Happy Wednesday!

- **Assignment 4 due tonight Nov 11th, 11:59 pm (midnight)**
  - **Exceptional late policy:** No penalty until Mon, Nov 23rd, 11:59 pm
- **Quiz 11, Friday, Oct 30th 6am until Nov 1st 11:59pm (midnight)**
  - Neural networks

Coming up soon

- **Touch-point 3**: deliverables due **Nov 22nd**, live-event Mon, Nov 23rd
  - Single-slide presentation outlining progress highlights and current challenges
  - Three-minute pre-recorded presentation with your progress and current challenges
- **Project final report due Dec 7th 11:59pm (midnight)**
  - GitHub page with all of the results you have achieved utilizing both unsupervised learning and supervised learning
  - Final seven-minute long pre-recorded presentation

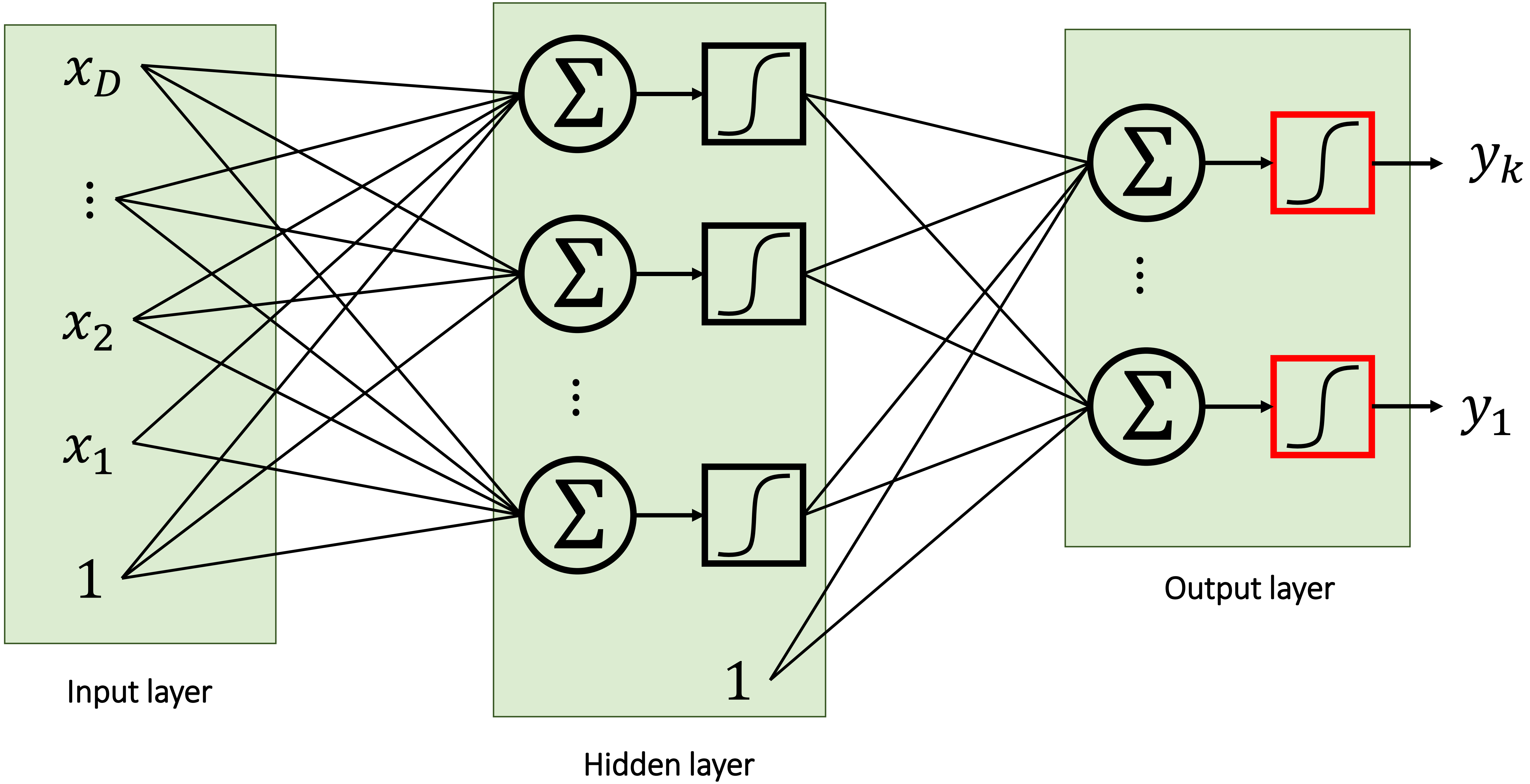
CS4641B Machine Learning

Lecture 22: Neural networks: Backpropagation algorithm

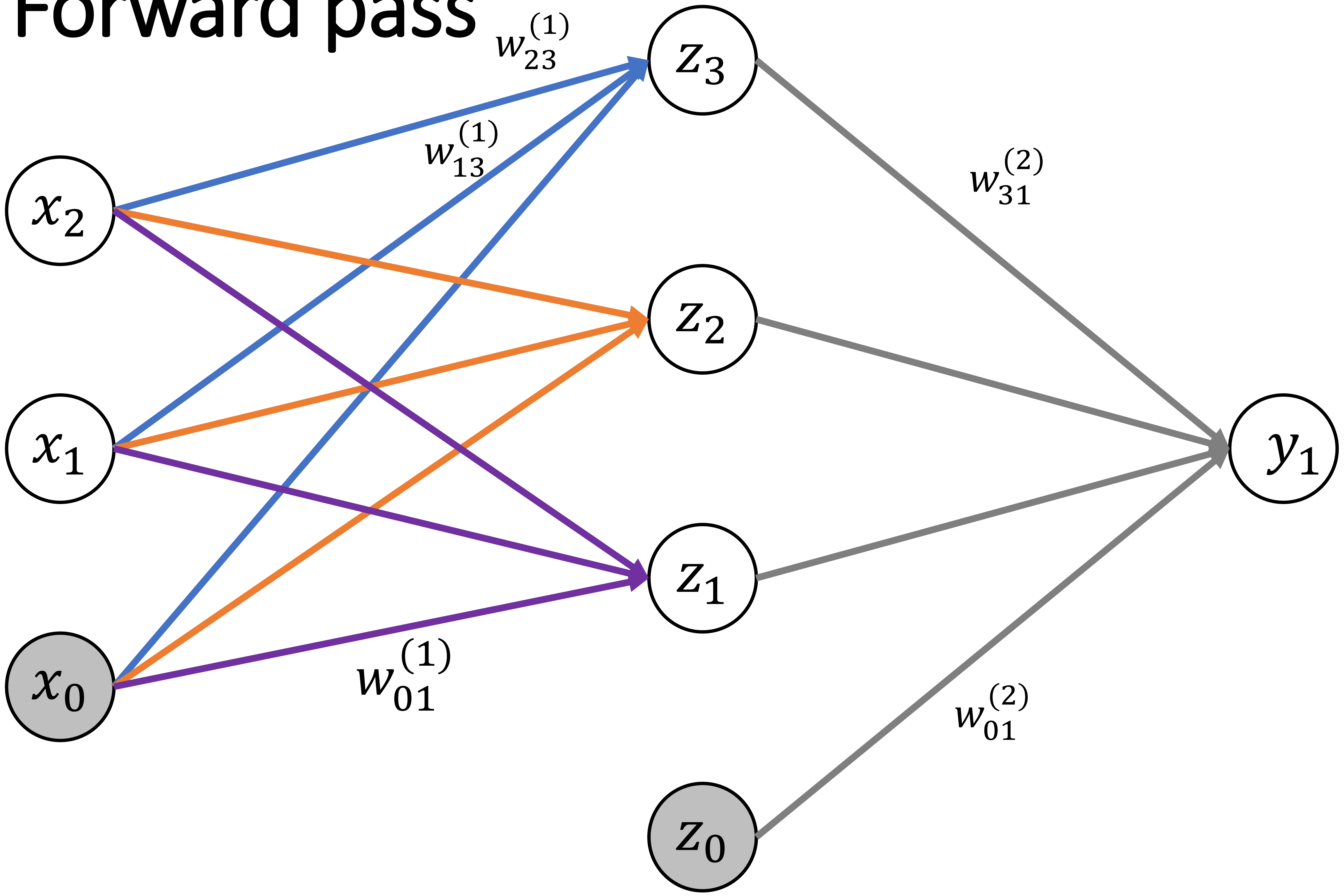
Rodrigo Borela ▸ rborelav@gatech.edu

Recap

- This is a **two layer** feed forward neural network



Recap: Forward pass



Recap: Forward pass

- Activations

$$a_1 = \left(\sum_{i=1}^{D} w_{i1}^{(1)} x_i \right) + w_{01}^{(1)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{01}^{(1)}$$

$$a_2 = \left(\sum_{i=1}^{D} w_{i2}^{(1)} x_i \right) + w_{02}^{(1)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{02}^{(1)}$$

