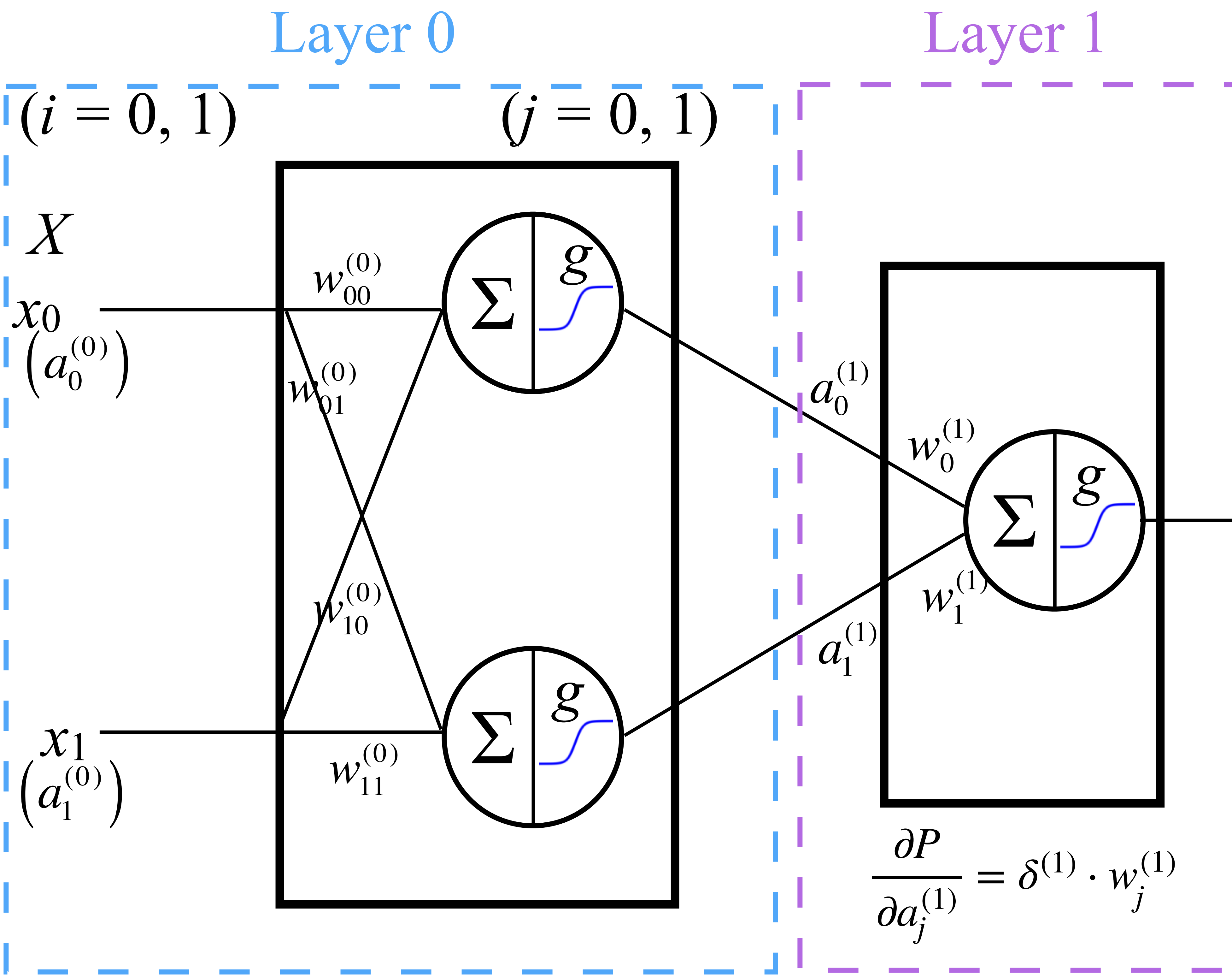
Forward-Feed, Multi-layer Artificial Neural Network — Part II



Note: eqn's (1) and (2) have the same structure:

$$(\text{delta} \otimes a) \cdot \alpha$$

Also note how the δ's are related.

