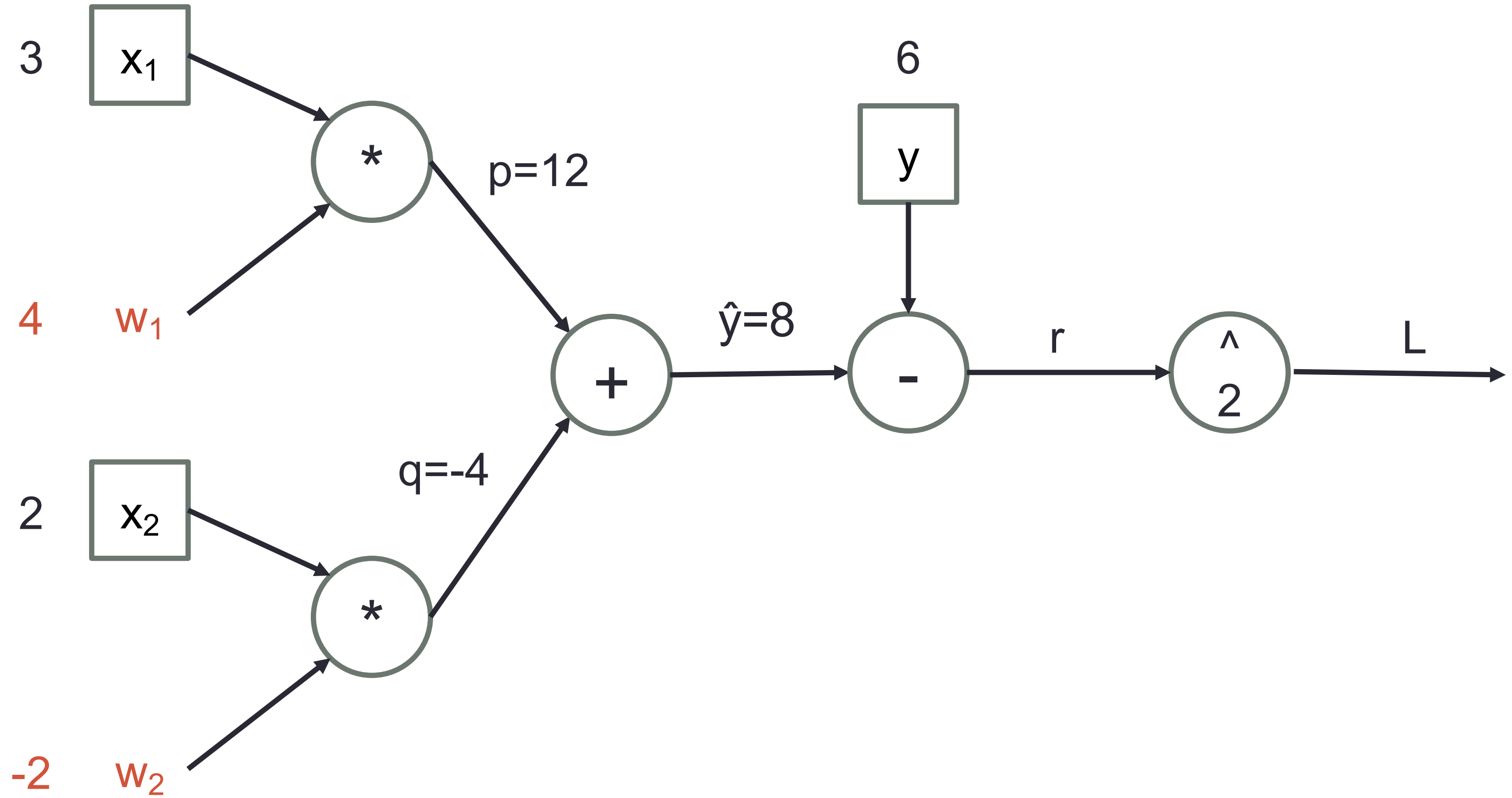
Computational Graph



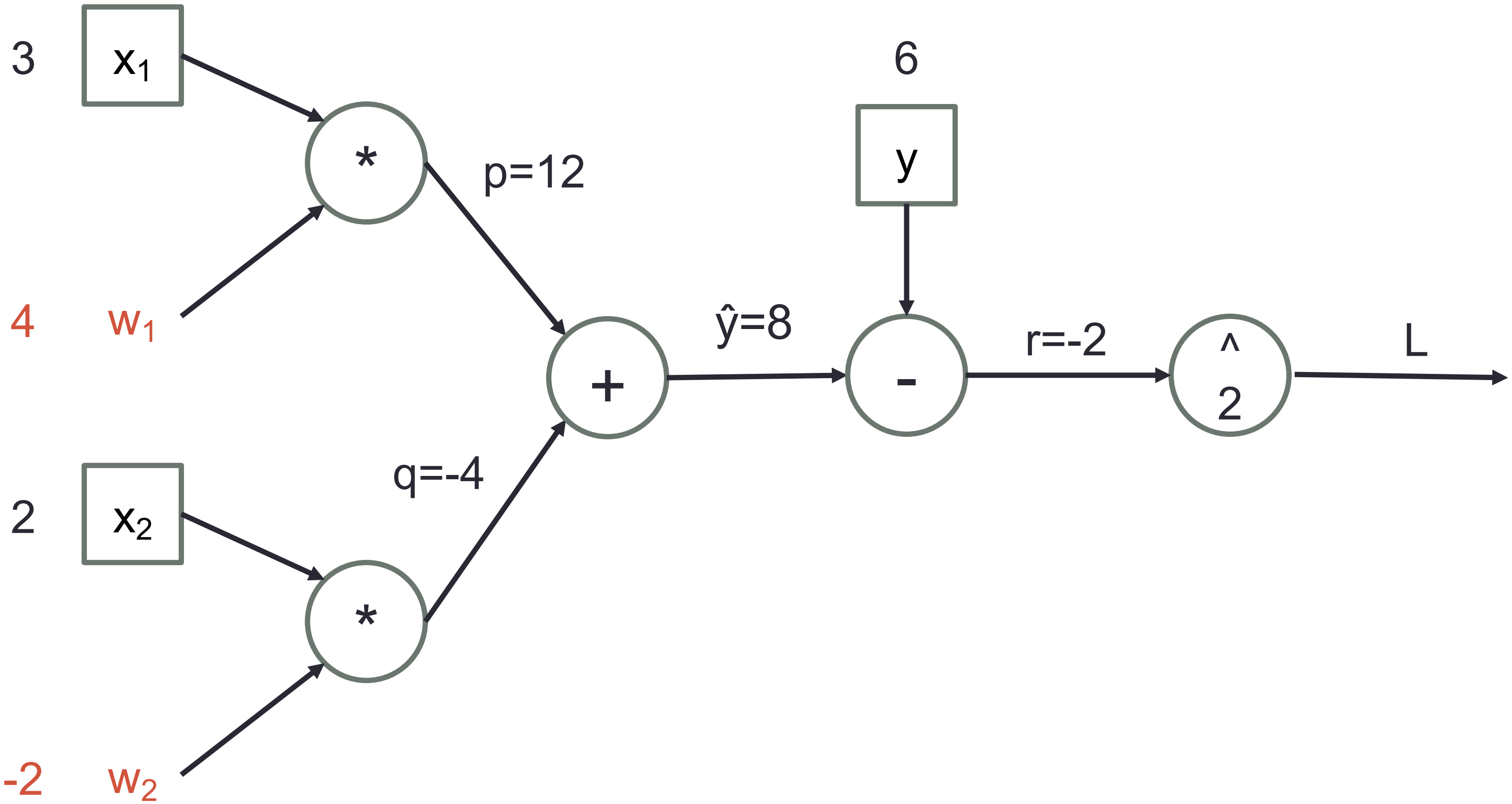