$$a_3 = \left(\sum_{i=1}^{D} w_{i3}^{(1)} x_i \right) + w_{03}^{(1)} = w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 + w_{03}^{(1)}$$

- Hidden units

$$z_1 = h(a_1) = \frac{1}{1 + \exp(-a_1)}$$

$$z_2 = h(a_2) = \frac{1}{1 + \exp(-a_2)}$$

$$z_3 = h(a_3) = \frac{1}{1 + \exp(-a_3)}$$

Recap: Forward pass

- Output

$$a_1^{(2)} = \left(\sum_{i=1}^{M} w_{ij}^{(2)} z_i \right) + w_{01}^{(2)} = w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2 + w_{31}^{(2)} z_3 + w_{01}^{(2)}$$

$$y_1 = \sigma\left(a_1^{(2)}\right) = a_1^{(2)}$$

Training the network

Training the network

- Stochastic gradient descent

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n\big(\mathbf{w}^{(\tau)}\big)$$

- Considering a linear model $y_k(\mathbf{x}, \mathbf{w}_k) = \mathbf{w}_k^T \mathbf{x} = \sum_{i=0}^{D} w_{ik} x_i$

- Error function for a datapoint $\mathbf{x}_n$ with a target vector of size k:

$$E_n = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

- Calculating the gradient of this error function with respect to w_{ik}:

$$\frac{\partial E_n}{\partial w_{ik}} = \big(y_{nj} - t_{nj}\big) x_{ni}$$

Training the network

- Activation:

$$a_j = \sum_i w_{ij} z_i \text{ (z_i input from another layer)}$$

- Hidden unit

$$z_j = h(a_j)$$

- The gradient of the error E_n depends on the weight w_{ij} only via the summed input a_j to unit j

$$\frac{\partial E_n}{\partial w_{ij}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}}$$