Eqn (2) is more representative: generally Δw is a $m \times n$ matrix, the outer product of two vectors, δ (n-dim) and the input, a (m-dim).

$$\text{delta0}_j = \delta_j^{(0)} = \delta^{(1)} w_j^{(1)} \cdot g'(a_j^{(1)})$$

$$\Delta w_{ij}^{(0)} = \delta_j^{(0)} a_i^{(0)} \alpha \qquad (2)$$

input for layer 0

output for layer 0

$$\text{delta1} = \delta^{(1)} = (y - z) \cdot z(1 - z) = (y - z) \cdot g'(a_0^{(2)})$$

$$\Delta w_j^{(1)} = \delta^{(1)} \, a_j^{(1)} \alpha \qquad (1')$$

input for layer 1

output for layer 1

More Than One Output

In the forward direction:

One output:

$$z = g\left(\sum_j w_j^{(1)} \overbrace{g\left(\sum_i w_{i,j}^{(0)} \underbrace{x_i}_{a_i^{(0)}}\right)}^{j \text{ inputs (coming from the } j \text{ neurons) for the next layer — } a_j^{(1)}}\right)$$

Multiple outputs:

$$z_k = g\left(\sum_j w_{j,k}^{(1)} g\left(\sum_i w_{i,j}^{(0)} x_i\right)\right)$$

$$P = -\frac{1}{2}\sum_k (y_k - z_k)^2$$

More Than One Output

Layer 1

In the backward direction:

One final output, z:

$$\frac{dP}{dz} = y - z = \text{error} \qquad \delta^{(1)} = (y-z)\cdot g'(a_0^{(2)}) \qquad \Delta w_j^{(1)} = \delta^{(1)}\ a_j^{(1)}\alpha \qquad (1')$$

Multiple outputs, z_k

$$\frac{\partial P}{\partial z_k} = y_k - z_k \qquad \delta_k^{(1)} = (y_k - z_k)\cdot g'(z_k) \qquad \Delta w_{jk}^{(1)} = \delta_k^{(1)}\ a_j^{(1)}\alpha \qquad (1)$$

Layer 0

$$\delta^{(1)}\cdot w_j^{(1)}$$

One final output:

$$\delta_j^{(0)} = \delta^{(1)}\cdot w_j^{(1)} g'(a_j^{(1)}) \ = g'(a_j^{(1)})\frac{\partial P}{\partial a_j^{(1)}}$$

Multiple outputs:

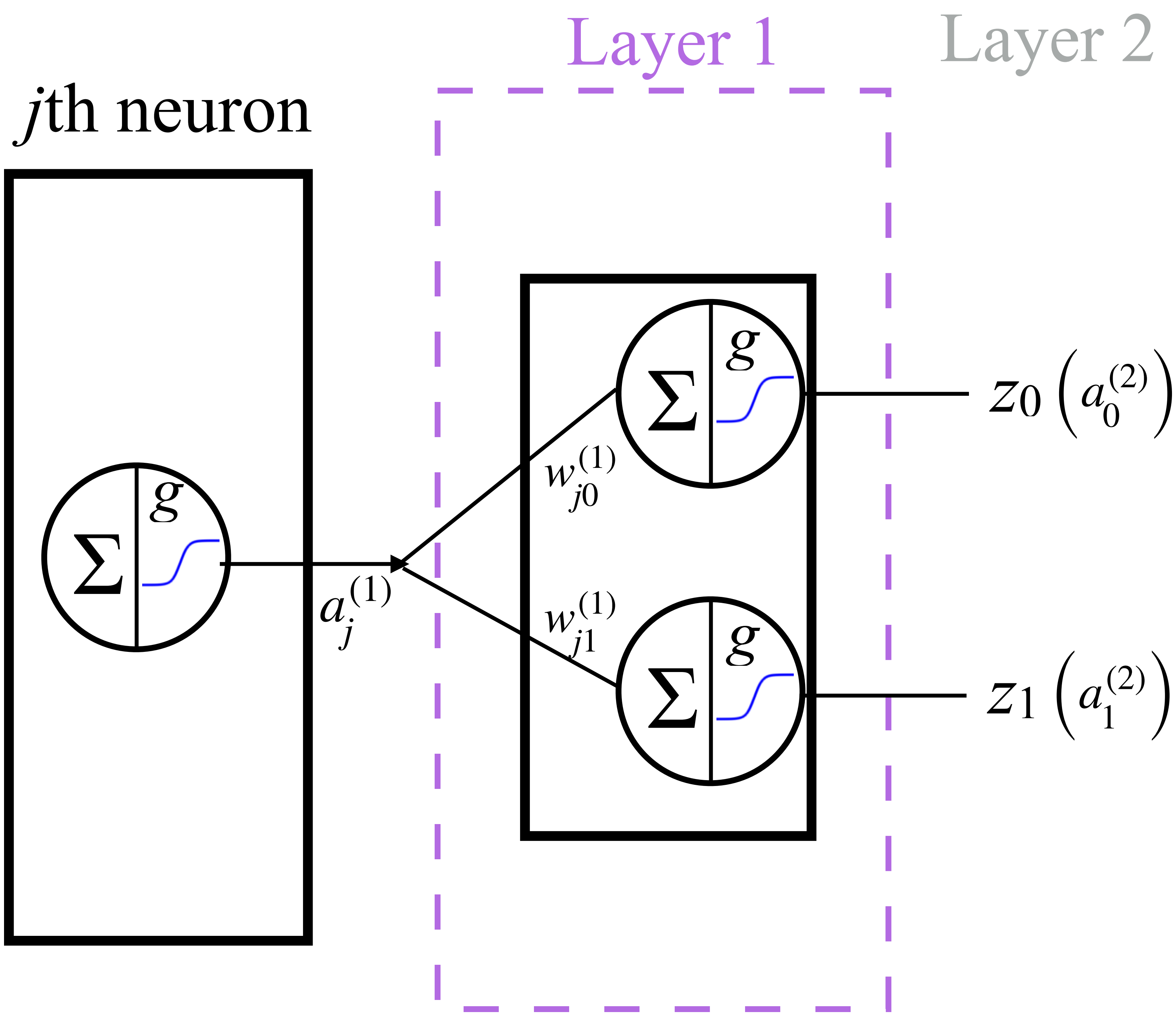
$$\Delta w_{ij}^{(0)} = \delta_j^{(0)} a_i^{(0)}\alpha \qquad (2)$$

