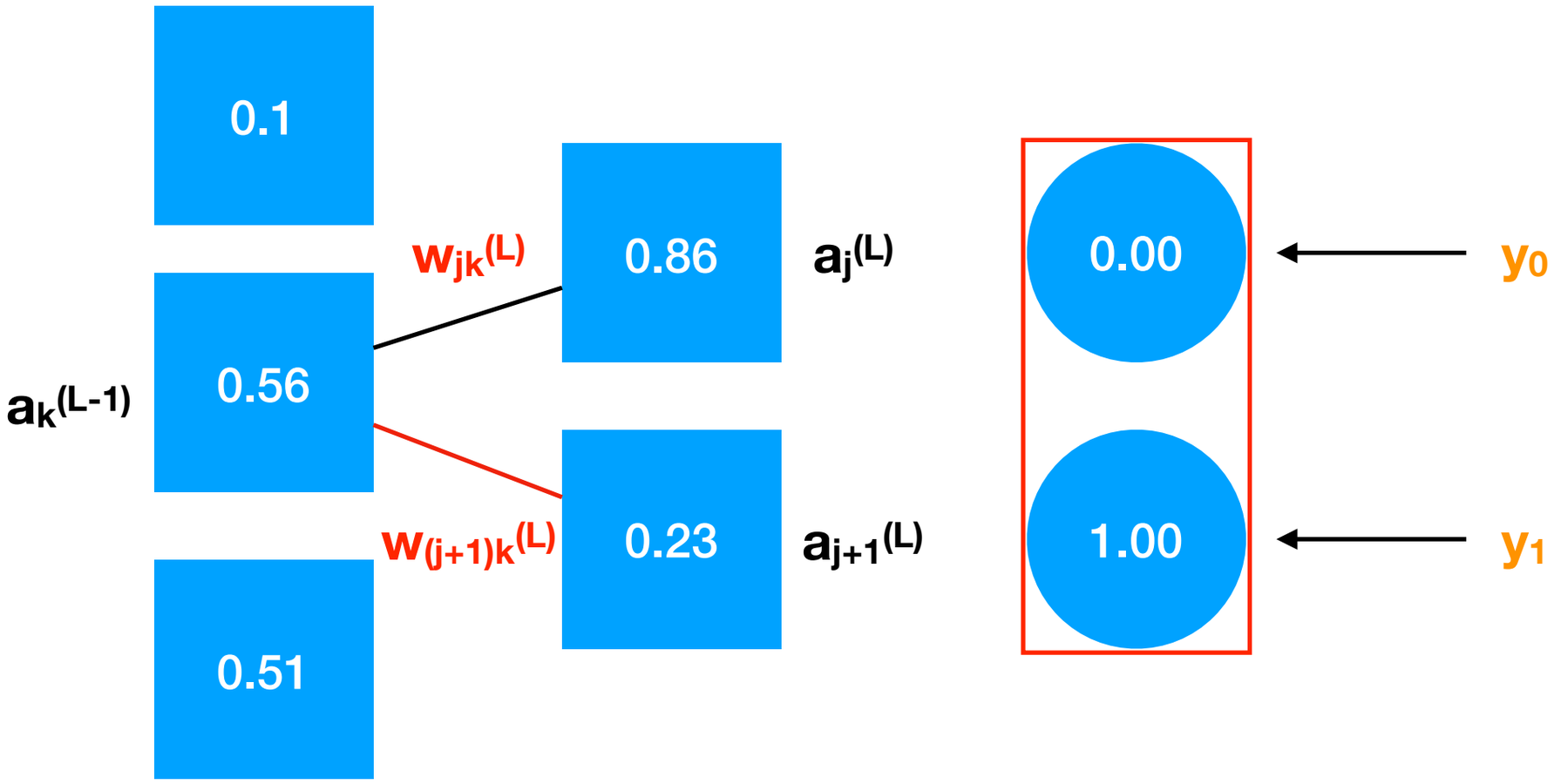
Backpropagation

Bastien Arcelin, Alexandre Boucaud
Workshop GPU - CC IN2P3

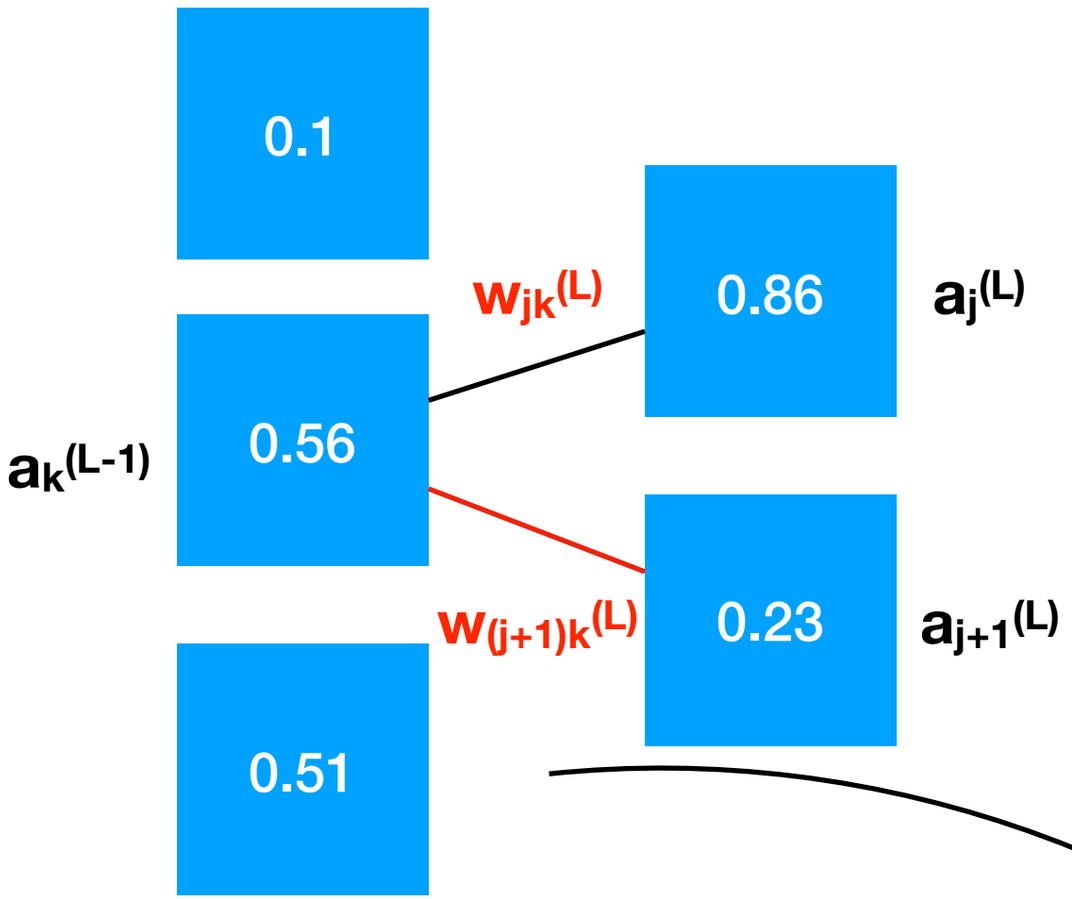
Gradient descent

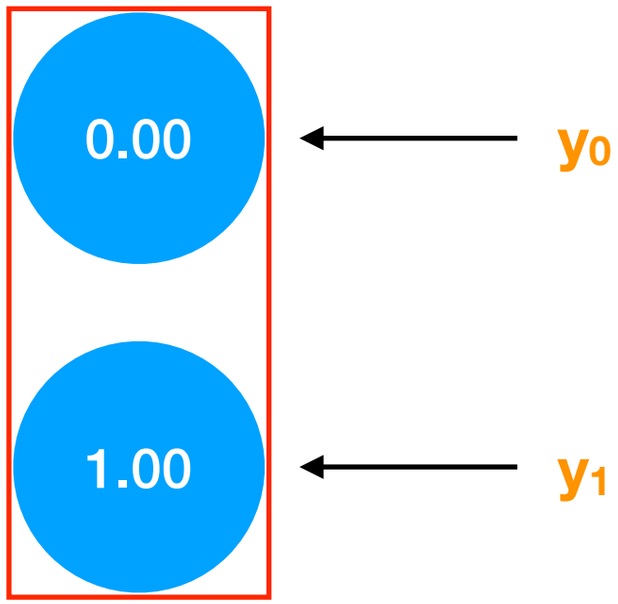


$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

Case:
- training on one element of the training sample
- batch_size = 1

Gradient descent

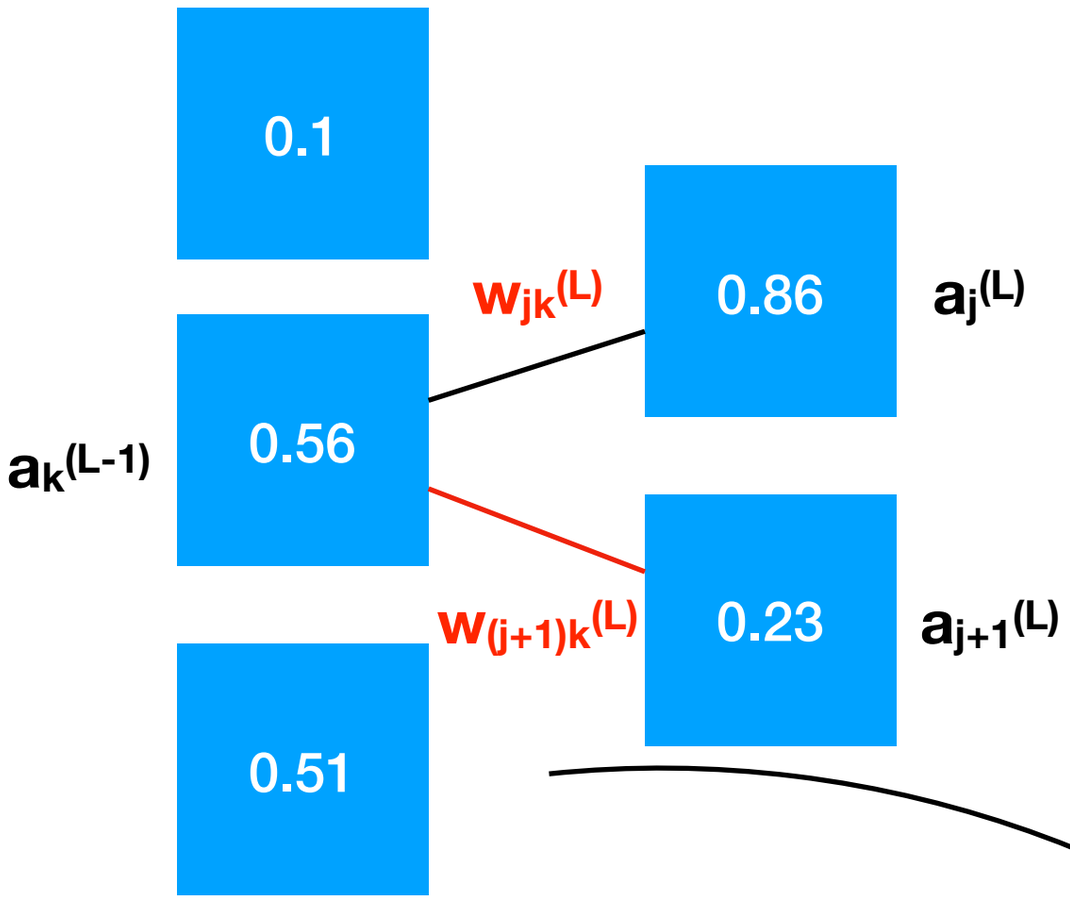


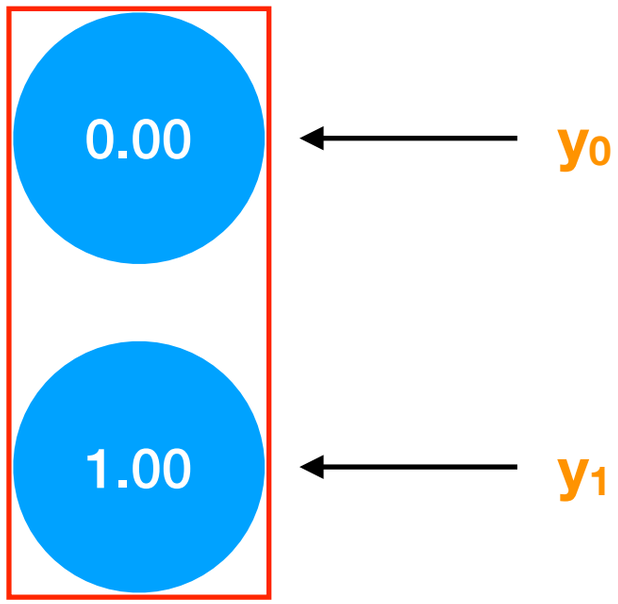


$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

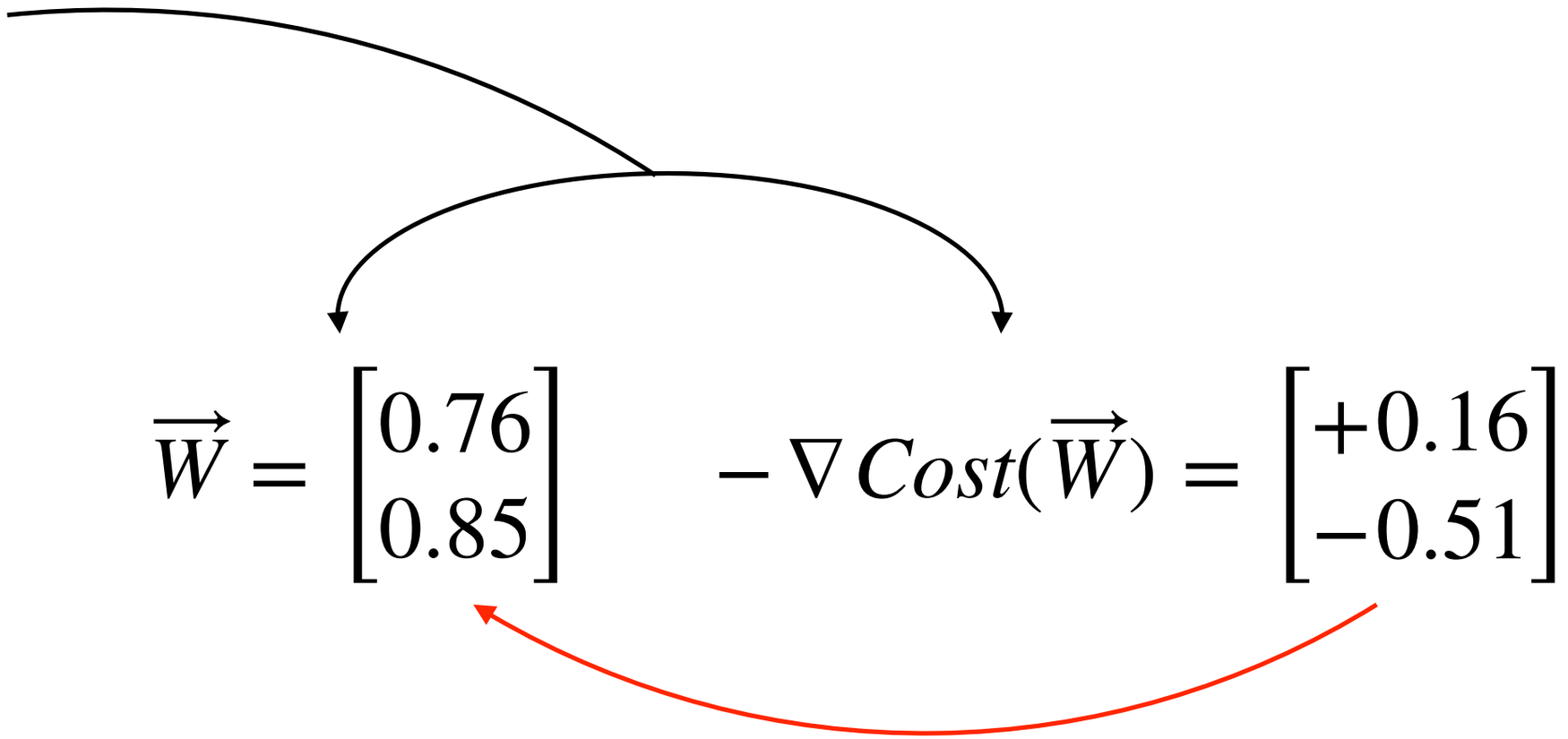