Training the network

- Output layer

$$\delta_k = y_k - t_k$$

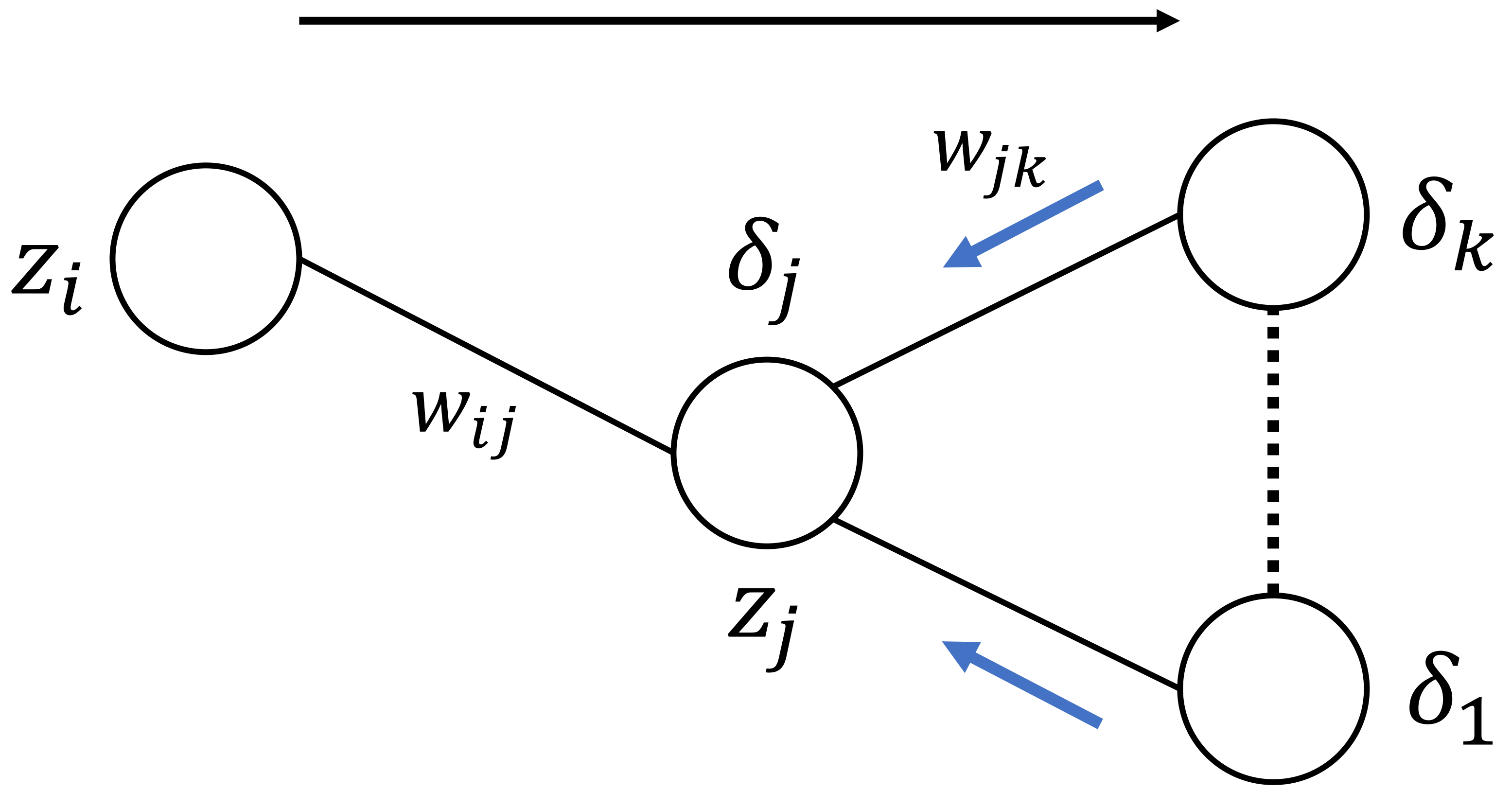
- Hidden layers

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

$$\delta_j = \frac{\partial E_n}{\partial a_j} = h'(a_j) \sum_k w_{jk} \delta_k$$

