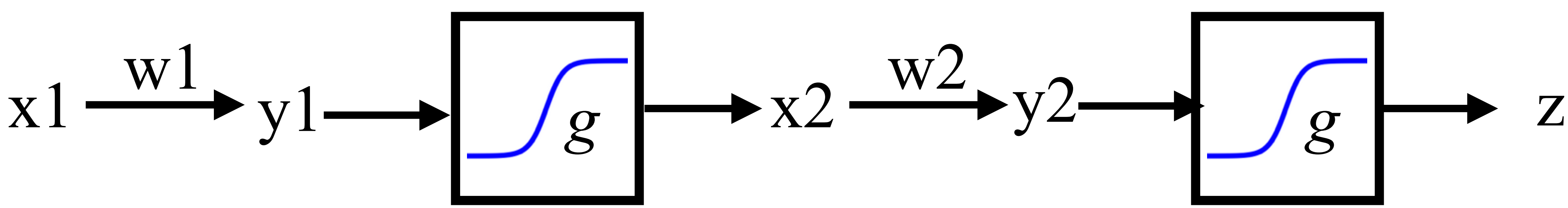
Forward-Feed, Multi-layer Artificial Neural Network

From two classes ago…

Forward-feed, Two-layer, One-input-one-output ANN

Graphic Representation:

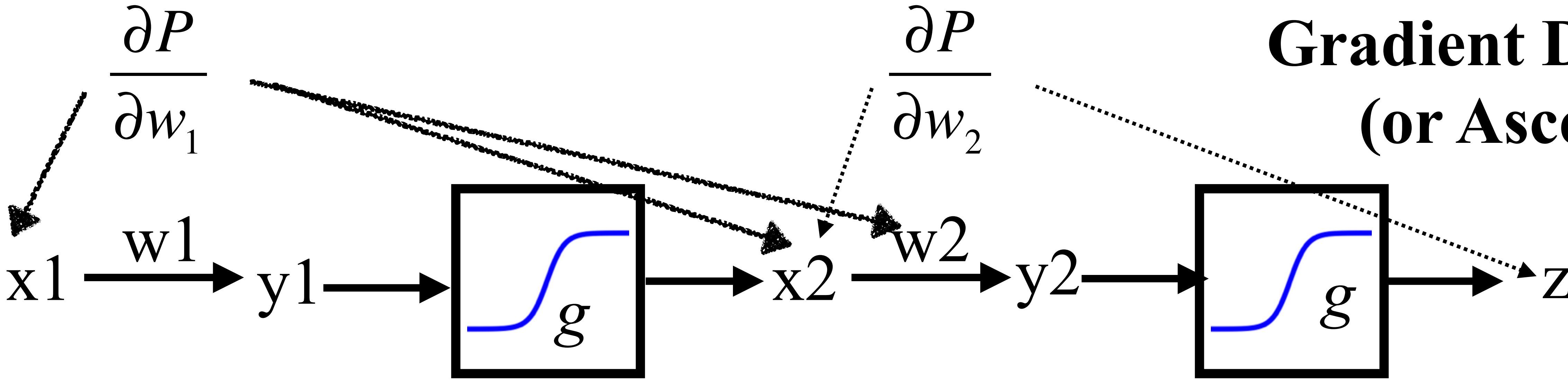


g = "activation function", "thresholding"

Mathematical statement:

$$z = g\left(w_2 g\left(w_1 x_1\right)\right)$$

Gradient Descent (or Ascent)



What we need to figure out:

$$\frac{\partial P}{\partial w_2} = \frac{dP}{dz}\frac{dz}{dy_2}\frac{dy_2}{dw_2}$$

$$\frac{\partial P}{\partial w_1} = \frac{dP}{dz}\frac{dz}{dw_1} = \frac{dP}{dz}\frac{dz}{dy_2}\frac{dy_2}{dx_2}\frac{dx_2}{dy_1}\frac{dy_1}{dw_1}$$