Computational Graph



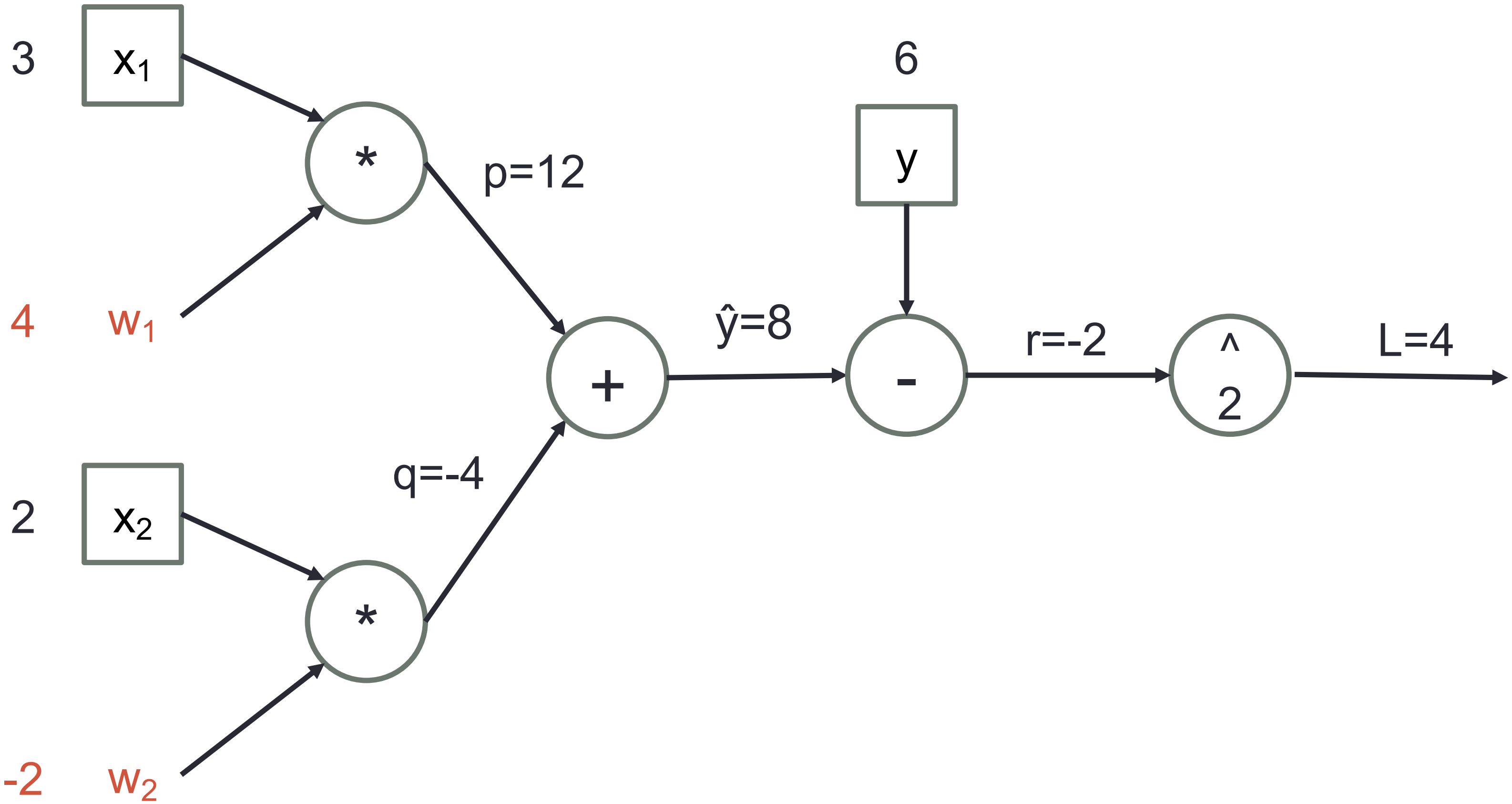
Computational Graph