$$\vec{W} = \begin{bmatrix} 0.76 \\ 0.85 \end{bmatrix} \qquad -\nabla Cost(\vec{W}) = \begin{bmatrix} +0.16 \\ -0.51 \end{bmatrix}$$

Gradient descent

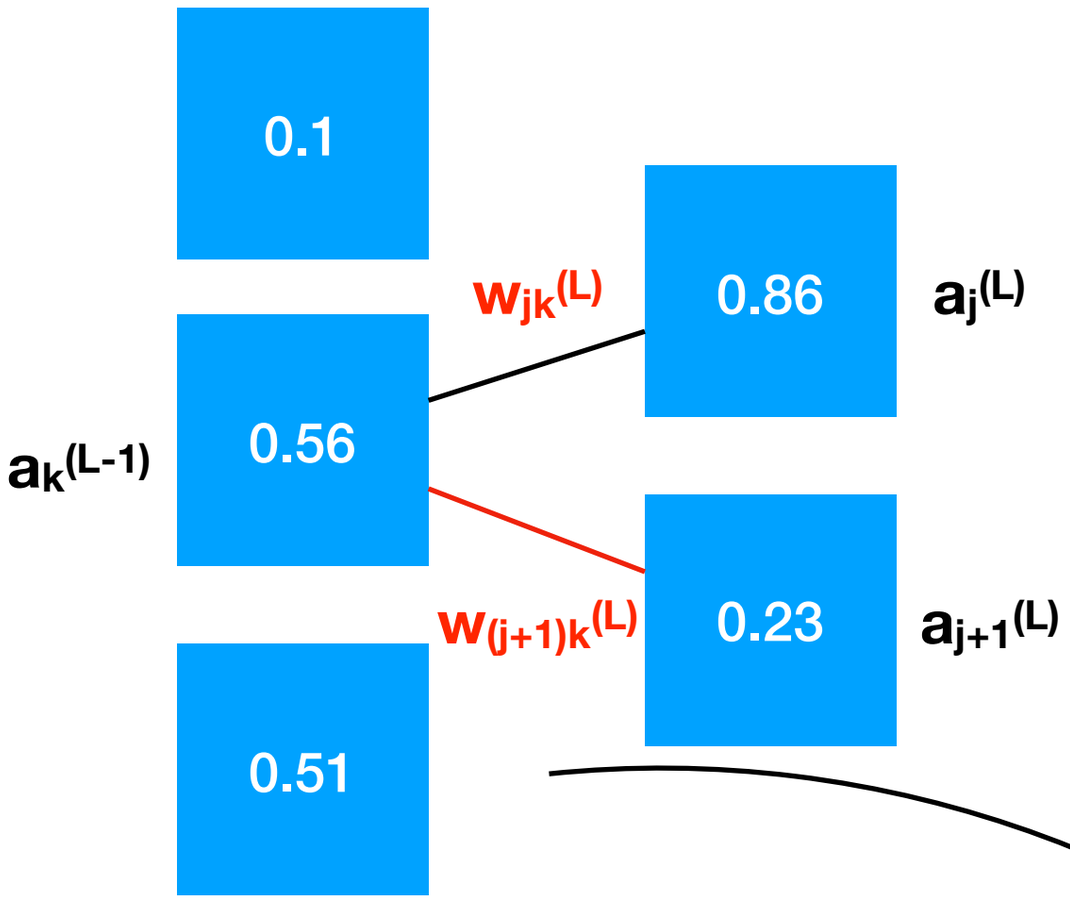


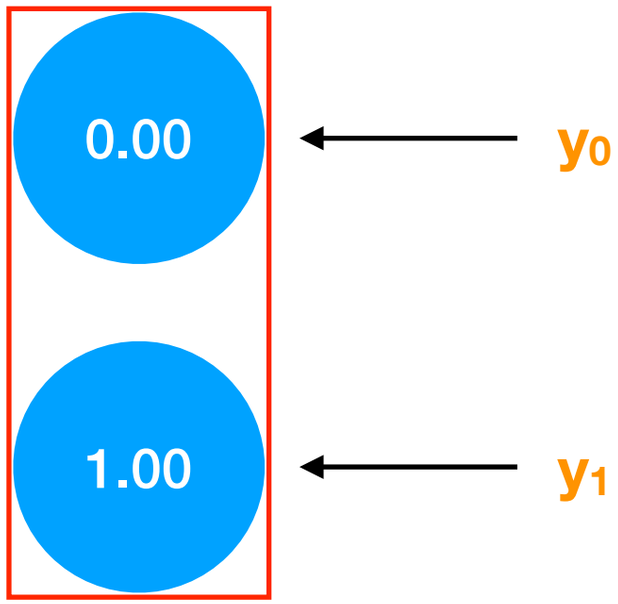


$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$



Gradient descent

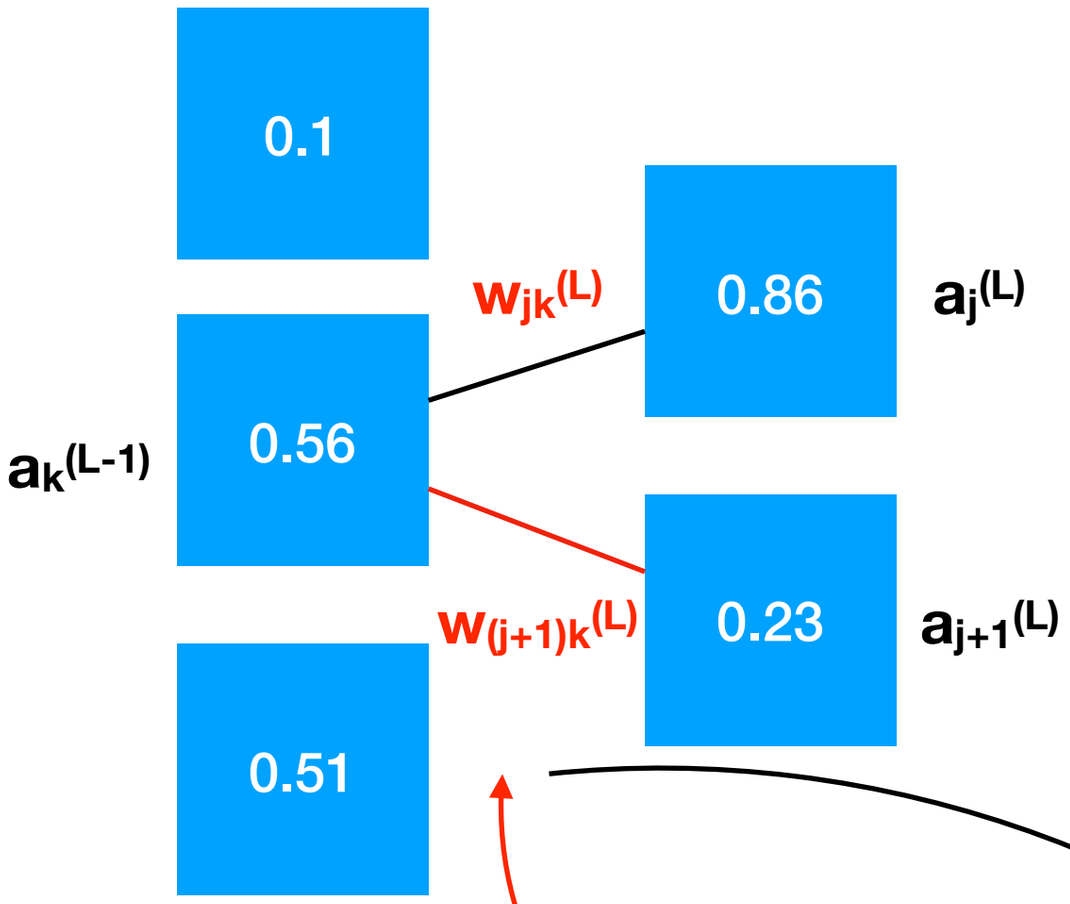


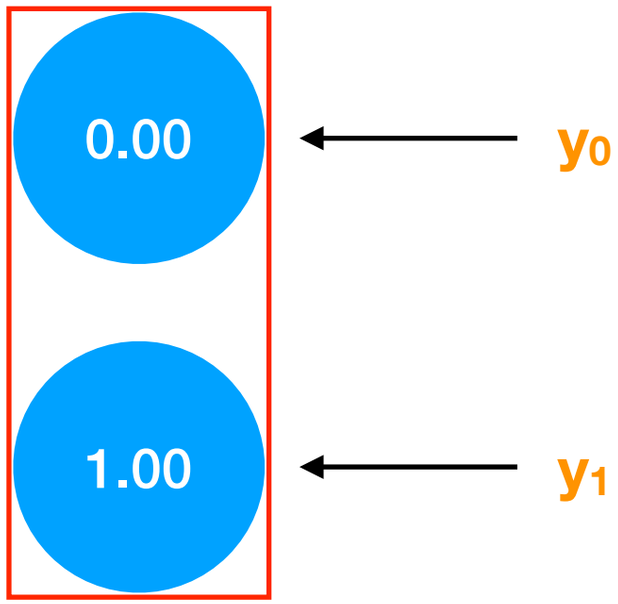


$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

$$\vec{W} = \begin{bmatrix} 0.92 \\ 0.34 \end{bmatrix} \qquad -\nabla Cost(\vec{W}) = \begin{bmatrix} +0.16 \\ -0.51 \end{bmatrix}$$