What we know:

$$\frac{dP}{dz} = d - z$$

$$\frac{dz}{dy_2} == z(1-z) \qquad \frac{dx_2}{dy_1} == x_2(1-x_2)$$

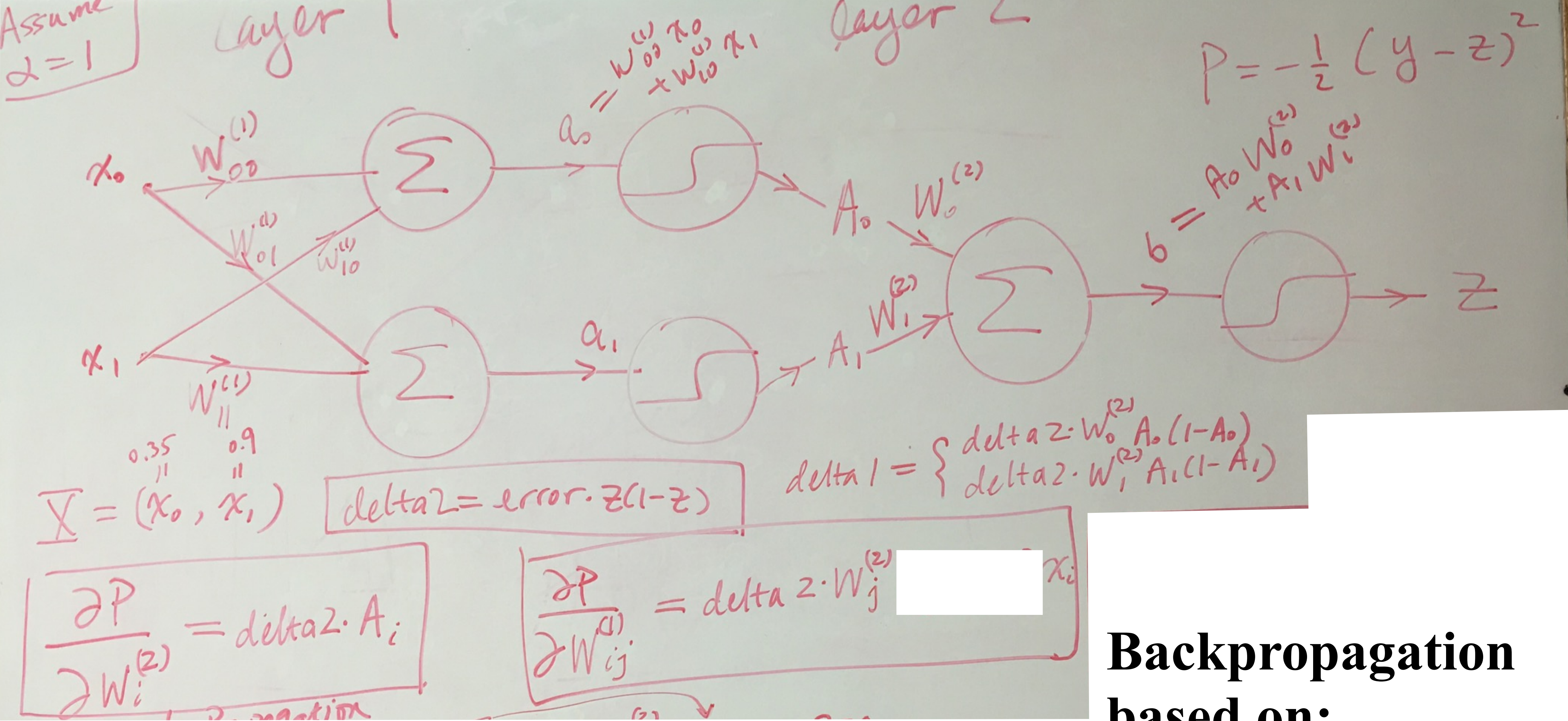
$$\frac{dy_2}{dw_2} = x_2 \qquad \frac{dy_1}{dw_1} = x_1$$

$$\frac{\partial P}{\partial w_2} = (d-z)z(1-z)x_2 = (d-z)g'(z)x_2$$

$$\frac{\partial P}{\partial w_1} = (d-z)g'(z)w_2 g'(x_2)x_1$$

g: activation function, or the threshold function

Summary:
For $\partial P/\partial w_i$: In addition to what you have already calculated, it only depends on x_i, x_{i+1}, and w_{i+1} (or x_i and z).



Backpropagation based on:

$$\frac{\partial P}{\partial w_2} = \frac{dP}{dz}\frac{dz}{dy_2}\frac{dy_2}{dw_2}$$

$$\frac{\partial P}{\partial w_1} = \frac{dP}{dz}\frac{dz}{dw_1}$$

$$= \frac{dP}{dz}\frac{dz}{dy_2}\frac{dy_2}{dx_2}\frac{dx_2}{dy_1}\frac{dy_1}{dw_1}$$