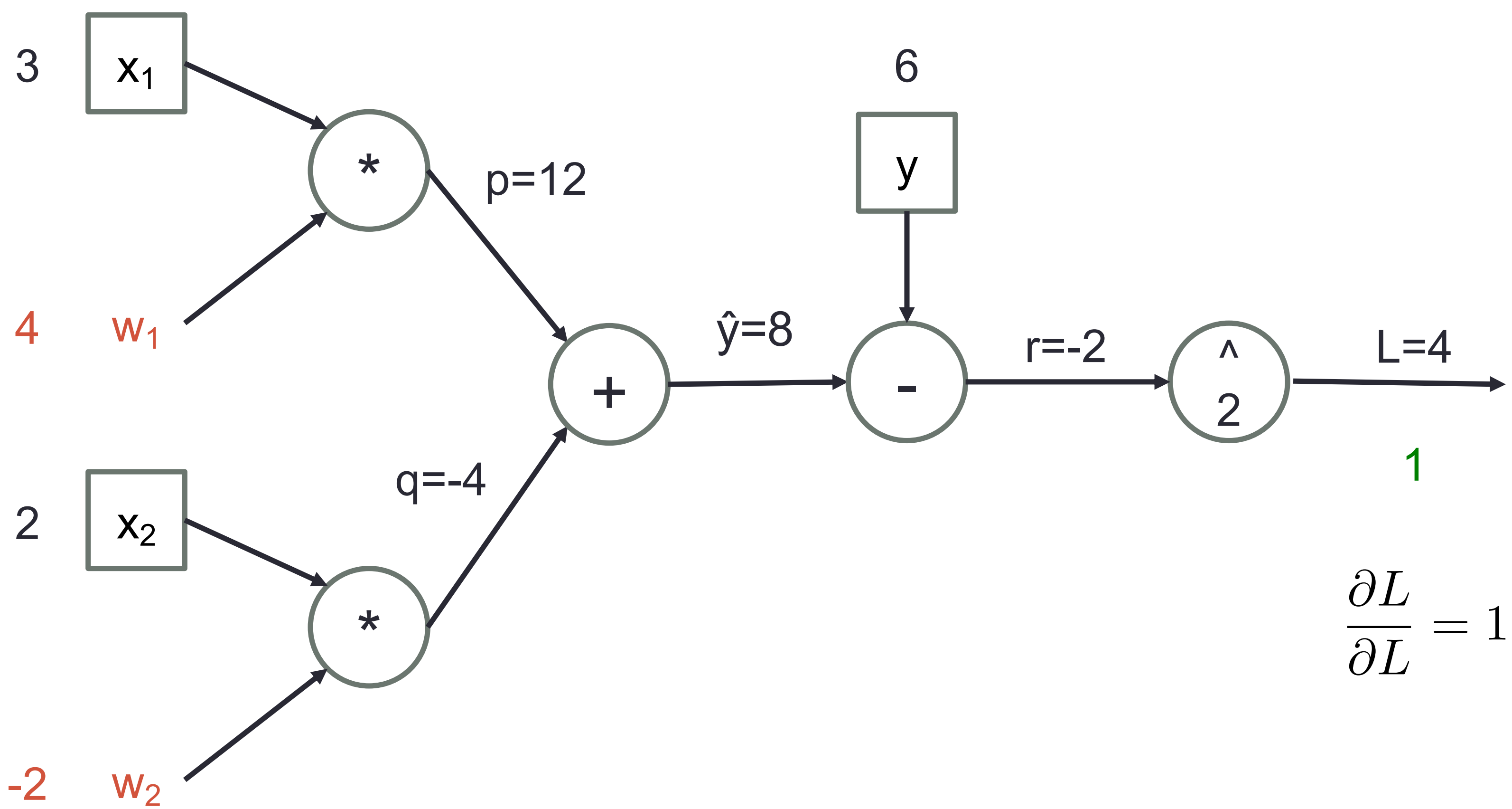
Computational Graph



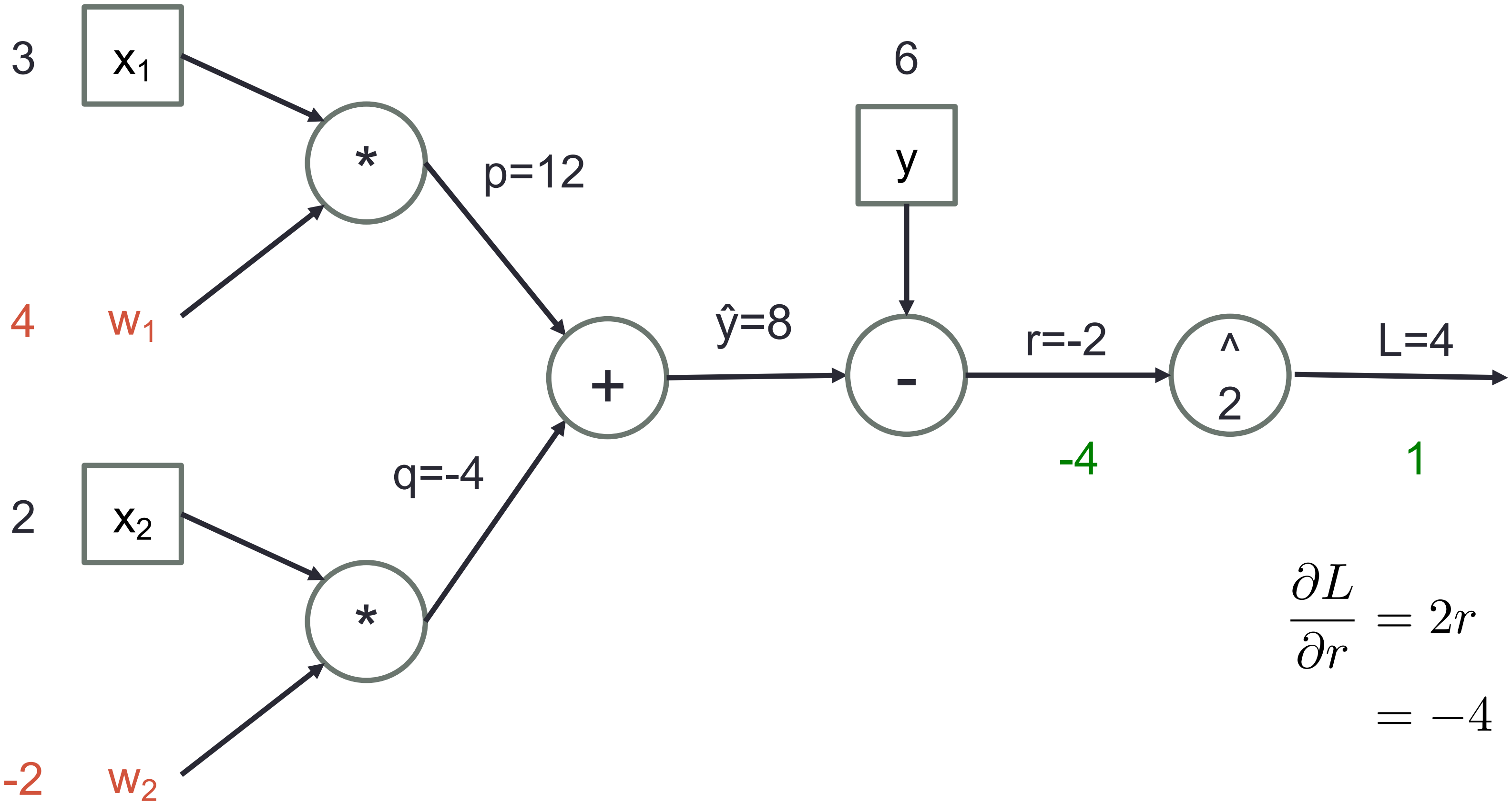