Gradient descent

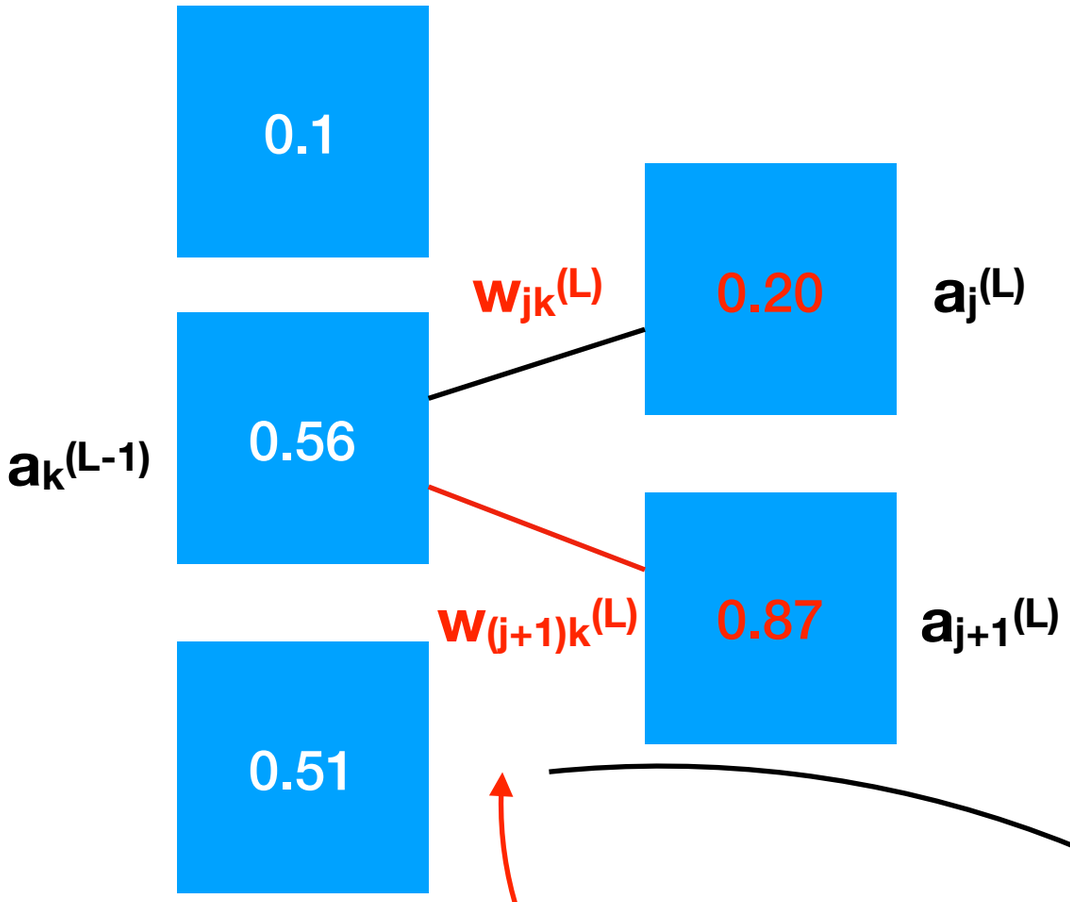


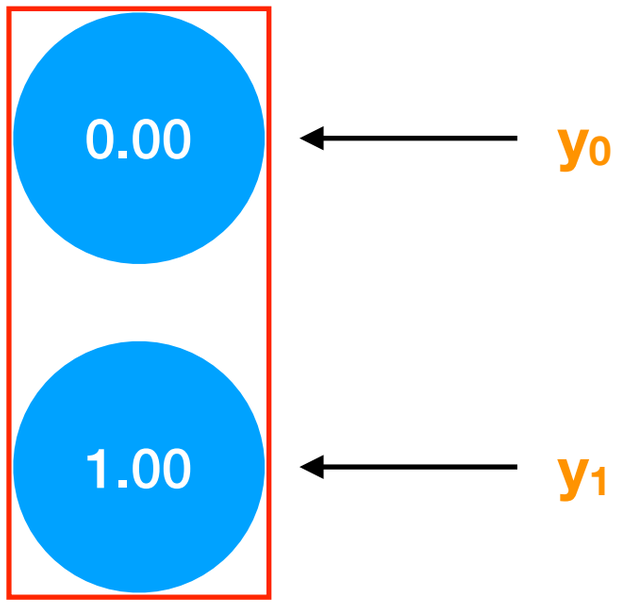


$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

$$\vec{W} = \begin{bmatrix} 0.92 \\ 0.34 \end{bmatrix} \qquad -\nabla Cost(\vec{W}) = \begin{bmatrix} +0.16 \\ -0.51 \end{bmatrix}$$

Gradient descent

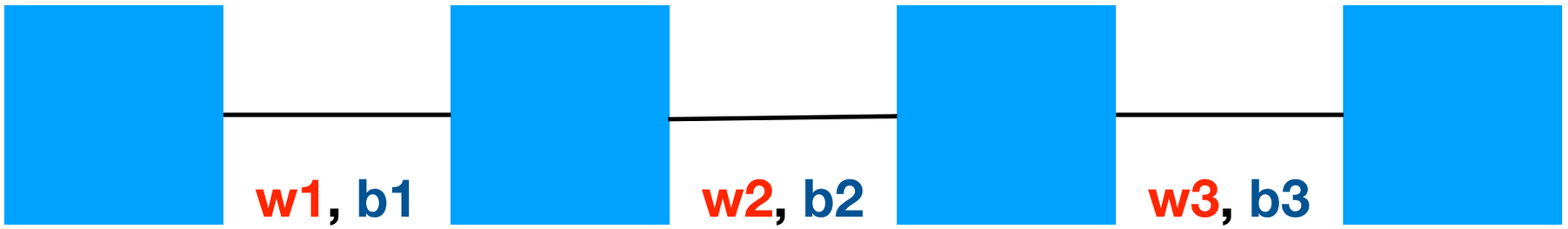




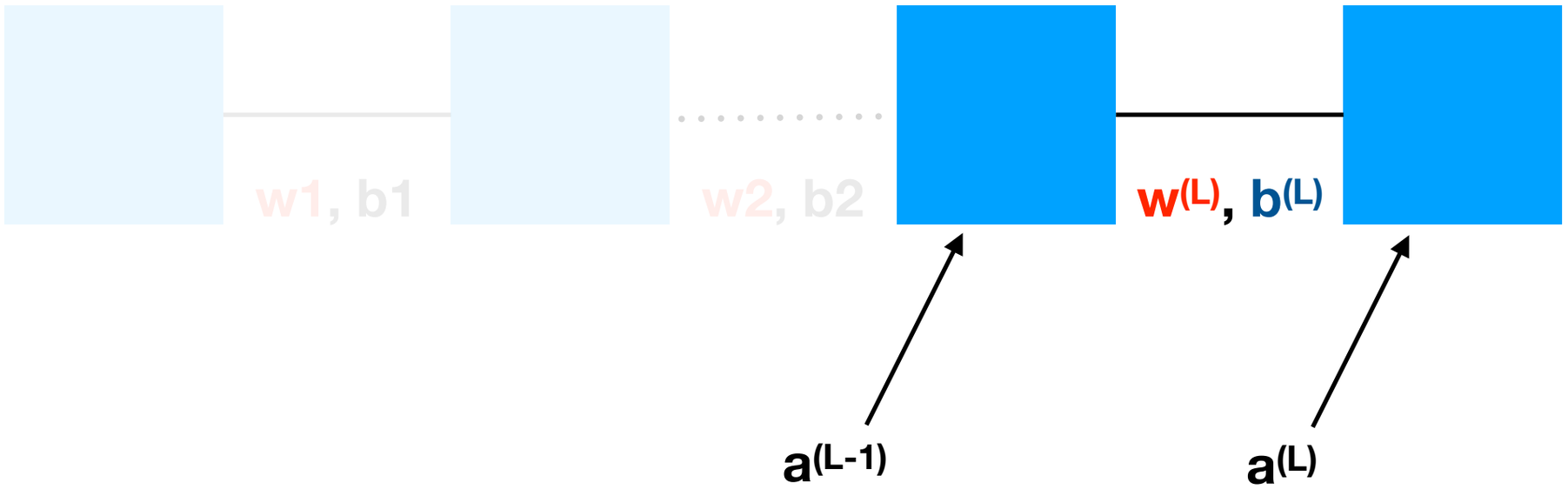
$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

$$\vec{W} = \begin{bmatrix} 0.92 \\ 0.34 \end{bmatrix} \qquad -\nabla Cost(\vec{W}) = \begin{bmatrix} +0.16 \\ -0.51 \end{bmatrix}$$

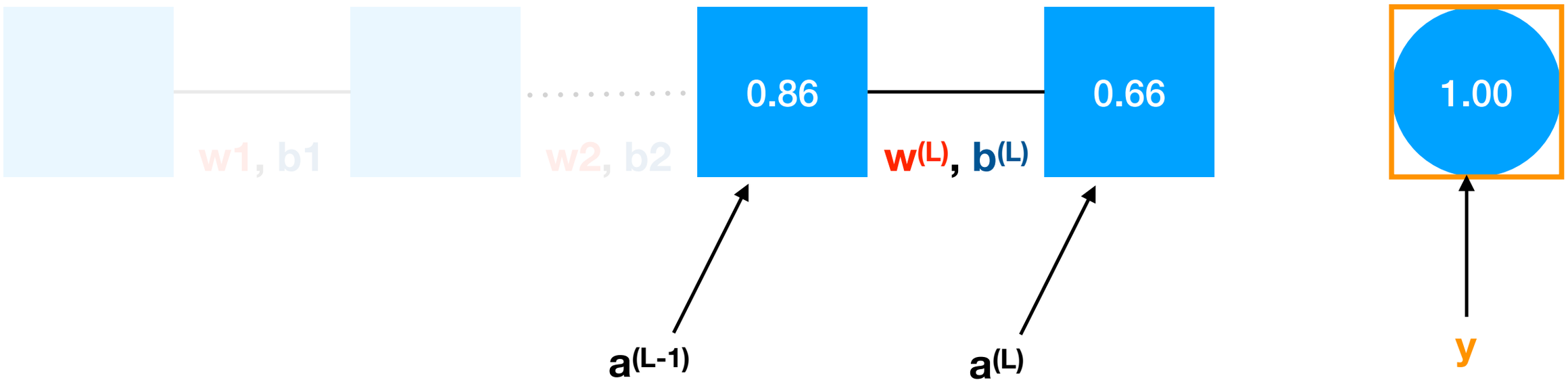
Network where each layer has a single neuron in it