Forward-feed, two-layer, two-input-one-output ANN:

$$z = g\left(\sum_h w_h^{(2)} g\left(\sum_j w_{j,h}^{(1)} x_j\right)\right)$$

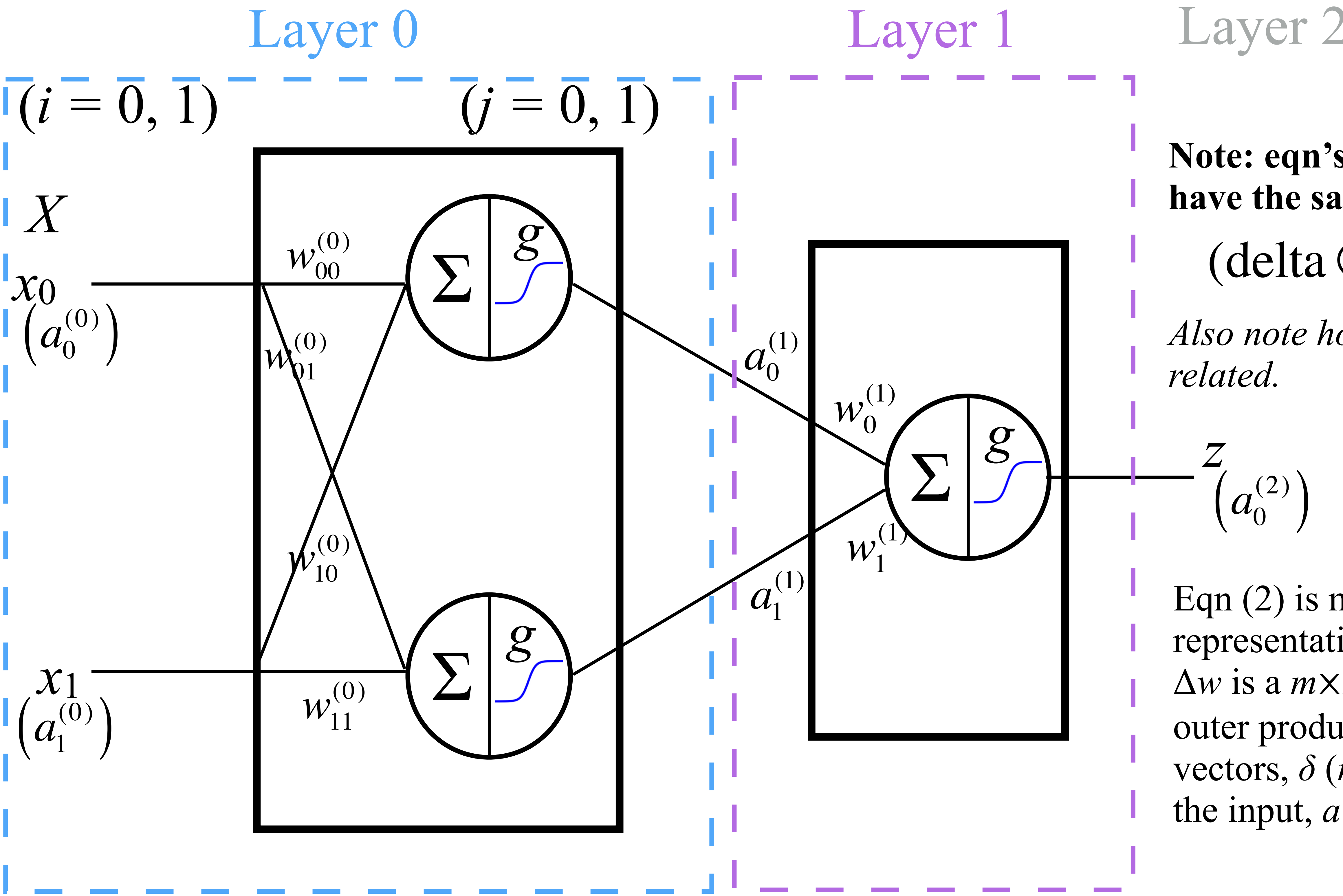
Changing Notations

Data: X

Target: $d \rightarrow y$

x_0	x_1	y
0	0	1
0	1	0
1	0	0
1	1	0

$$\text{error} = (d - z) \rightarrow (y - z)$$



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:

$$z = g\left(\sum_h w_h g\left(\sum_j w_{j,h} x_j\right)\right) \rightarrow$$

$$z_i = g\left(\sum_h w_{h,i} g\left(\sum_j w_{j,h} x_j\right)\right)$$

More Than One Output

Backward propagation:

For one final output, z,

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

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

For one final output,

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

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

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

$f(u_1,u_2)$ with $u_1(v_1,v_2)$ and $u_2(v_1,v_2)$

$$\rightarrow \frac{\partial f}{\partial v_1} = \frac{\partial f}{\partial u_1}\frac{\partial u_1}{\partial v_1} + \frac{\partial f}{\partial u_2}\frac{\partial u_2}{\partial v_1}$$

With respect to k: dot product (contract on k);

With respect to j: element-by-element multiplication (no contraction)

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, $l = 1$.