Computational Graph



Computational Graph

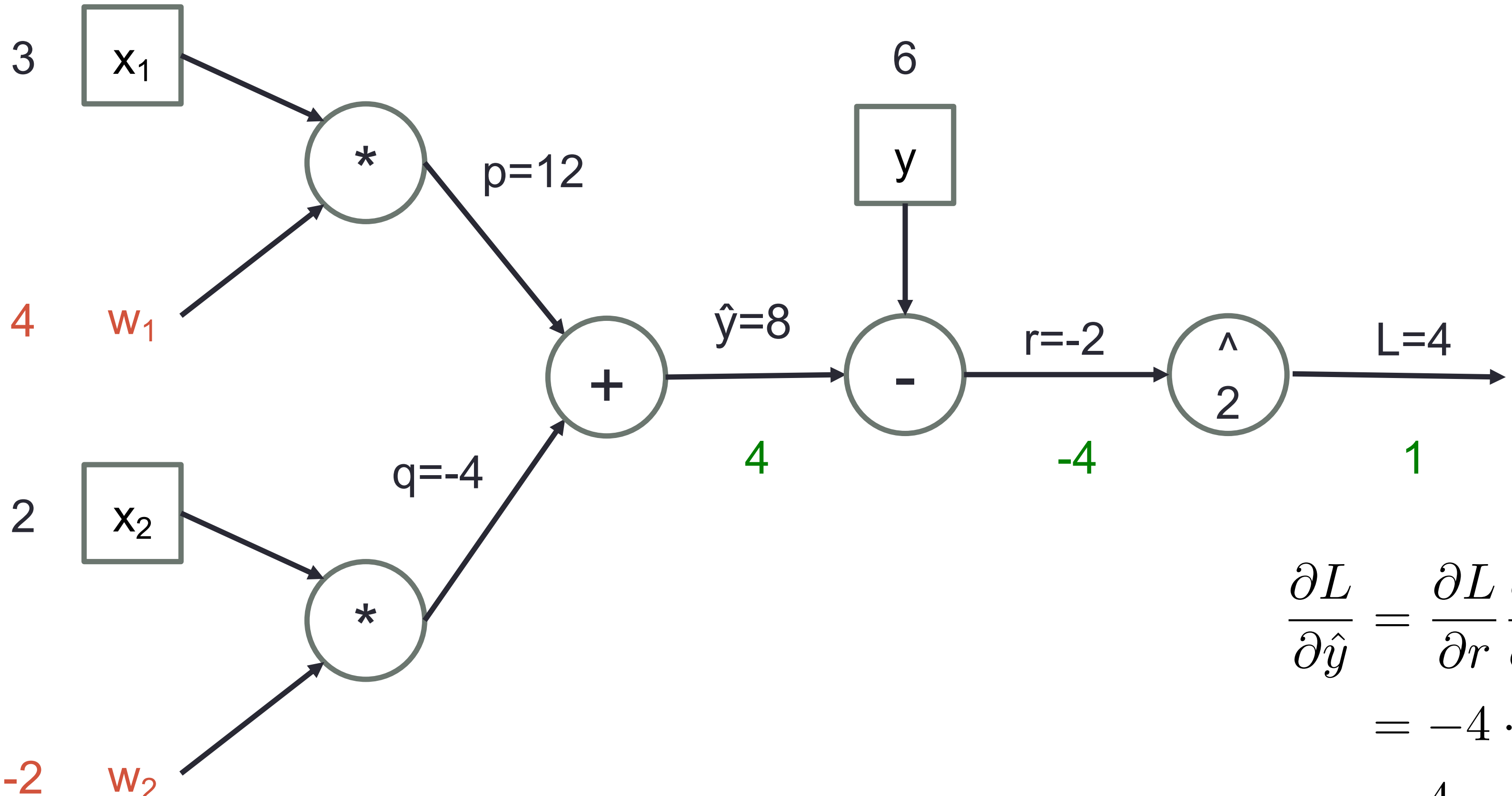


Computational Graph