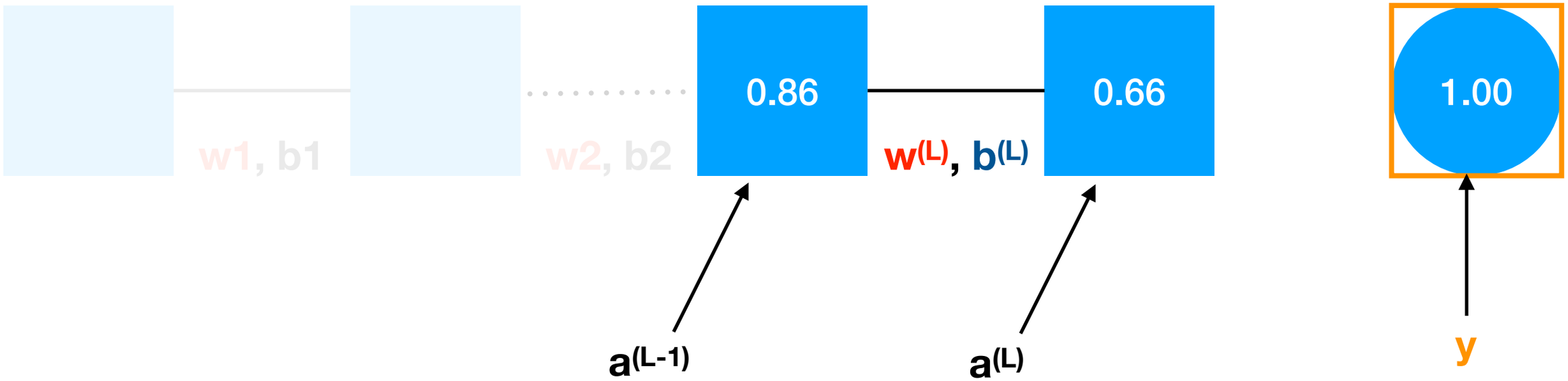
Network where each layer has a single neuron in it



Network where each layer has a single neuron in it

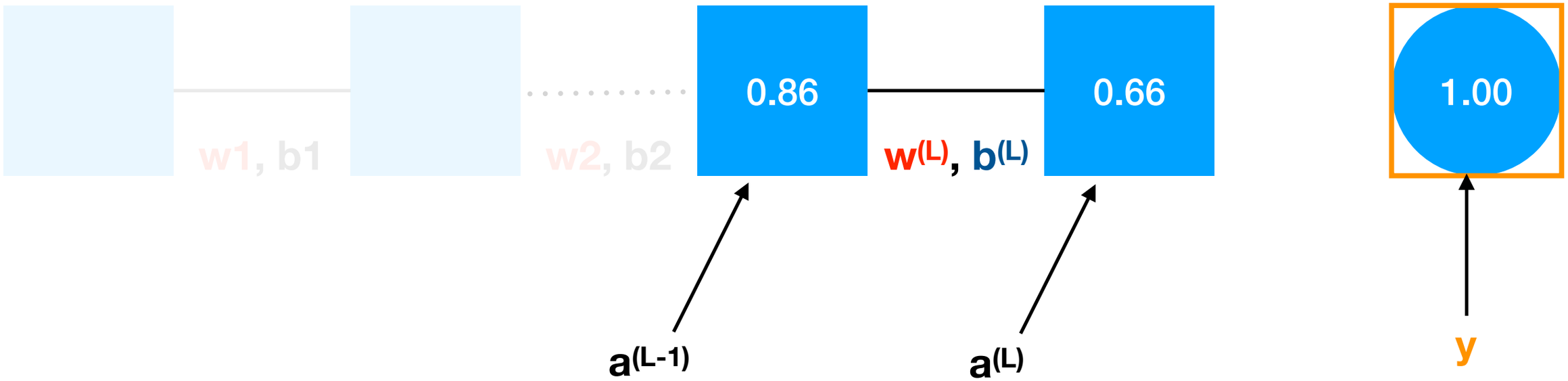


Network where each layer has a single neuron in it



$$\text{Cost}\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

Network where each layer has a single neuron in it

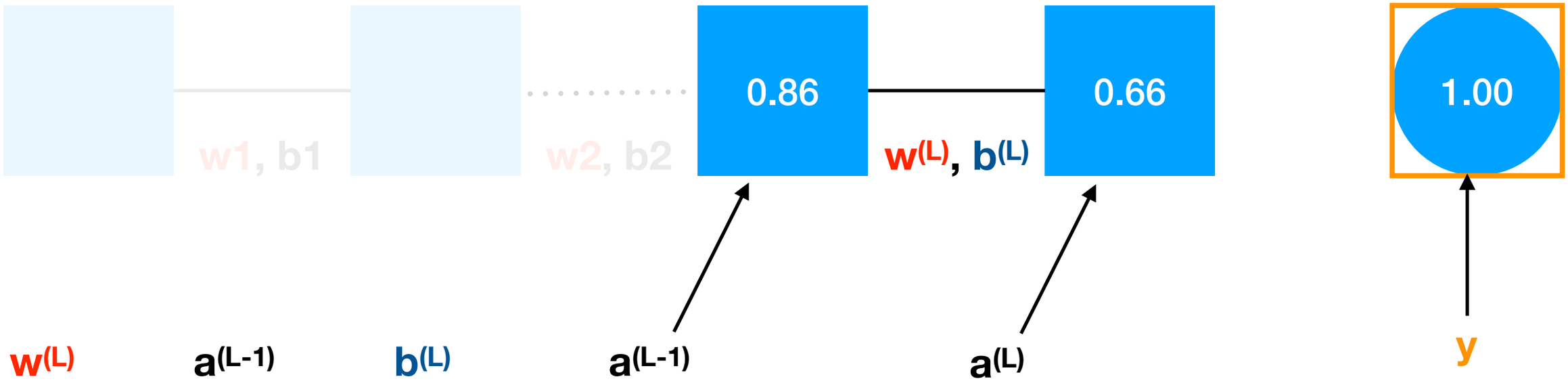


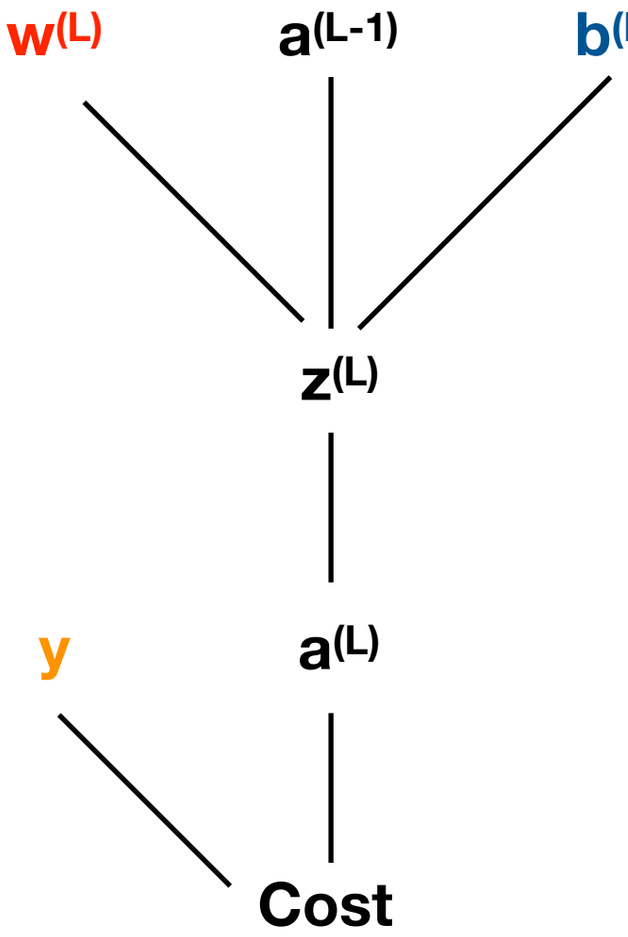
$$\text{Cost}\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f\left(w^{(L)} * a^{(L-1)} + b^{(L)} \right) = f\left(z^{(L)} \right)$$