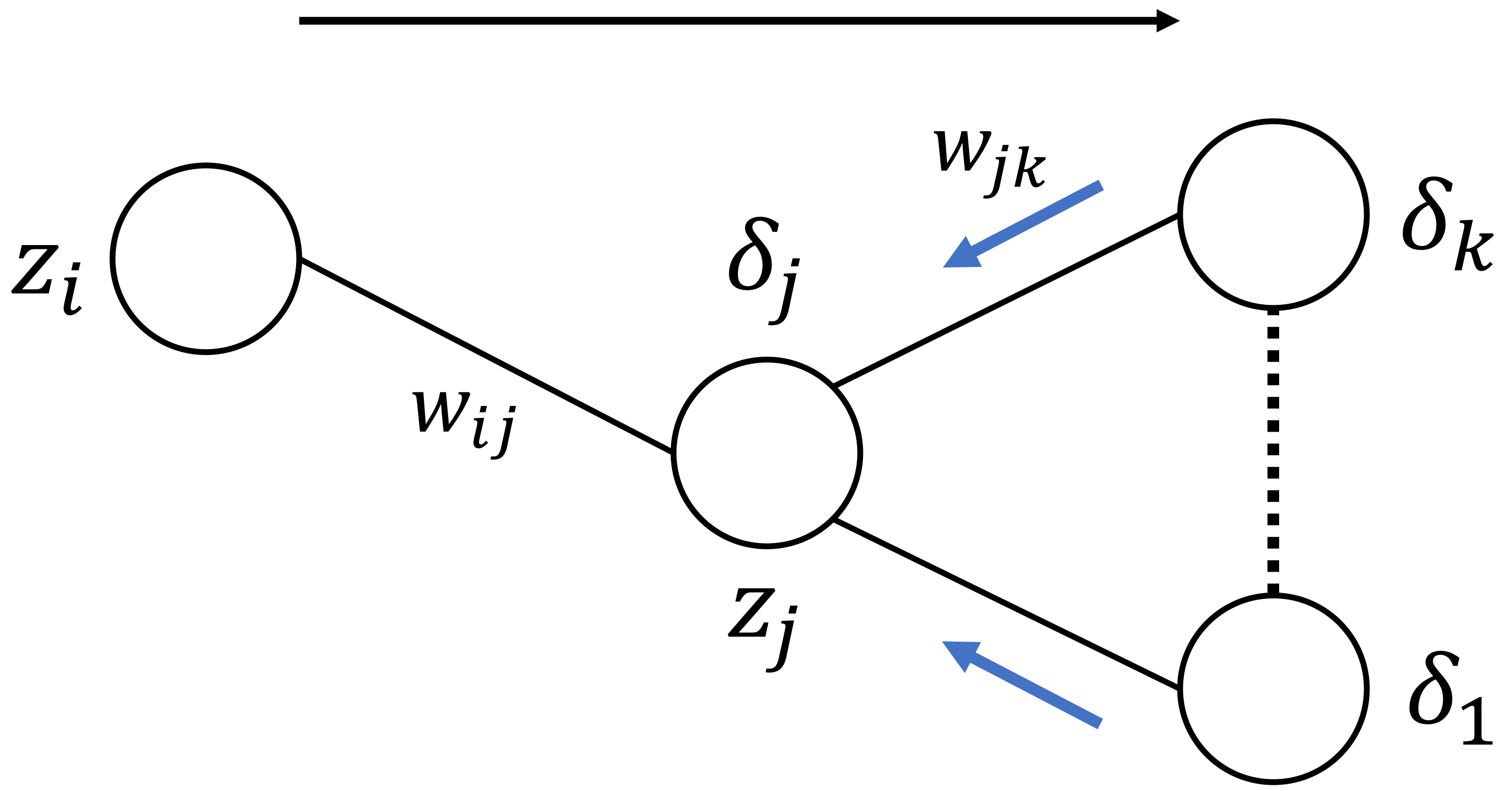
Error backpropagation algorithm

- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Error backpropagation algorithm

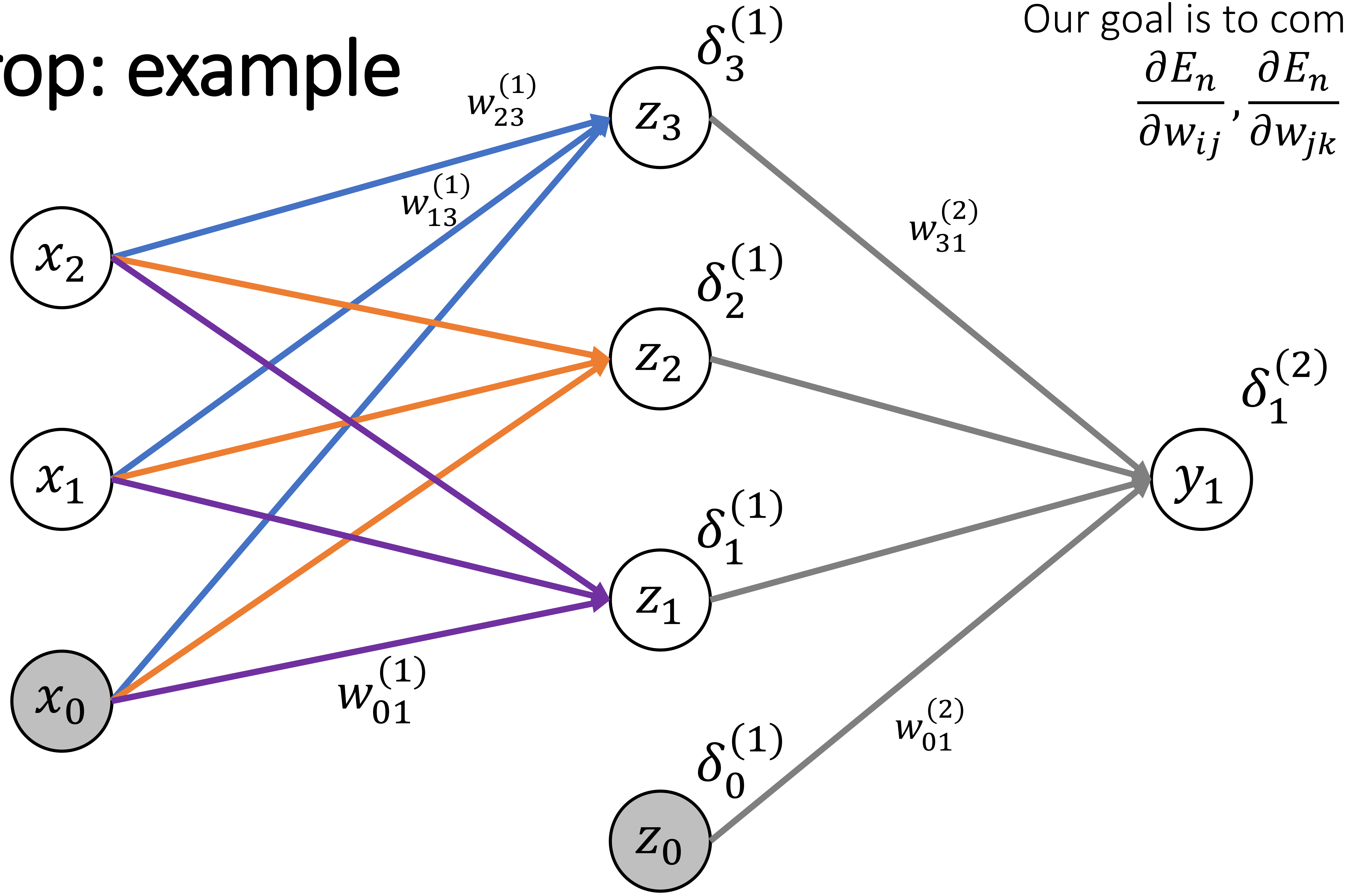
- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Backprop: example