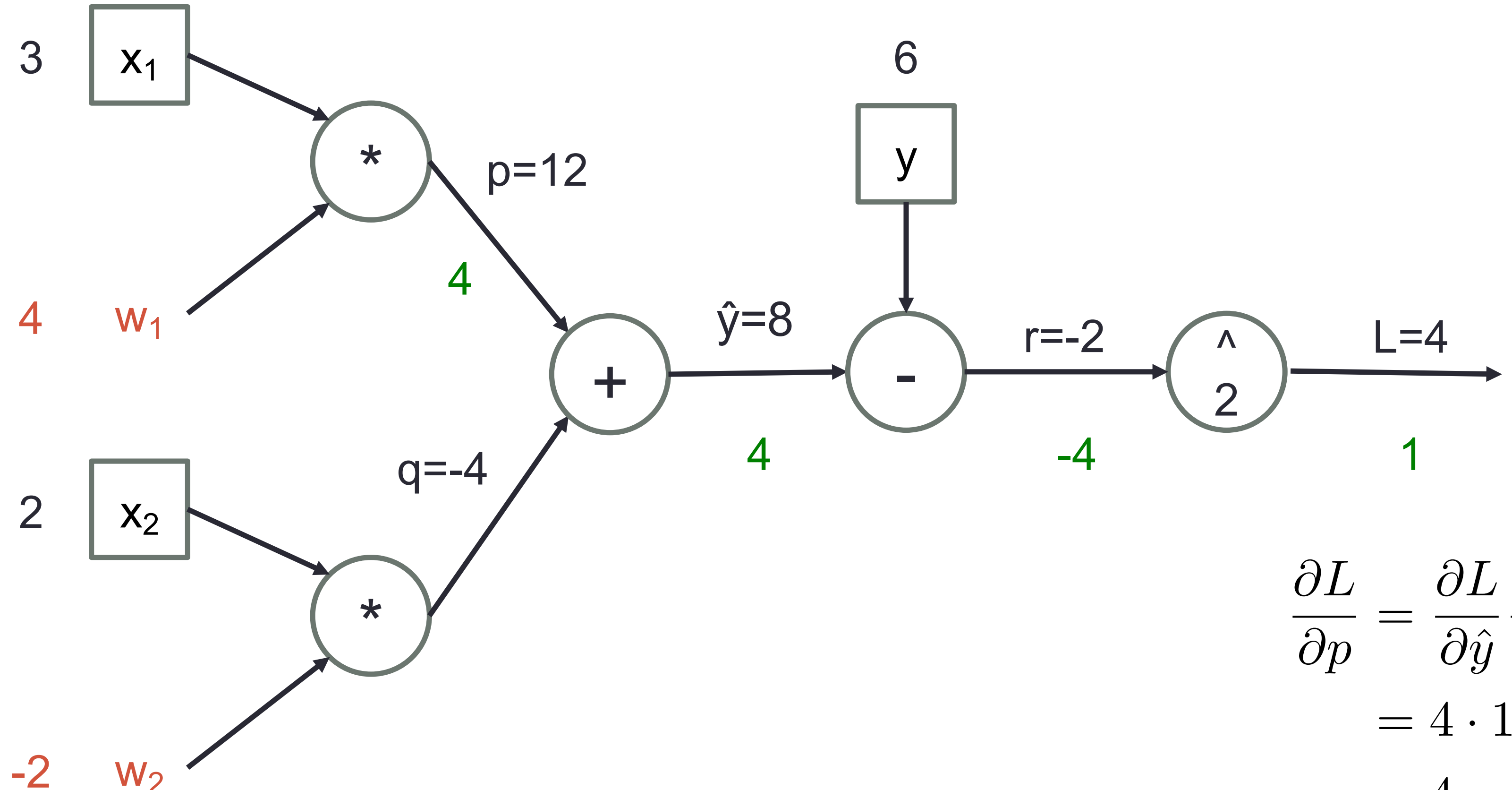
$$\frac{\partial L}{\partial r} = 2r$$
$$= -4$$