f: activation function

Network where each layer has a single neuron in it



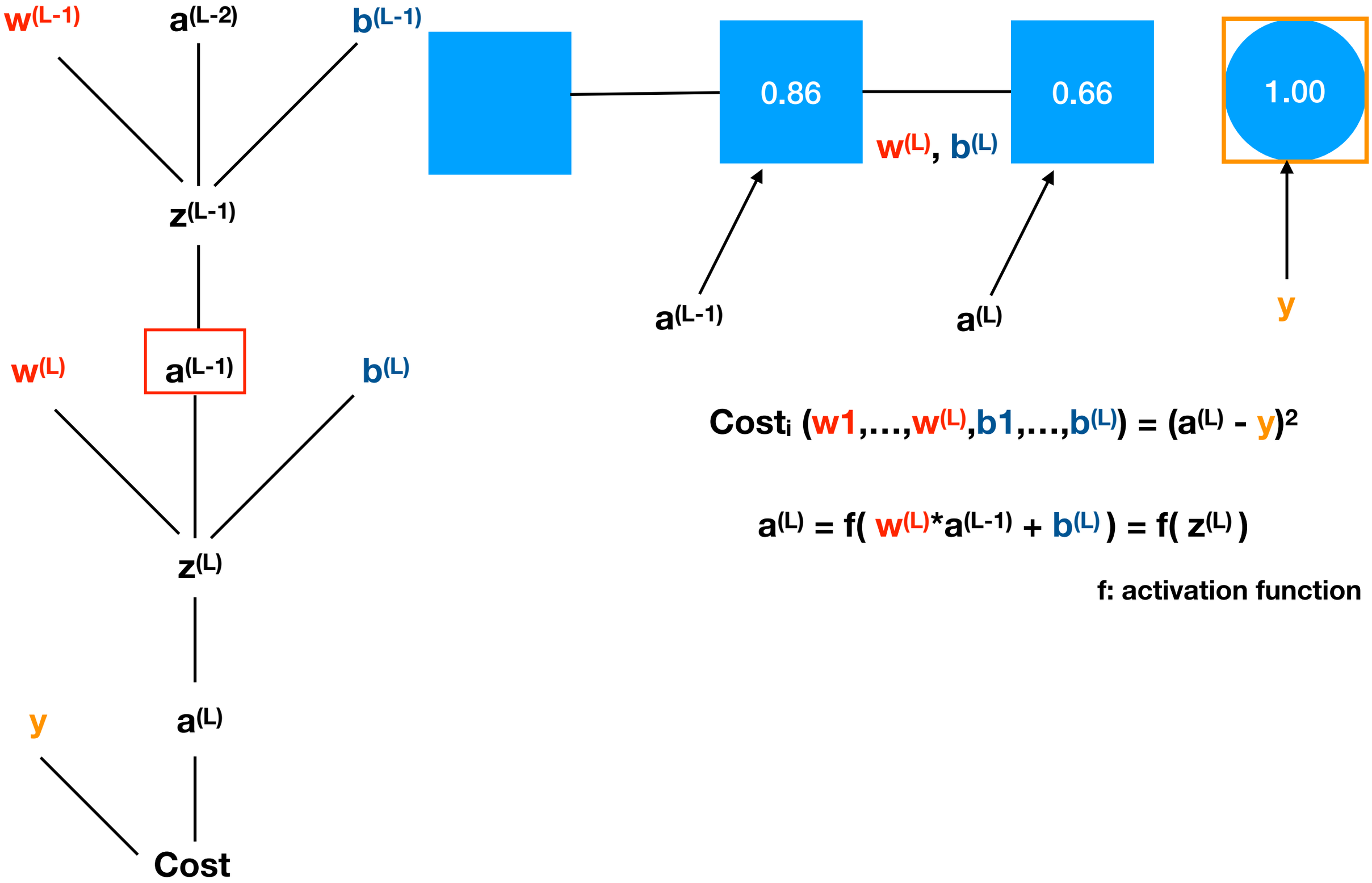


$$\text{Cost}_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

f: activation function

Network where each layer has a single neuron in it

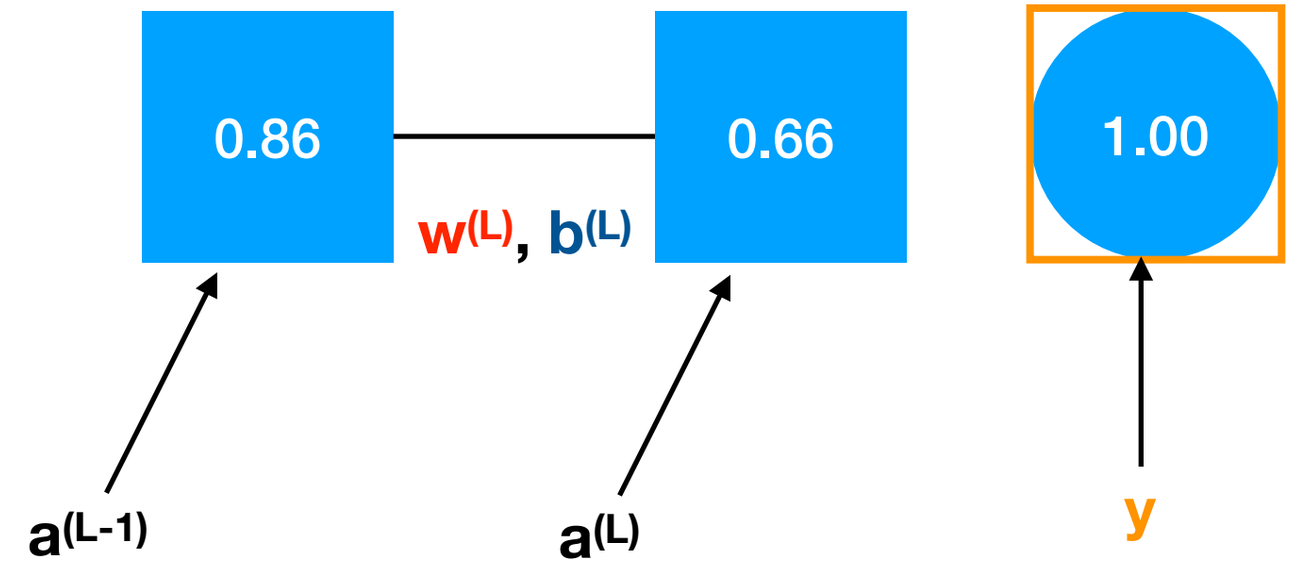


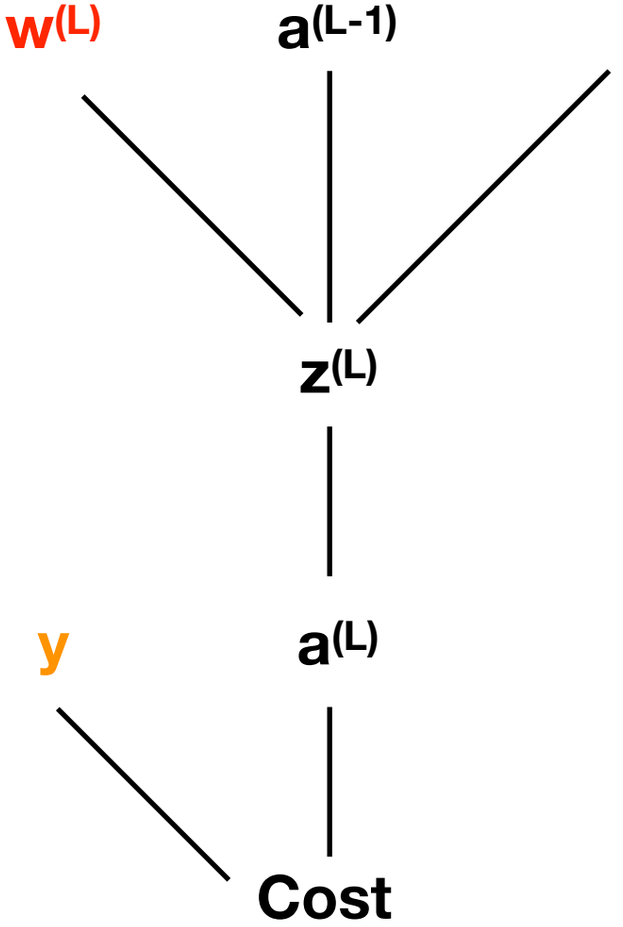
$$Cost_i\ (w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f(\ w^{(L)} * a^{(L-1)} + b^{(L)}\ ) = f(\ z^{(L)}\ )$$

f: activation function

Network where each layer has a single neuron in it



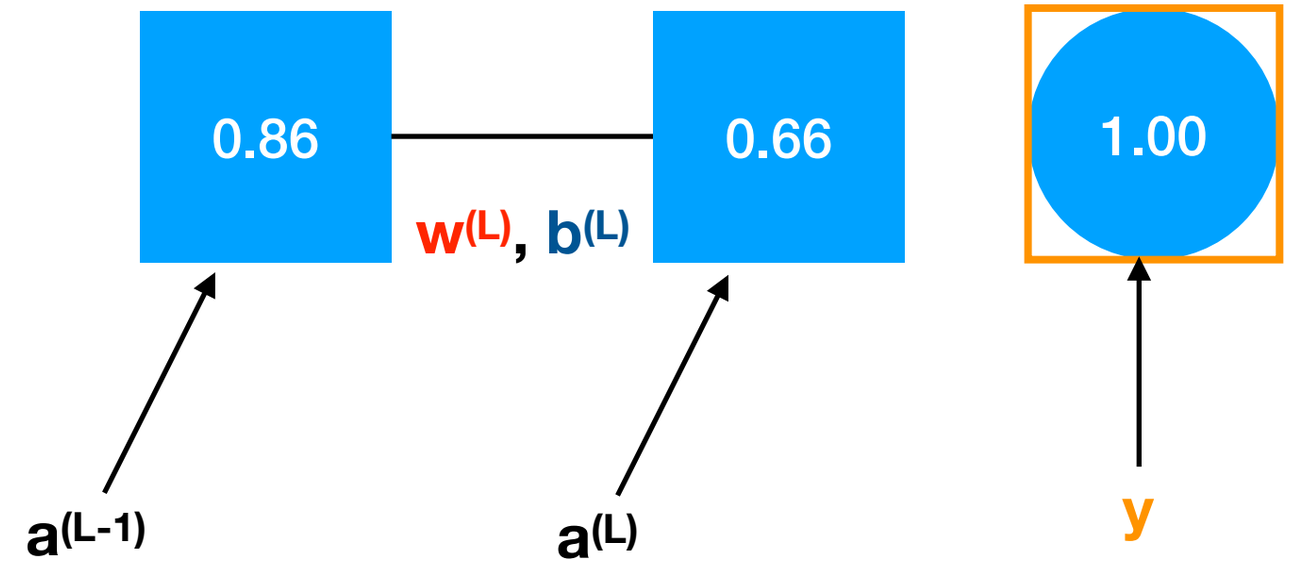


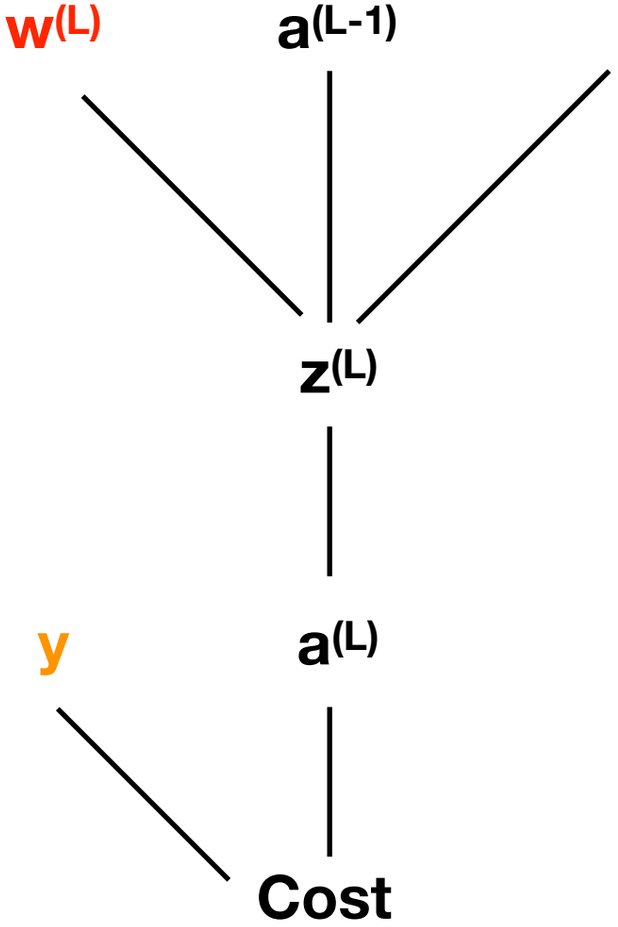
$$\text{Cost}_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

f: activation function

Network where each layer has a single neuron in it





$$Cost_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

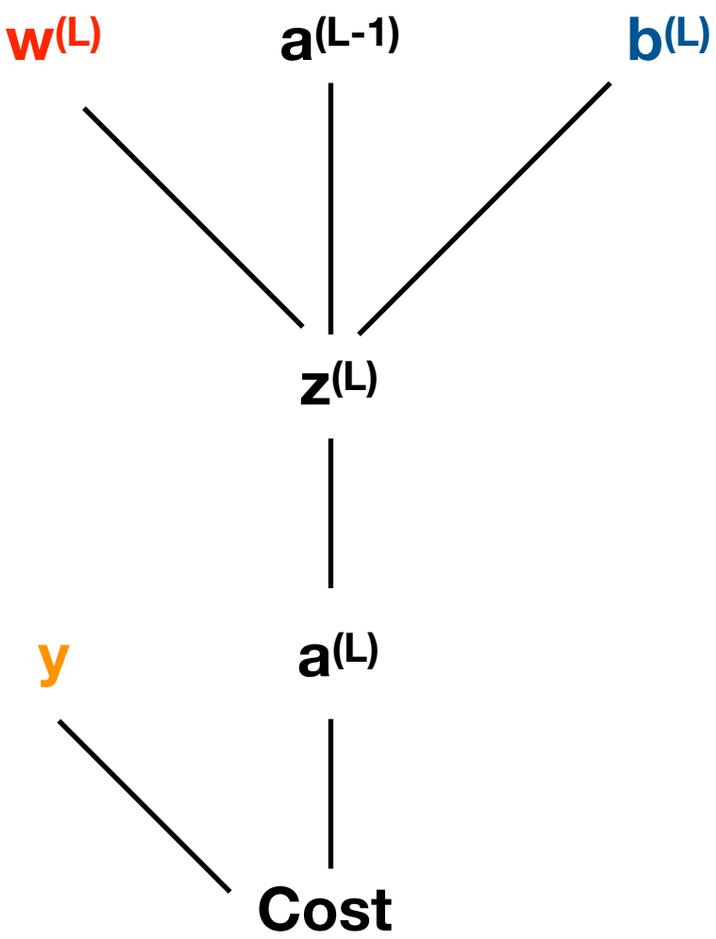
$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

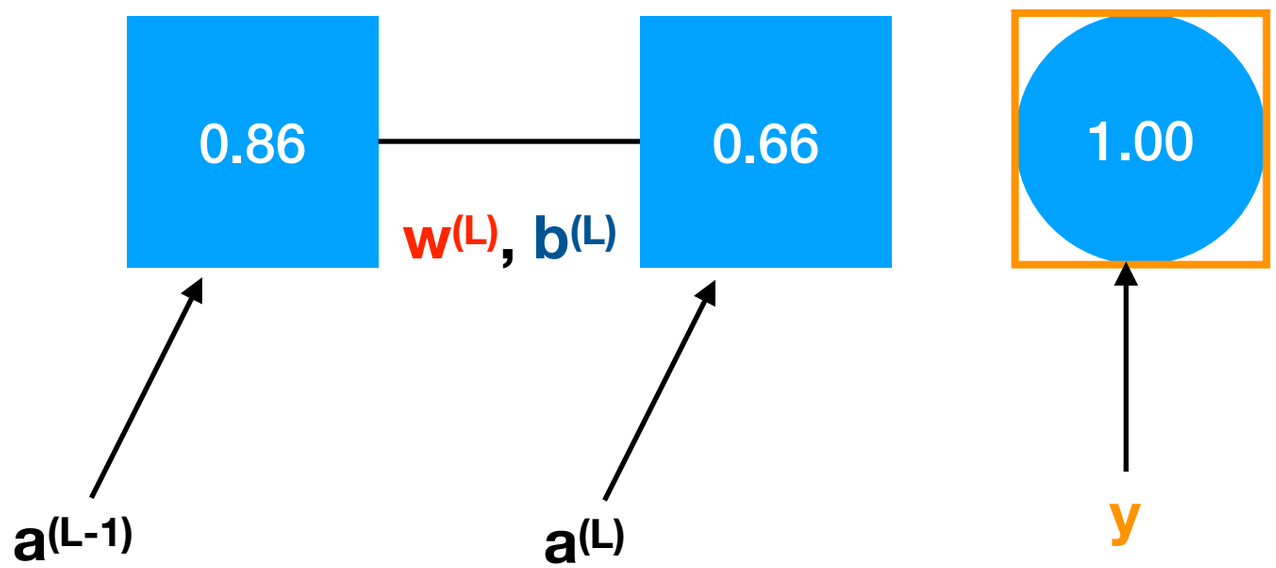
f: activation function

Chain Rule :

$$\frac{\partial Cost_i}{\partial w^{(L)}} = \frac{\partial Cost_i}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}$$

Network where each layer has a single neuron in it





$\text{Cost}_i\ (w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$

$a^{(L)} = f(\ w^{(L)} * a^{(L-1)} + b^{(L)}\ ) = f(\ z^{(L)}\ )$

f: activation function

Chain Rule :