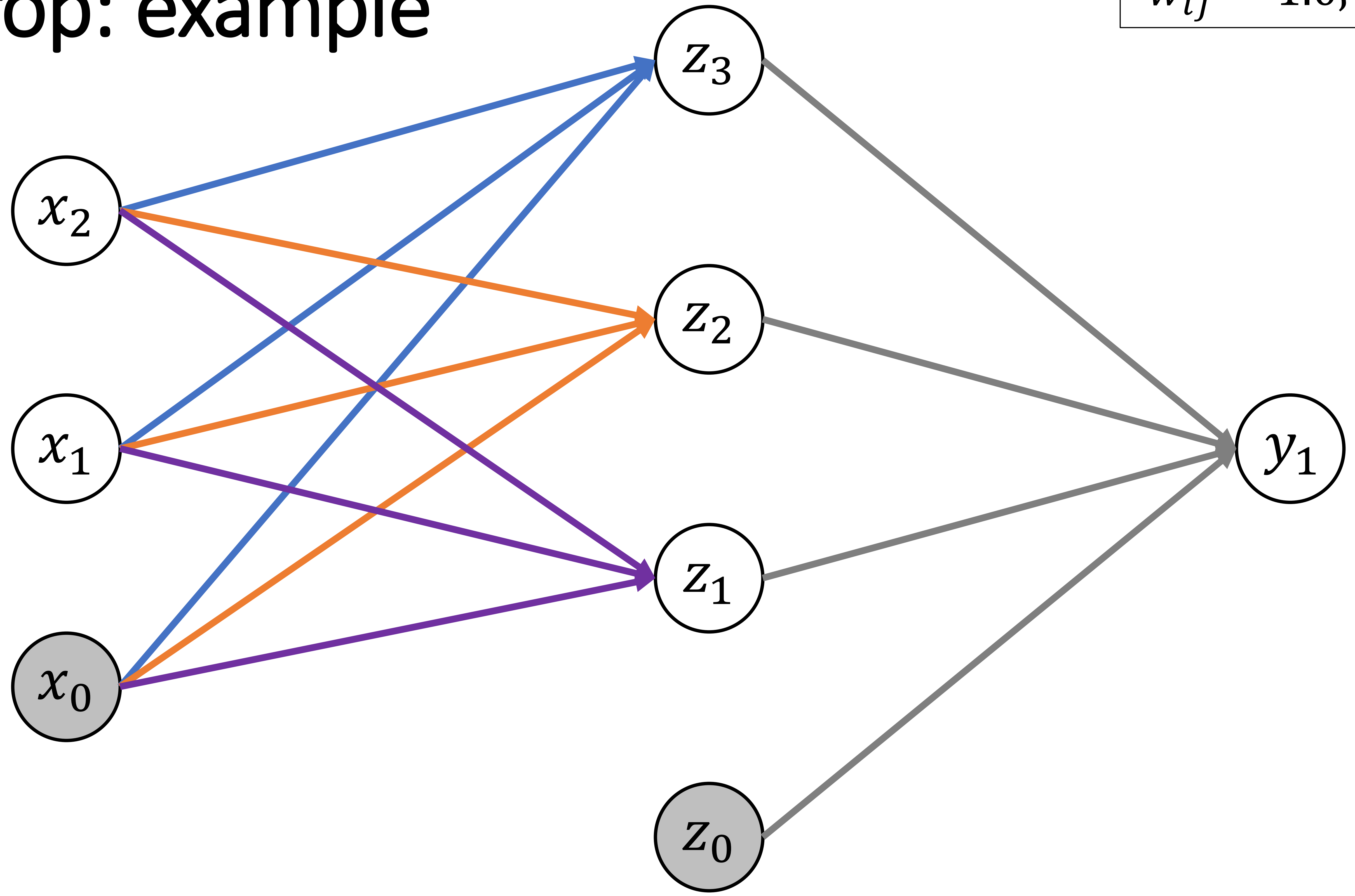
Our goal is to compute:

$$\frac{\partial E_n}{\partial w_{ij}}, \frac{\partial E_n}{\partial w_{jk}}$$



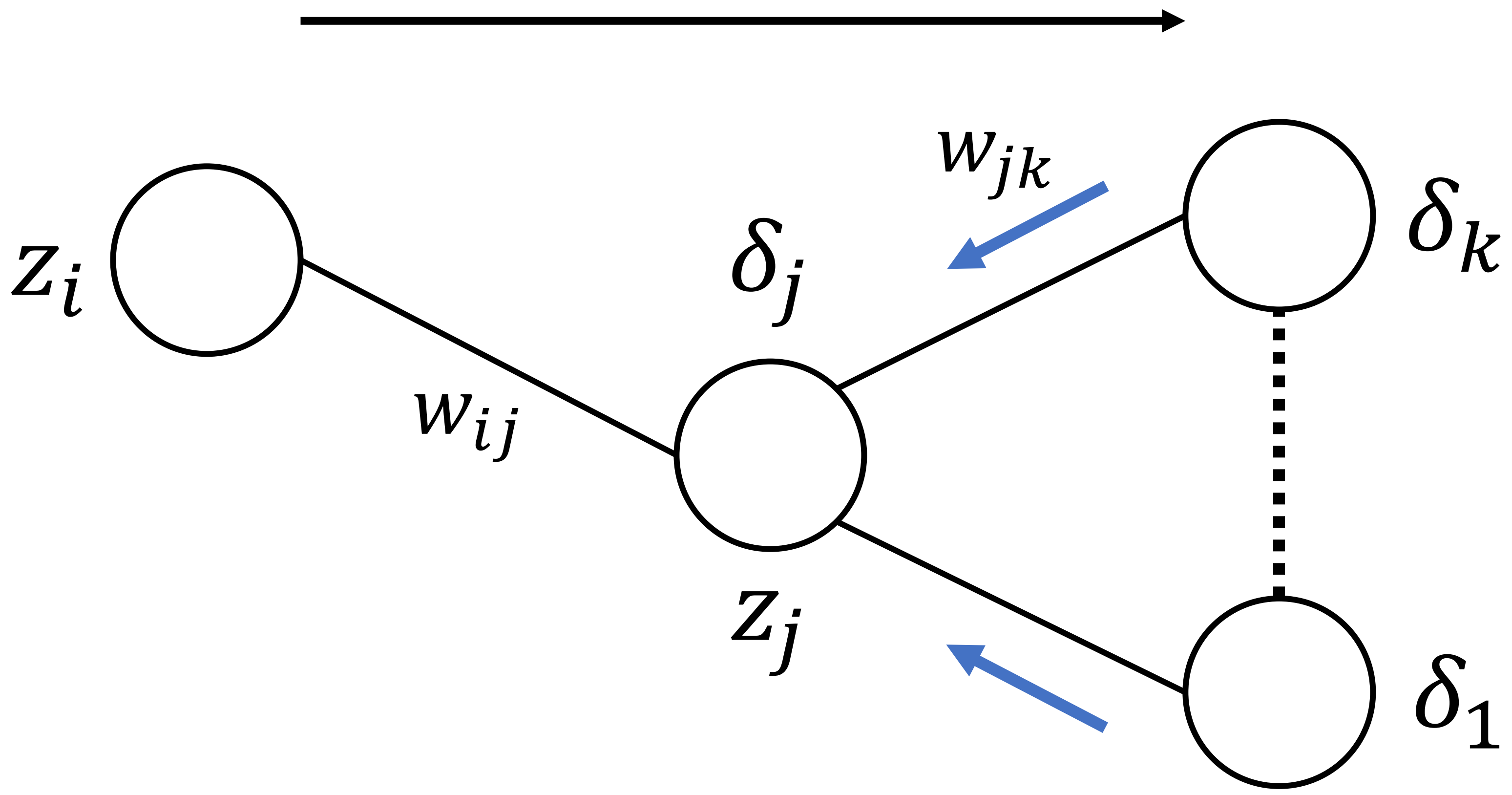
Backprop: example

$w_{ij} = 1.0, w_{jk} = 1$



Error backpropagation algorithm

- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Backprop: example

- Forward pass with input vector $\mathbf{x}_n = [0.5 \quad 0.5]^T$, $t_n = 0.5$

$$a_1^{(1)} = \left(\sum_{i=1}^{D} w_{i1}^{(1)} x_i\right) + w_{01}^{(1)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{01}^{(1)} = 1 \times 0.5 + 1 \times 0.5 + 1 = 2.0$$

$$a_2^{(1)} = \left(\sum_{i=1}^{D} w_{i2}^{(1)} x_i\right) + w_{02}^{(1)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{02}^{(1)} = 1 \times 0.5 + 1 \times 0.5 + 1 = 2.0$$

$$a_3^{(1)} = \left(\sum_{i=1}^{D} w_{i3}^{(1)} x_i\right) + w_{03}^{(1)} = w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 + w_{03}^{(1)} = 1 \times 0.5 + 1 \times 0.5 + 1 = 2.0$$

- Hidden units:

$$z_1 = h\left(a_1^{(1)}\right) = \frac{1}{1 + \exp\left(-a_1^{(1)}\right)} = \frac{1}{1 + \exp(-2)} = 0.88$$

$$z_2 = h\left(a_2^{(1)}\right) = \frac{1}{1 + \exp\left(-a_2^{(1)}\right)} = \frac{1}{1 + \exp(-2)} = 0.88$$

$$z_3 = h\left(a_3^{(1)}\right) = \frac{1}{1 + \exp\left(-a_3^{(1)}\right)} = \frac{1}{1 + \exp(-2)} = 0.88$$

Backprop: example

- Output

$$a_1^{(2)} = \left(\sum_{i=1}^{M} w_{ij}^{(2)} z_i \right) + w_{01}^{(2)} = w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2 + w_{31}^{(2)} z_3 + w_{01}^{(2)} = 3.64$$

$$y_1 = \sigma\left(a_1^{(2)}\right) = a_1^{(2)} = 3.64$$

Backprop: example

Input variable ($\mathbf{x}_n$)

- $x_1 = 0.5$
- $x_2 = 0.5$

Hidden units:

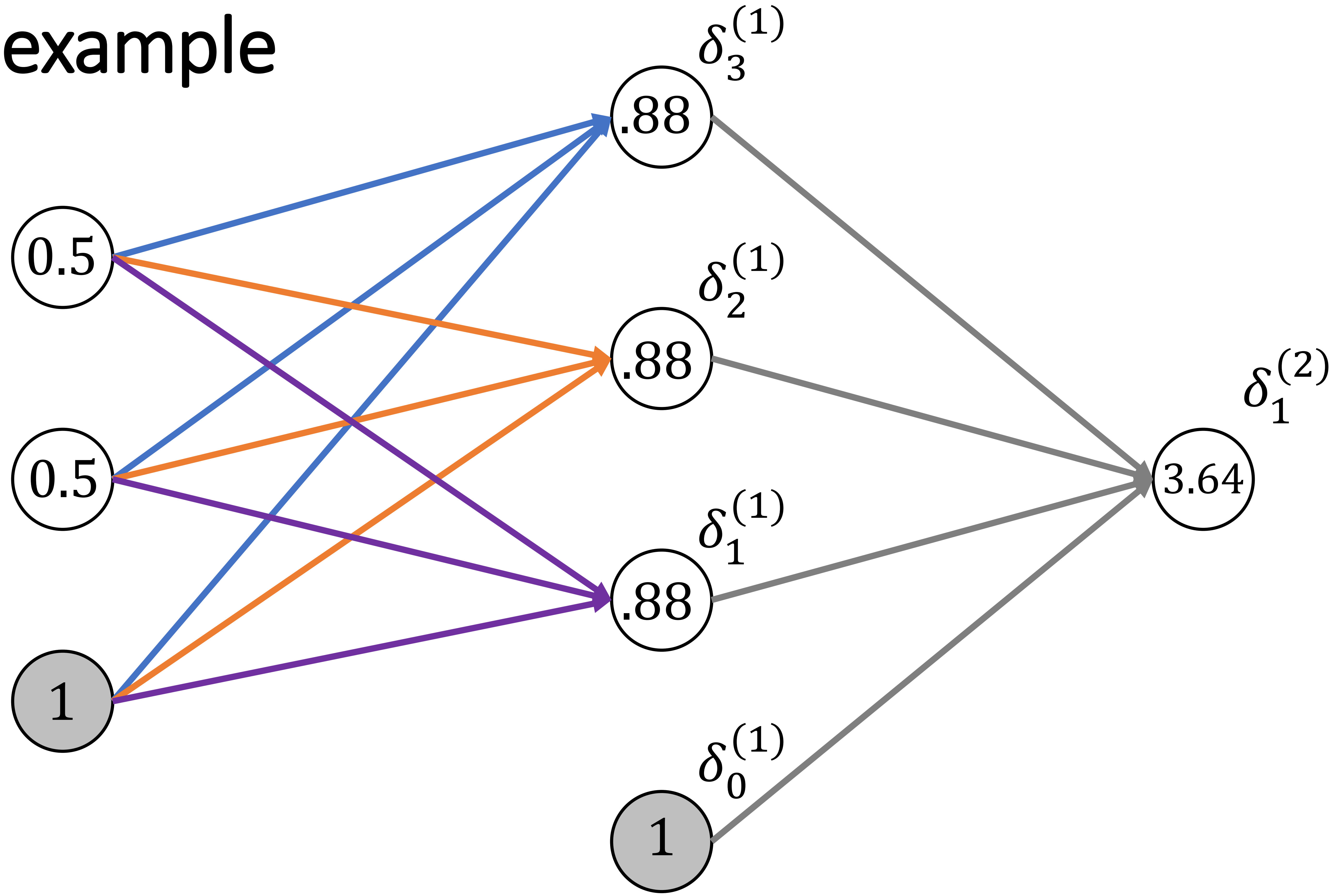
- $z_1 = 0.88$
- $z_2 = 0.88$
- $z_3 = 0.88$