Computational Graph



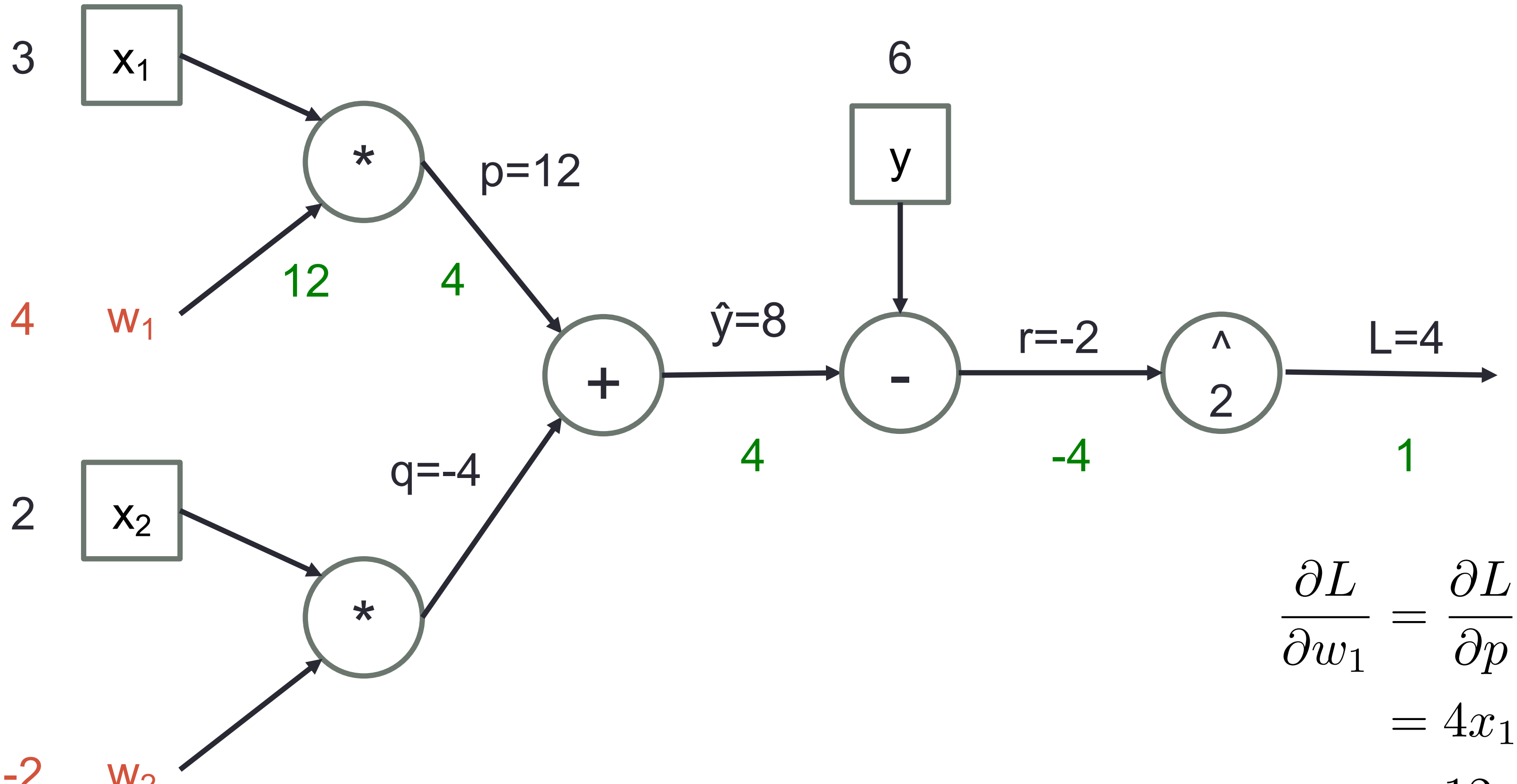
$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial L}{\partial r}\frac{\partial r}{\partial \hat{y}}$$
$$= -4 \cdot -1$$
$$= 4$$

Computational Graph



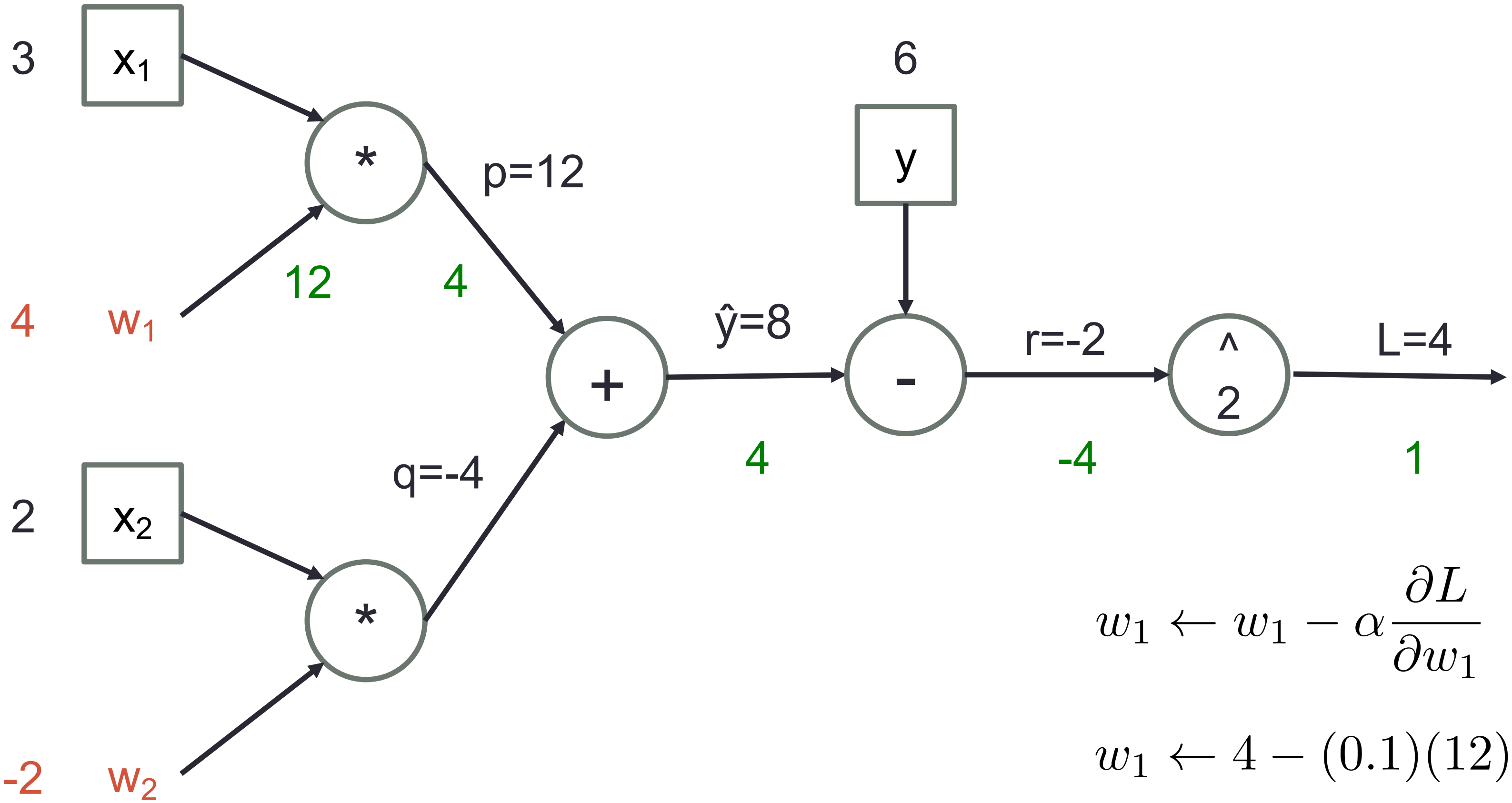
$$\frac{\partial L}{\partial p} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial p}$$
$$= 4 \cdot 1$$
$$= 4$$