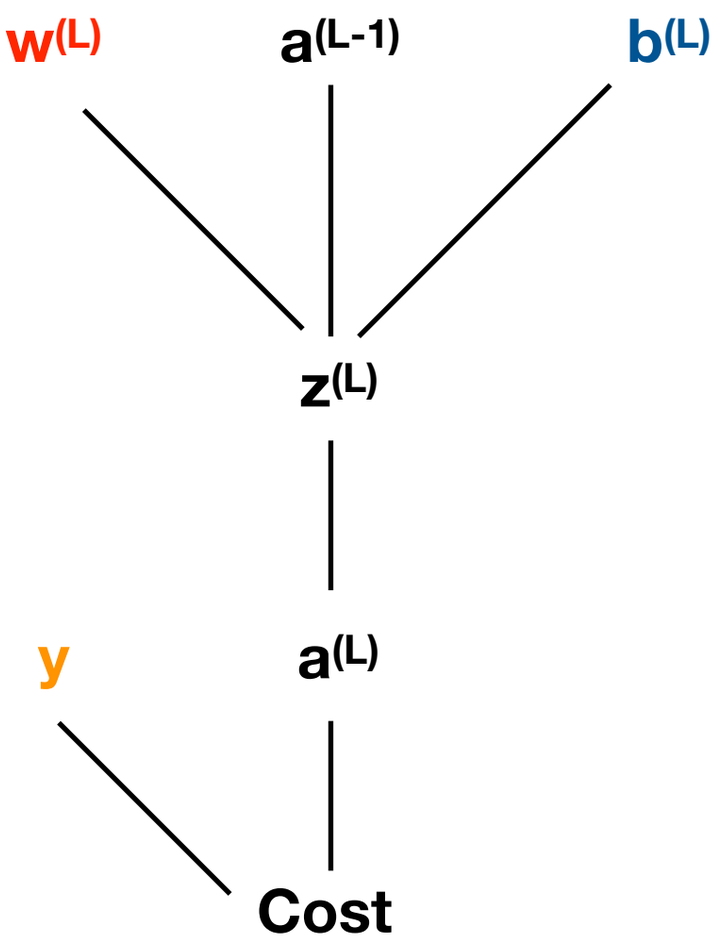
$$\frac{\partial Cost_i}{\partial w^{(L)}} = \frac{\partial Cost_i}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}$$

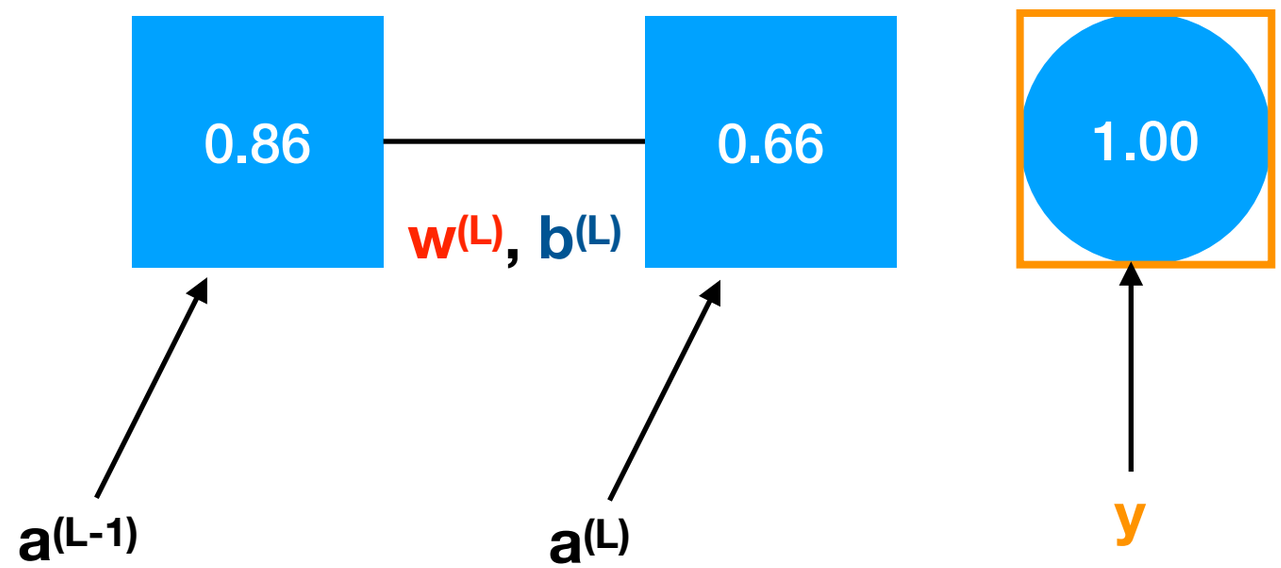
$$\frac{\partial Cost_i}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = f'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

Network where each layer has a single neuron in it





$$Cost_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

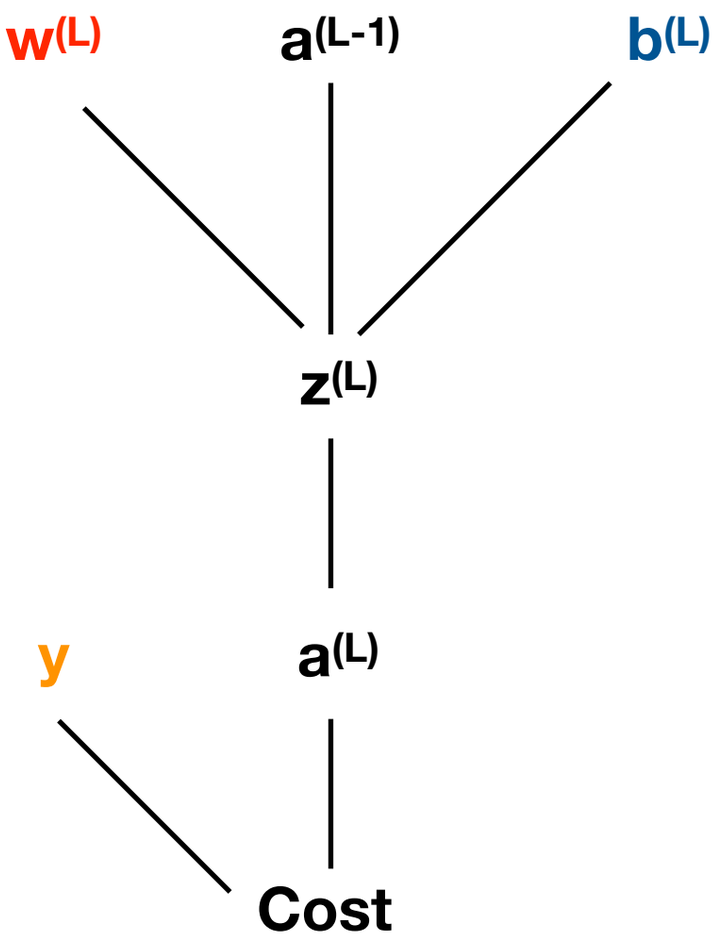
$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

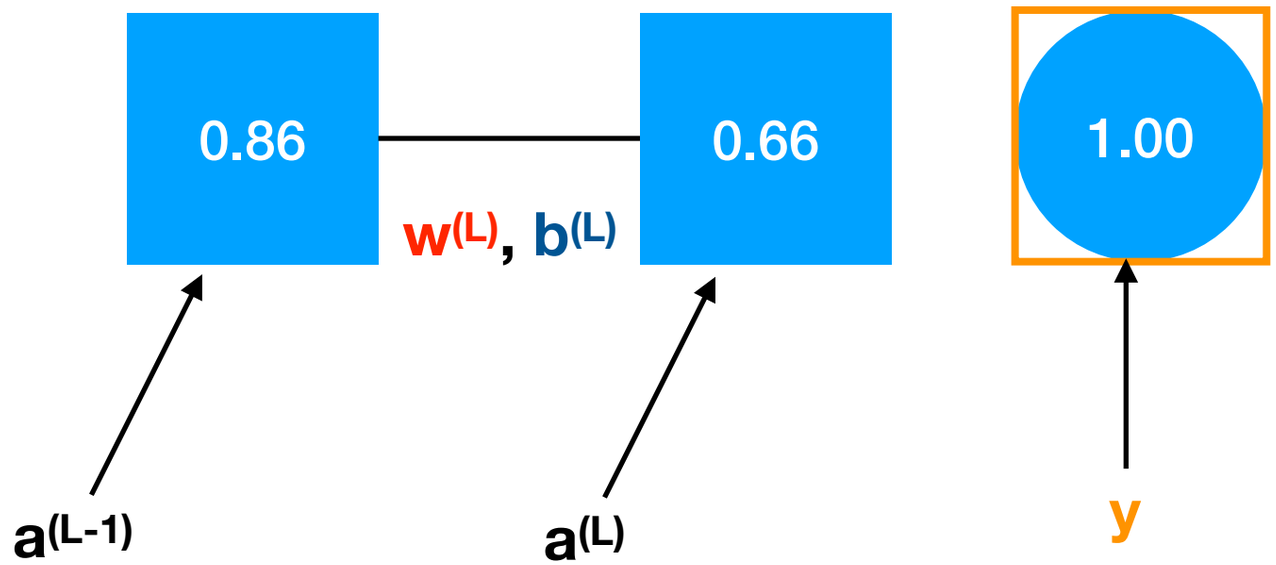
f: activation function

Chain Rule :

$$\frac{\partial Cost_i}{\partial w^{(L)}} = \frac{\partial Cost_i}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}} = 2(a^{(L)} - y) f'(z^{(L)}) a^{(L-1)}$$

Network where each layer has a single neuron in it





$$Cost_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

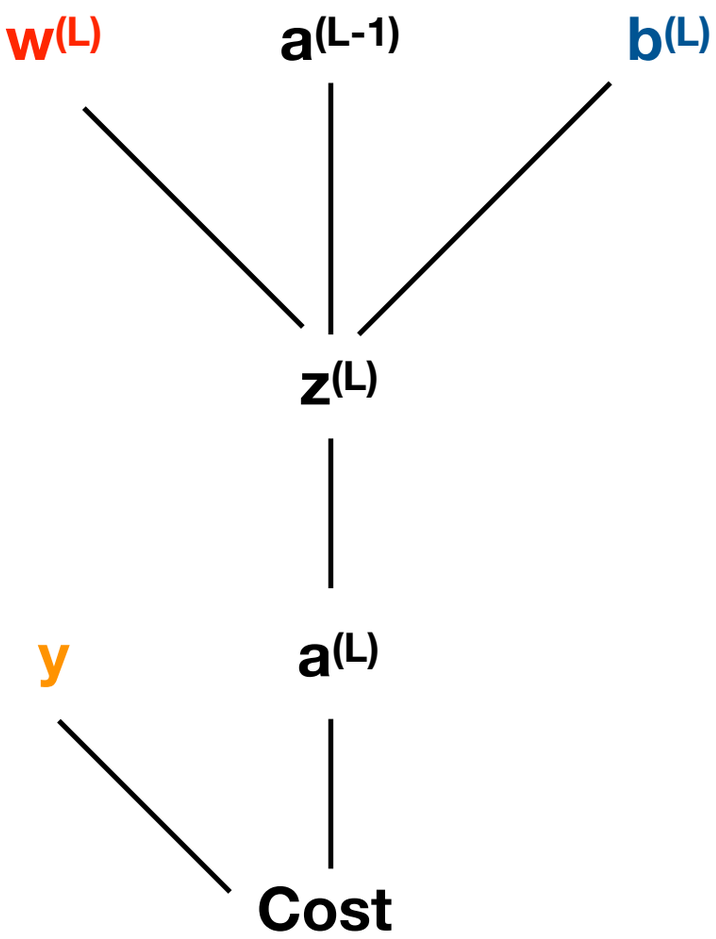
f: activation function

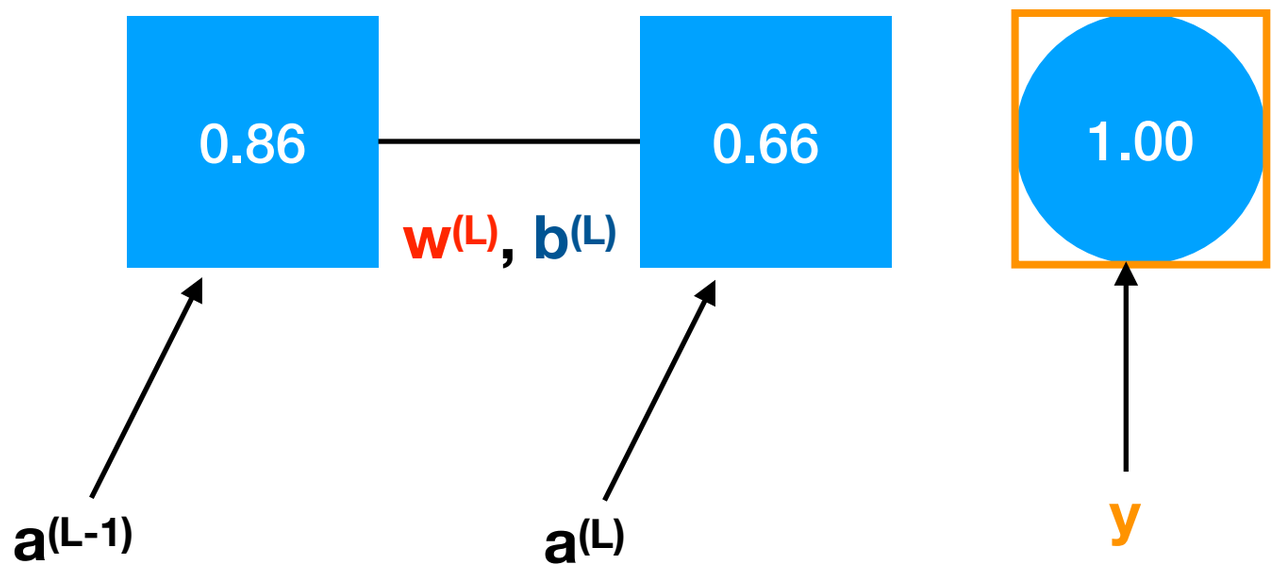
Chain Rule :