$$\delta_j^{(0)} = g'(a_j^{(1)})\sum_k \delta_k^{(1)} w_{jk}^{(1)}$$

k: dot product (contract on k);

j: element-by-element multiplication (no contraction)

Key Equation for Multiple Outputs in Backprop



$$P = -\frac{1}{2}[(y_0 - z_0)^2 + (y_1 - z_1)^2]$$

$$= -\frac{1}{2}[(y_0 - g(w_{j0}^{(1)}a_j^{(1)}))^2$$

$$+(y_1 - g(w_{j1}^{(1)}a_j^{(1)}))^2]$$

$$\frac{\partial P}{\partial a_j^{(1)}} = (y_0 - z_0)g'(z_0)w_{j0}^{(1)}$$

$$+(y_1 - z_1)g'(z_1)w_{j1}^{(1)}$$

$$\delta_0^{(1)} = (y_0 - z_0)g'(z_0)$$

$$\delta_1^{(1)} = (y_1 - z_1)g'(z_1)$$

$$\boxed{\frac{\partial P}{\partial a_j^{(1)}} = \delta_0^{(1)}w_{j0}^{(1)} + \delta_1^{(1)}w_{j1}^{(1)} = \sum_k \delta_k^{(1)}w_{jk}^{(1)}} \longrightarrow \delta_j^{(0)} = g'(a_j^{(1)})\sum_k \delta_k^{(1)}w_{jk}^{(1)}$$

Multi-layer Forward-Feed ANN Backpropagation

Suppose there are L+1 layers: The inputs, x_b's count as layer 0, and outputs, z_k's count as layer L. Hidden layers are l = 1 to L–1.

In our simple example, L = 2; thus only one hidden layer, l = 1.

$$\frac{\partial P}{\partial z_k} = y_k - z_k$$

Output to the Lth layer

Input to the Lth layer

$$\delta_k^{(L-1)} = (y_k - z_k)g'(z_k) = (y_k - z_k)g'(a_k^{(L)}) \quad (3)$$

$$\Delta w_{jk}^{(L-1)} = \alpha\delta_k^{(L-1)}\, a_j^{(L-1)} \quad (1)$$

$$\delta_j^{(L-2)} = g'(a_j^{(L-1)})\sum_k \delta_k^{(L-1)} w_{jk}^{(L-1)}$$

$$\Delta w_{ij}^{(L-2)} = \alpha\delta_j^{(L-2)} a_i^{(L-2)} \quad (2)$$

$$\delta_i^{(L-3)} = g'(a_i^{(L-2)})\sum_j \delta_j^{(L-2)} w_{ij}^{(L-2)}$$

$$\Delta w_{hi}^{(L-3)} = \alpha\delta_i^{(L-3)} a_h^{(L-3)}$$

$\vdots$

$$\boxed{\delta_s^{(l-1)} = g'(a_s^{(l)})\sum_t \delta_t^{(l)} w_{st}^{(l)} \quad (4) \qquad \Delta w_{rs}^{(l-1)} = \alpha\delta_s^{(l-1)} a_r^{(l-1)} \quad (5)}$$