Computational Graph



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial w_1}$$
$$= 4x_1$$
$$= 12$$

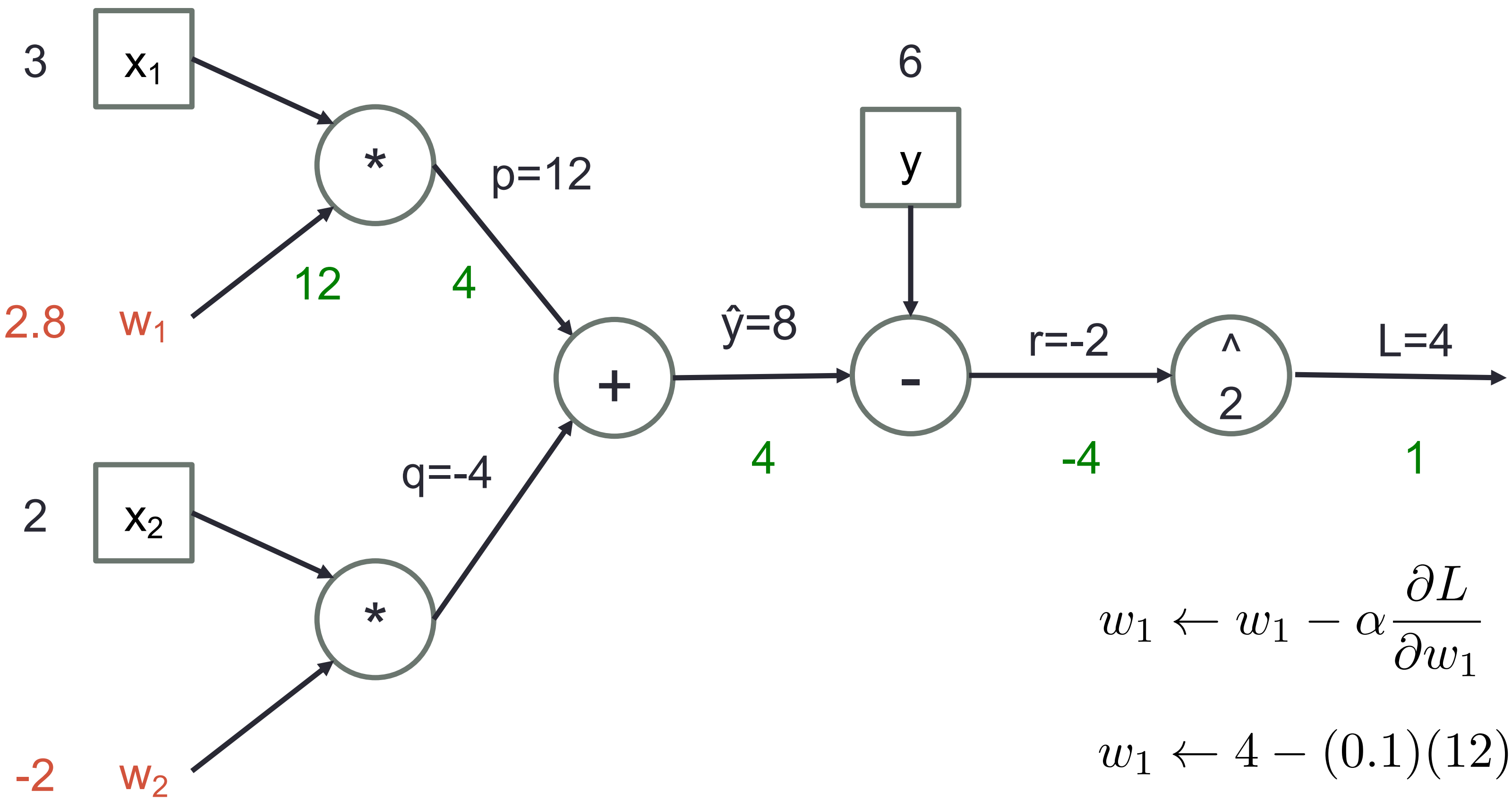
Computational Graph



$$w_1 \leftarrow w_1 - \alpha \frac{\partial L}{\partial w_1}$$

$$w_1 \leftarrow 4 - (0.1)(12) = 2.8$$

Computational Graph



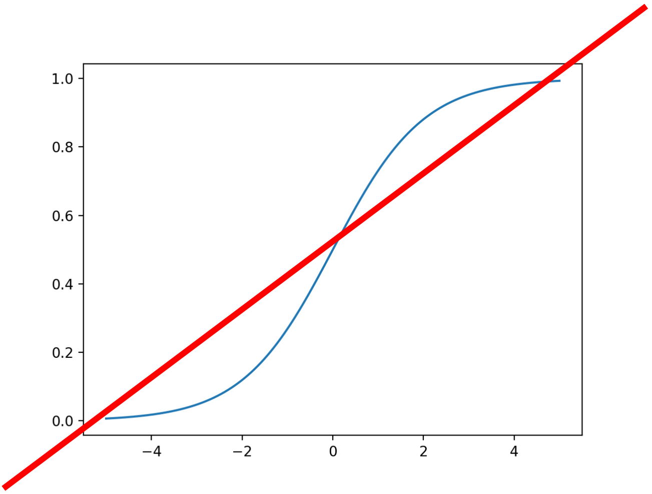
$$w_1 \leftarrow w_1 - \alpha \frac{\partial L}{\partial w_1}$$