For n training examples

$$\frac{\partial Cost_i}{\partial w^{(L)}} = \frac{\partial Cost_i}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}} = 2(a^{(L)} - y)f'(z^{(L)})a^{(L-1)} \longrightarrow \frac{\partial Cost}{\partial w^{(L)}} = \frac{1}{n}\sum_{i=0}^{n-1}\left(\frac{\partial Cost_i}{\partial w^{(L)}}\right)$$

Network where each layer has a single neuron in it





$$Cost_i\ (w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

$$a^{(L)} = f(\ w^{(L)} * a^{(L-1)} + b^{(L)}\ ) = f(\ z^{(L)}\ )$$

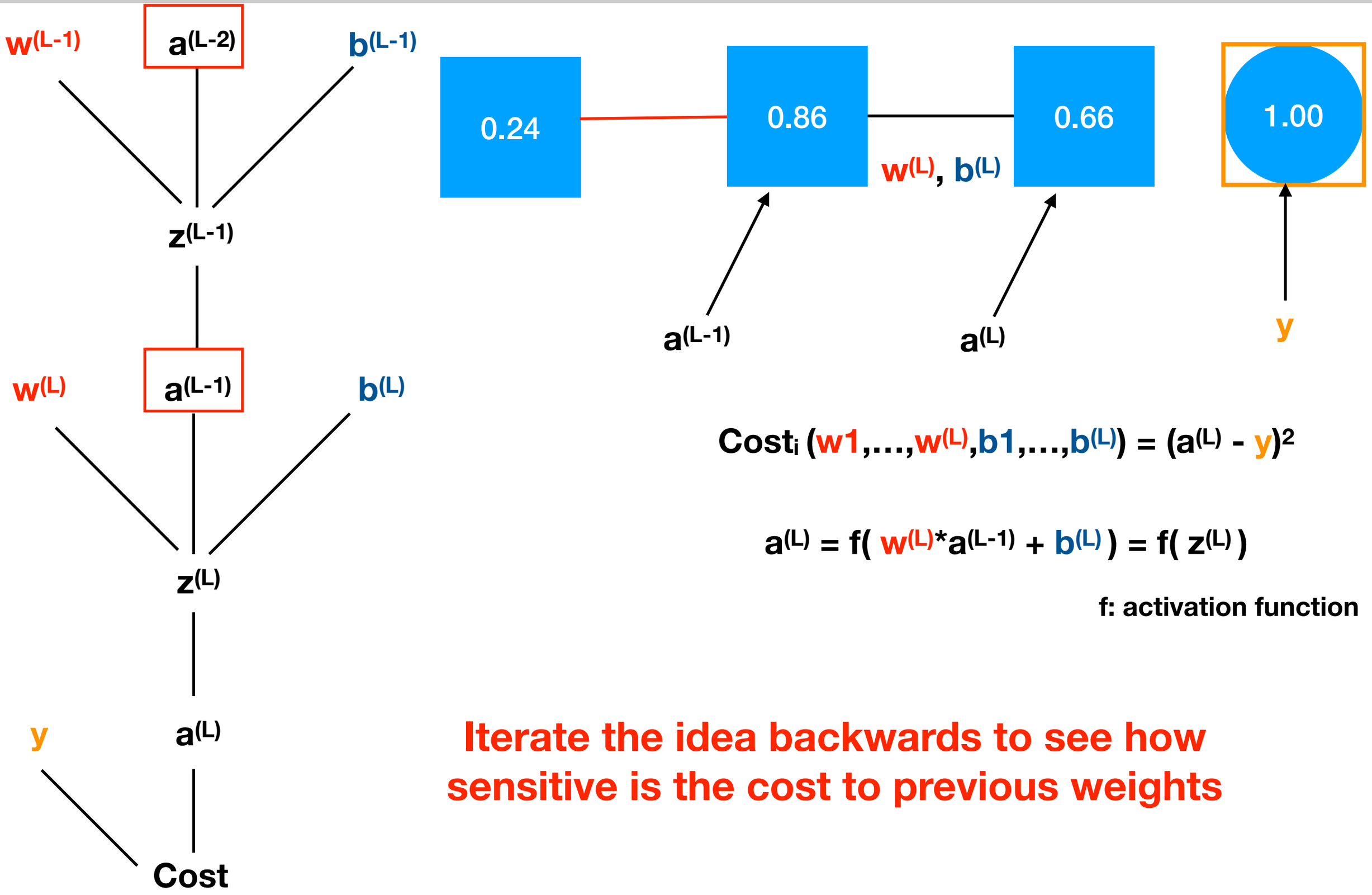
f: activation function

Same use of chain rule for $b^{(L)}$, and $a^{(L-1)}$: backpropagation

$$\frac{\partial Cost_i}{\partial b^{(L)}} = 2(a^{(L)} - y) f'(z^{(L)})$$

$$\frac{\partial Cost_i}{\partial a^{(L-1)}} = 2(a^{(L)} - y) f'(z^{(L)}) w^{(L)}$$

Network where each layer has a single neuron in it



$$Cost_i\,(w1,\ldots,w^{(L)},b1,\ldots,b^{(L)}) = (a^{(L)} - y)^2$$

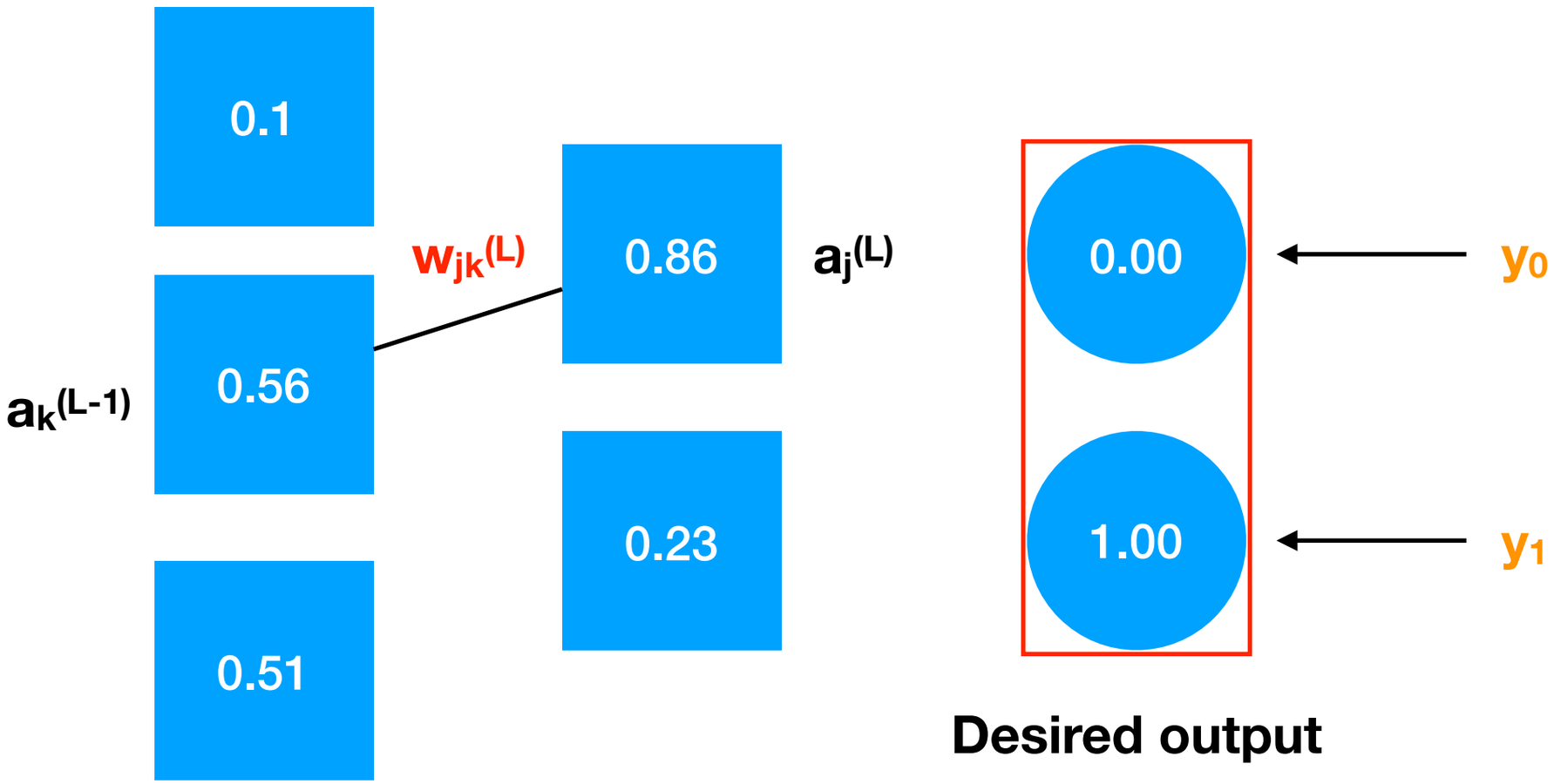
$$a^{(L)} = f(\, w^{(L)} * a^{(L-1)} + b^{(L)}\,) = f(\, z^{(L)}\,)$$