$\vdots$

$$\delta_d^{(1)} = g'(a_d^{(2)})\sum_e \delta_e^{(2)} w_{de}^{(2)}$$

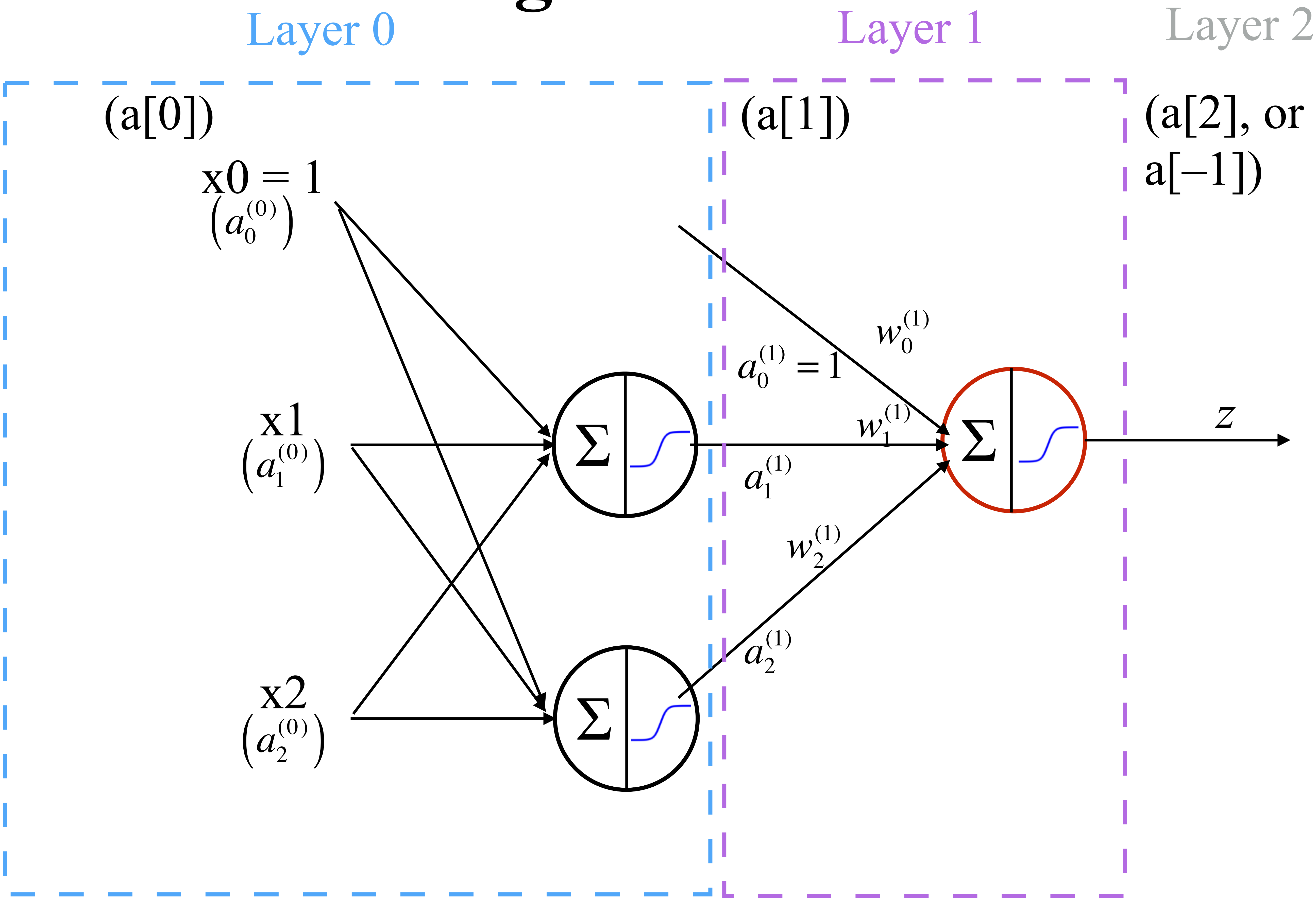
$$\Delta w_{cd}^{(1)} = \alpha\delta_d^{(1)} a_c^{(1)}$$

$$\delta_c^{(0)} = g'(a_c^{(1)})\sum_d \delta_d^{(1)} w_{cd}^{(1)}$$

$$\Delta w_{bc}^{(0)} = \alpha\delta_c^{(0)} x_b \;= \alpha\delta_c^{(0)} a_b^{(0)}$$

Computationally, x is a[0] and z is a[–1].

Solving The XOR



What determines a NN

- The activation function
- Forward feed and fully connected, or something more complicated (e.g., recurrent or convolutional)
- The number of hidden layers
- The number of neurons in each hidden layer

The number of weights is actually set once the above is set. If one of the connections (called a synapse) turns out to be not important, it will be assigned a low weight in the training process.

The number of weights shouldn’t be greater than the number of input possibilities.

End of Week 8-1