$$w_1 \leftarrow 4 - (0.1)(12) = 2.8$$

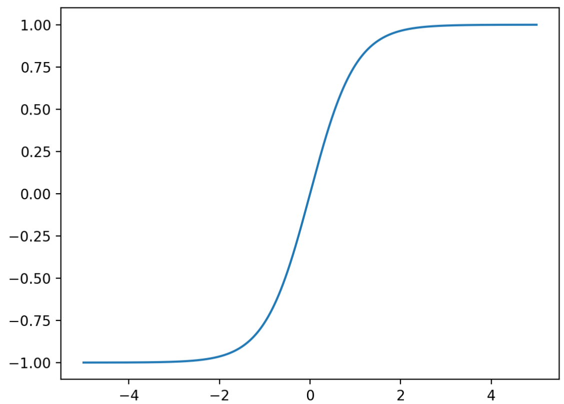
Preparing Your Data

- Shuffle your data
- Mean center your data
  - Why?
- Normalize the variance
  - Why?
- “Whitening”
  - Decorrelates data
  - Can be hit or miss
- When to do train/test split?

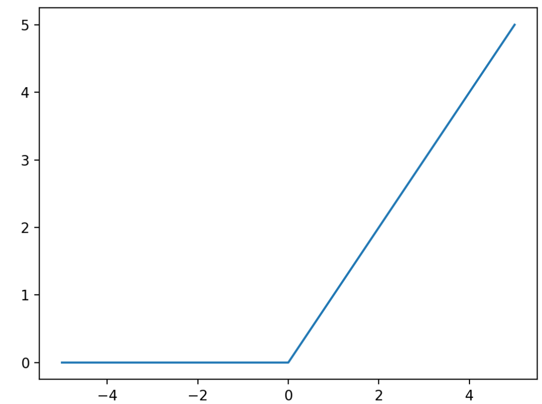
Choosing an Activation Function



$$f(x) = \frac{1}{1 + e^{-x}}$$



$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



$$f(x) = \max(0, x)$$

Initializing Your Network

- Set all weights to 0?
  - Terrible idea
- Set all weights to random values?
  - ***Small*** random values
- State-of-the-art: Xavier or Glorot initialization
  - Takes into account fan-in/fan-out of a neuron when initializing its weights

Optimization Methods

- Stochastic gradient descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla L(\mathbf{w})$$

- Stochastic gradient descent + momentum

$$\mathbf{z} \leftarrow \beta \mathbf{z} + \nabla L(\mathbf{w})$$

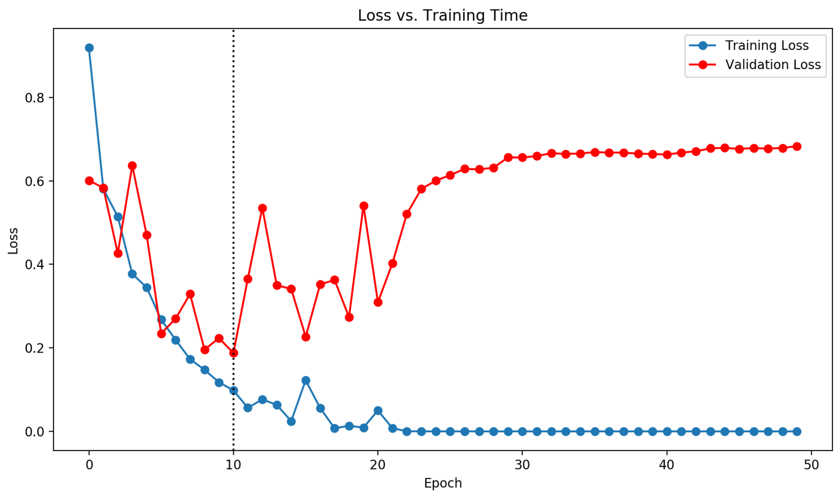
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha z$$

- State-of-the-art approaches:
  - RMSProp
  - Adam

Regularization

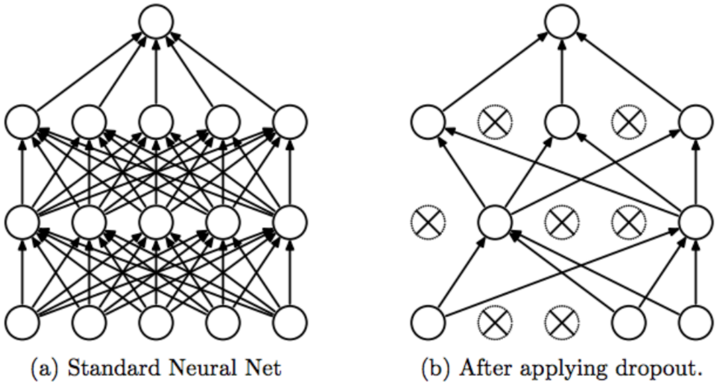
Classical Approaches

- Weight decay
  - L2 term
- Early stopping



Modern Approaches

- Dropout



- Batch normalization