f: activation function

Iterate the idea backwards to see how sensitive is the cost to previous weights

Network where each layer has more than neuron in it

$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

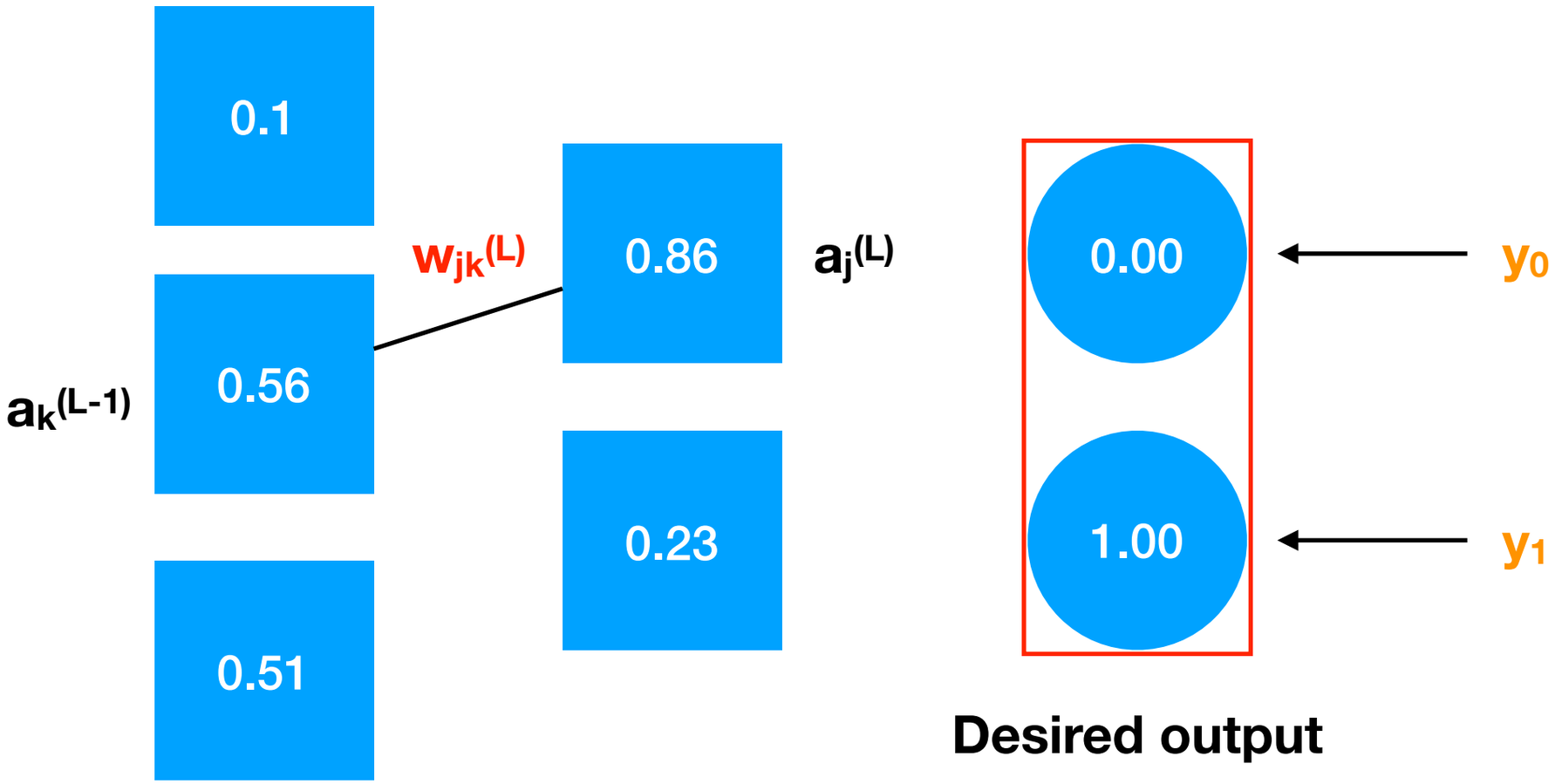


Network where each layer has more than neuron in it

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + w_{j2}^{(L)} a_2^{(L-1)} + b_j^{(L)}$$

$$a_j^{(L)} = f(z_j^{(L)})$$

$$Cost_i = \sum_{i=0}^{n_L - 1} (a_j^{(L)} - y_j)^2$$



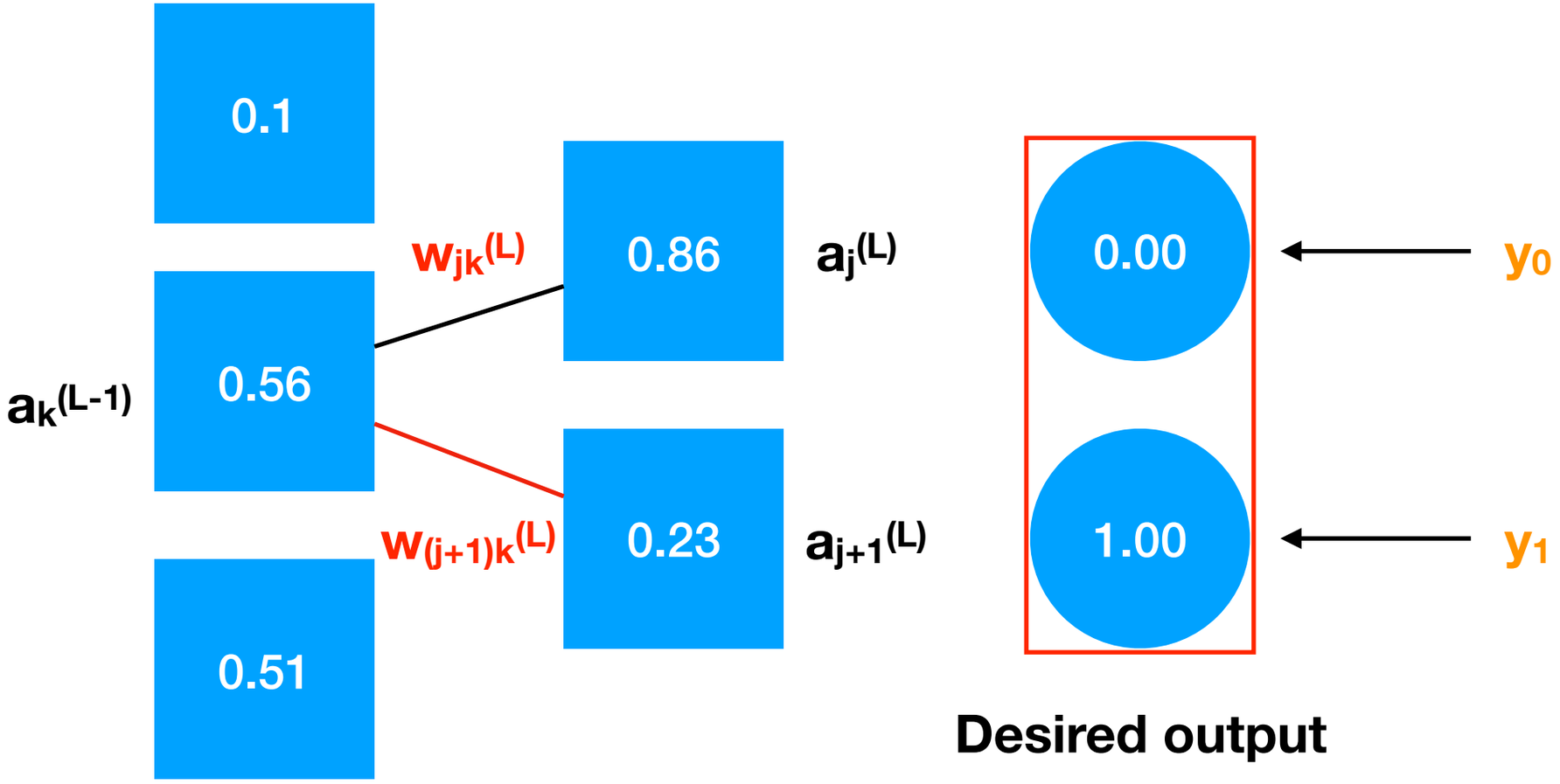
Network where each layer has more than neuron in it

$$\frac{\partial Cost_i}{\partial w_{jk}^{(L)}} = \frac{\partial Cost_i}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}}$$

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + w_{j2}^{(L)} a_2^{(L-1)} + b_j^{(L)}$$

$$\frac{\partial Cost_i}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial Cost_i}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \longrightarrow \textbf{Sum over layer L}$$

$$a_j^{(L)} = f(z_j^{(L)})$$



$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

Network where each layer has more than neuron in it

$$\frac{\partial Cost_i}{\partial w_{jk}^{(L)}} = \frac{\partial Cost_i}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}}$$

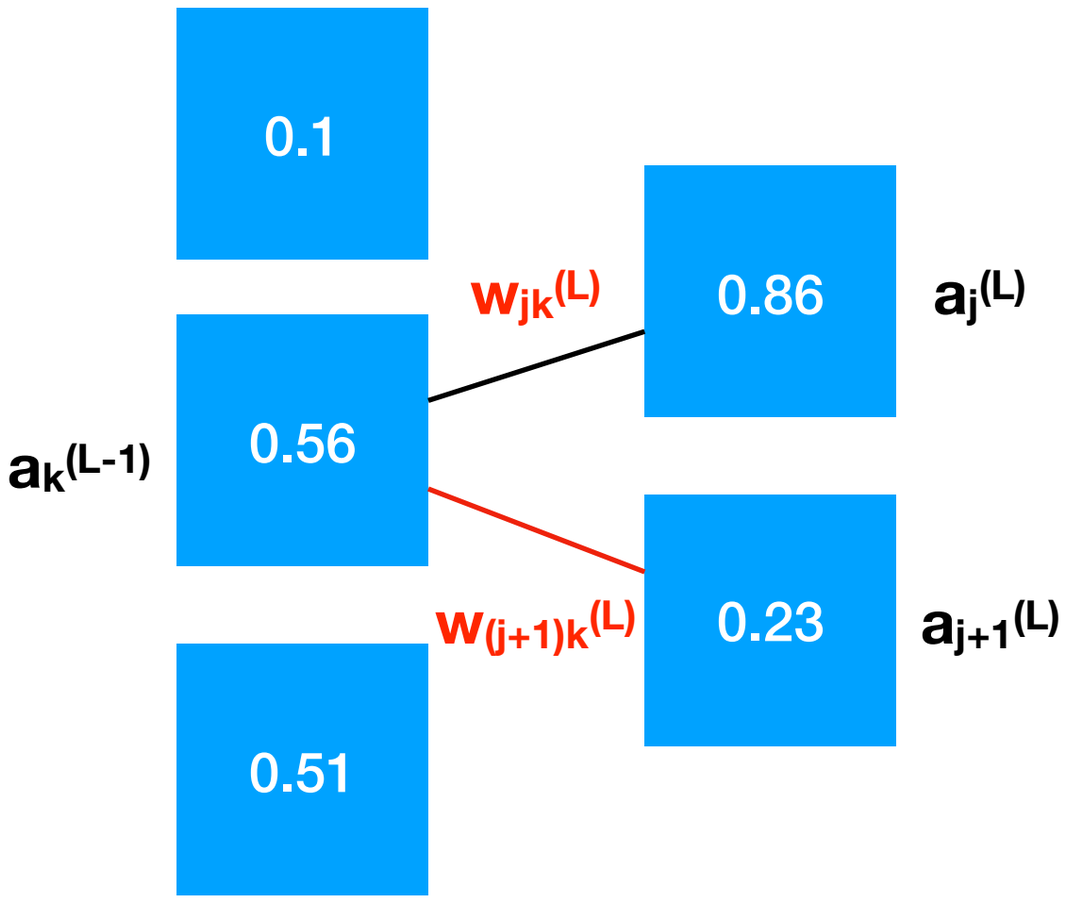
$$\frac{\partial Cost_i}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial Cost_i}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}}$$

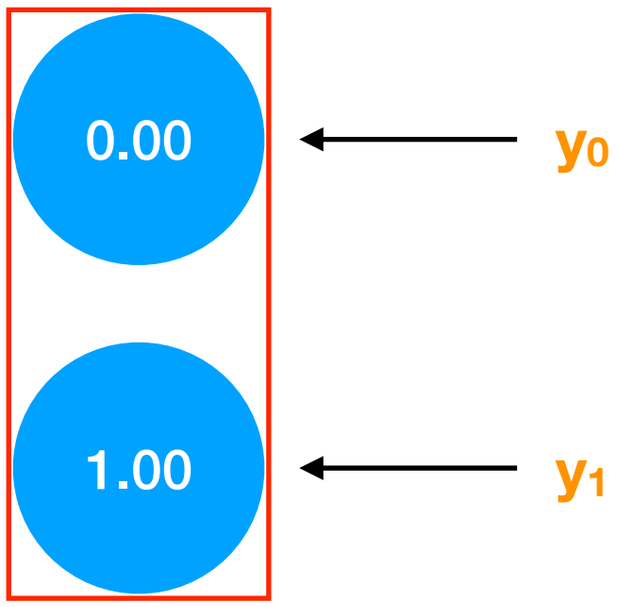
→ **Sum over layer L**

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + w_{j2}^{(L)} a_2^{(L-1)} + b_j^{(L)}$$

$$a_j^{(L)} = f(z_j^{(L)})$$

You can now compute each component of the gradient !





Desired output

$$Cost_i = \sum_{i=0}^{n_L-1} (a_j^{(L)} - y_j)^2$$

And how do I use this to update my bias and weights ?

Easy: use the learning rate to update :

$$w_{jk}^{(L)+} = w_{jk}^{(L)} - l_r * \frac{\partial Cost_i}{\partial w_{jk}^{(L)}}$$

$$b_{jk}^{(L)+} = b_{jk}^{(L)} - l_r * \frac{\partial Cost_i}{\partial b_{jk}^{(L)}}$$

Sources :

- https://www.youtube.com/watch?v=tleHLnjs5U8&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=4
- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/