Output to the Lth layer

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

Input to the Lth layer

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

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

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

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

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

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

$\vdots$

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

$\vdots$

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

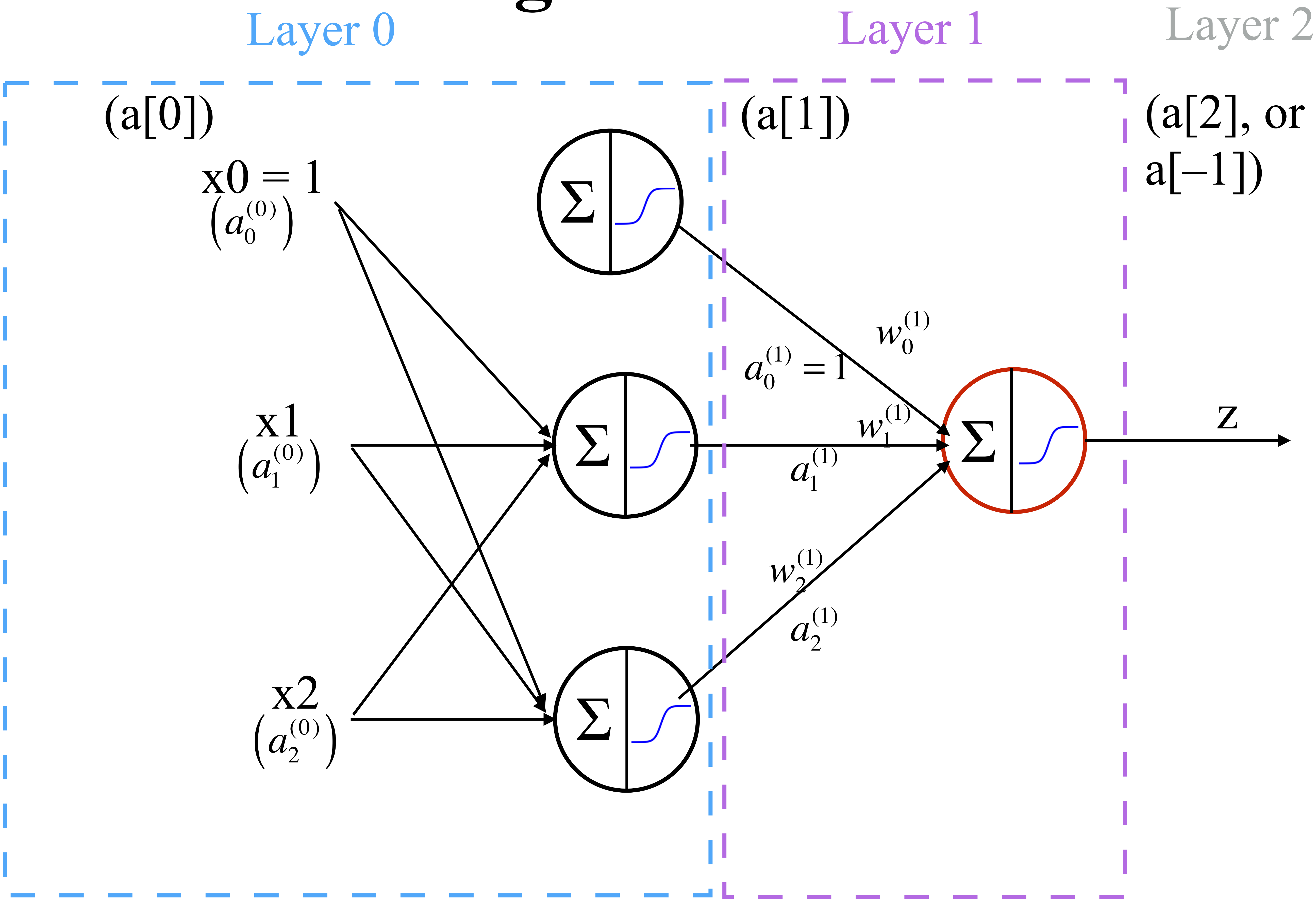
$$\Delta w_{cd}^{(1)} = \alpha \delta_d^{(1)} a_c^{(1)}$$

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

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

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

Solving The XOR



Stochastic Gradient Descent (or Ascent)

x_0	x_1	y
0	0	1
0	1	0
1	0	0
1	1	0

End of Week 8-1