Output

- $y_1 = 3.64$

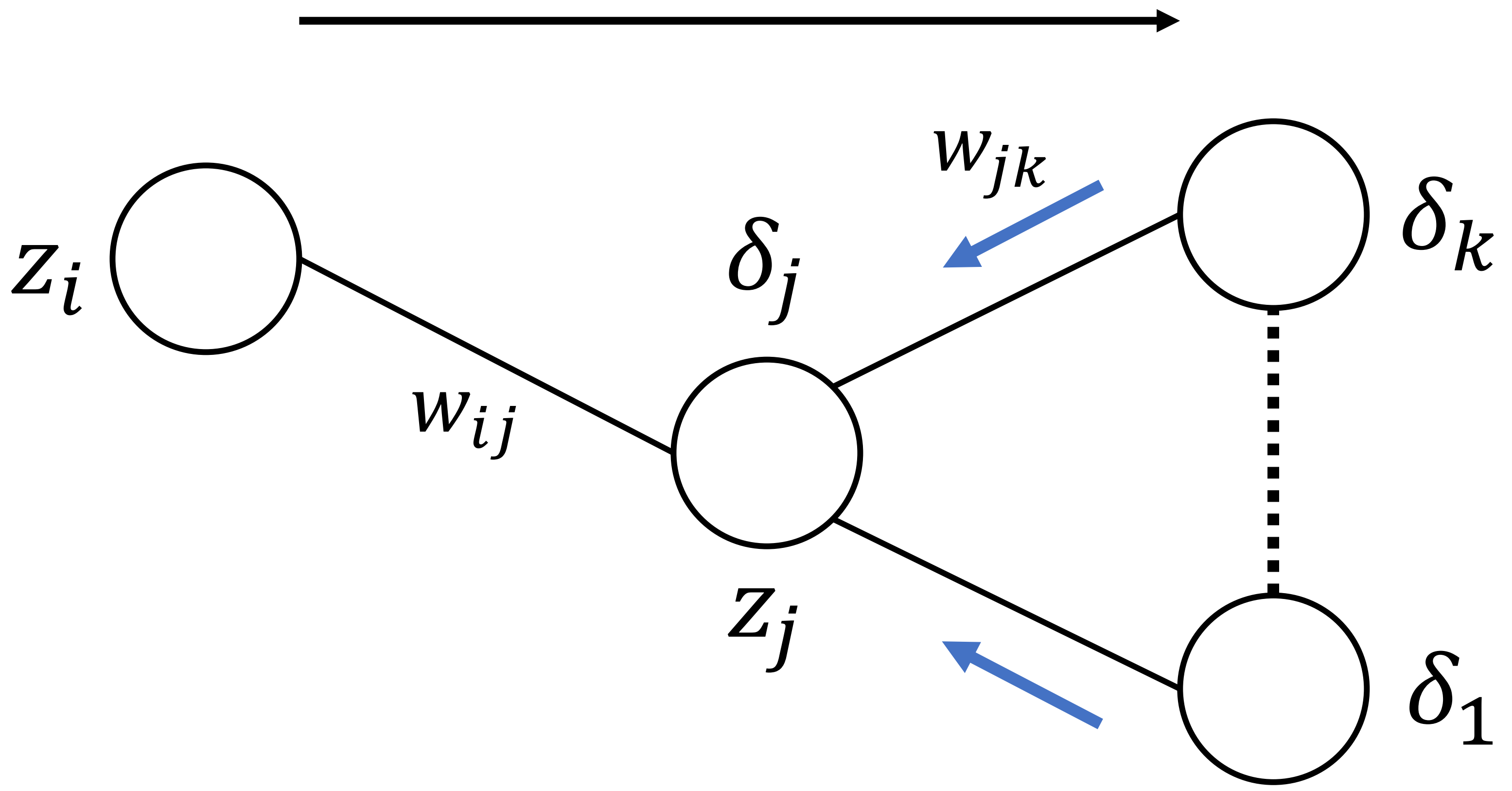
Target

- $t_n = 0.5$



Error backpropagation algorithm

- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Backprop: example

- Only one output with a linear activation function, therefore:

$$\delta_1^{(2)} = y_1 - t_{n1} = 3.64 - 0.5 = 3.14$$

- Backpropagate

$$\delta_j^{(1)} = h'(a_j) w_{j1}^{(1)} \delta_1^{(2)}$$

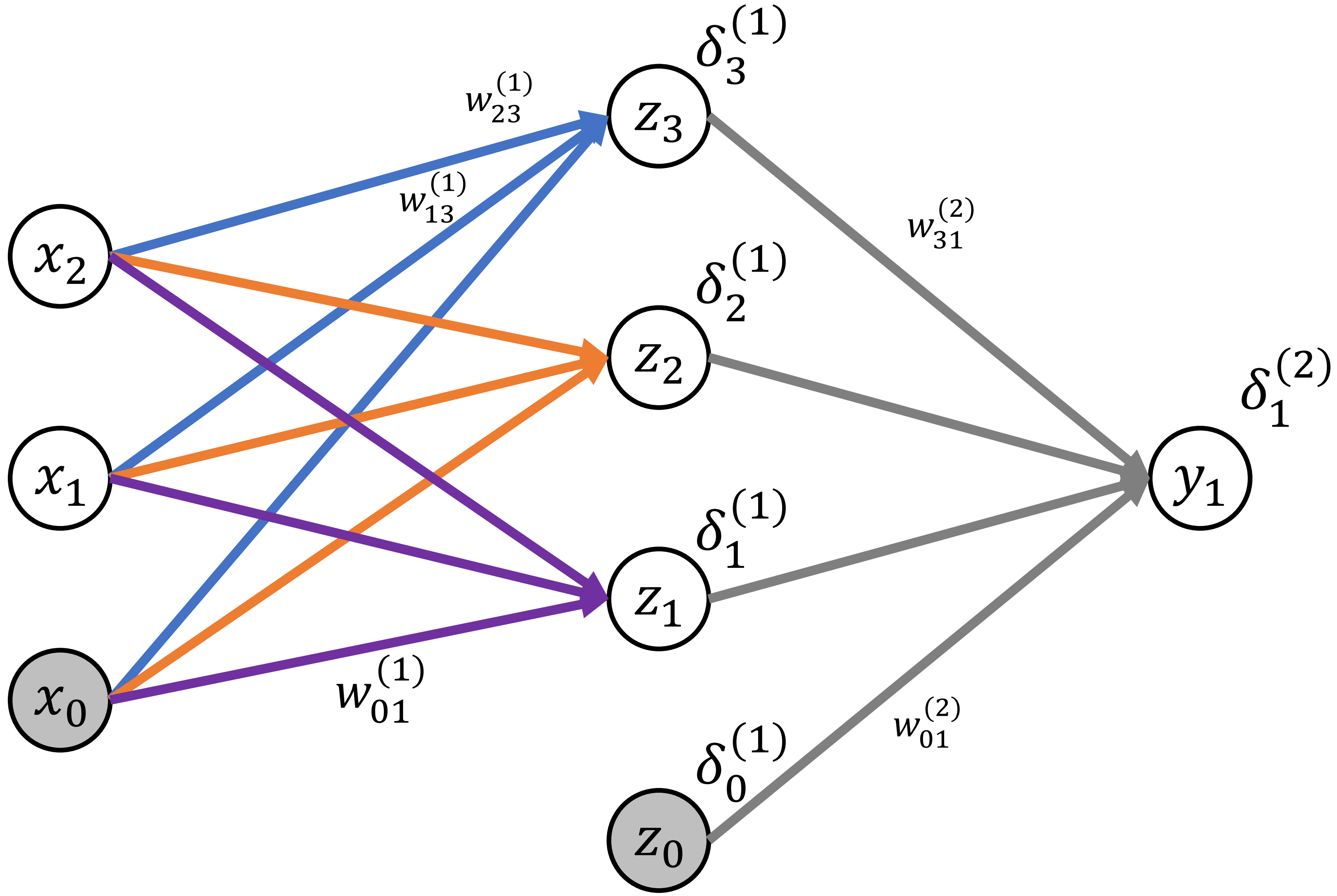
- For the sigmoid function:

$$h(a) = \frac{1}{1 + \exp(-a)} \rightarrow h'(a) = h(a)(1 - h(a))$$

- Since $z_j = h(a_j)$, the expression then becomes:

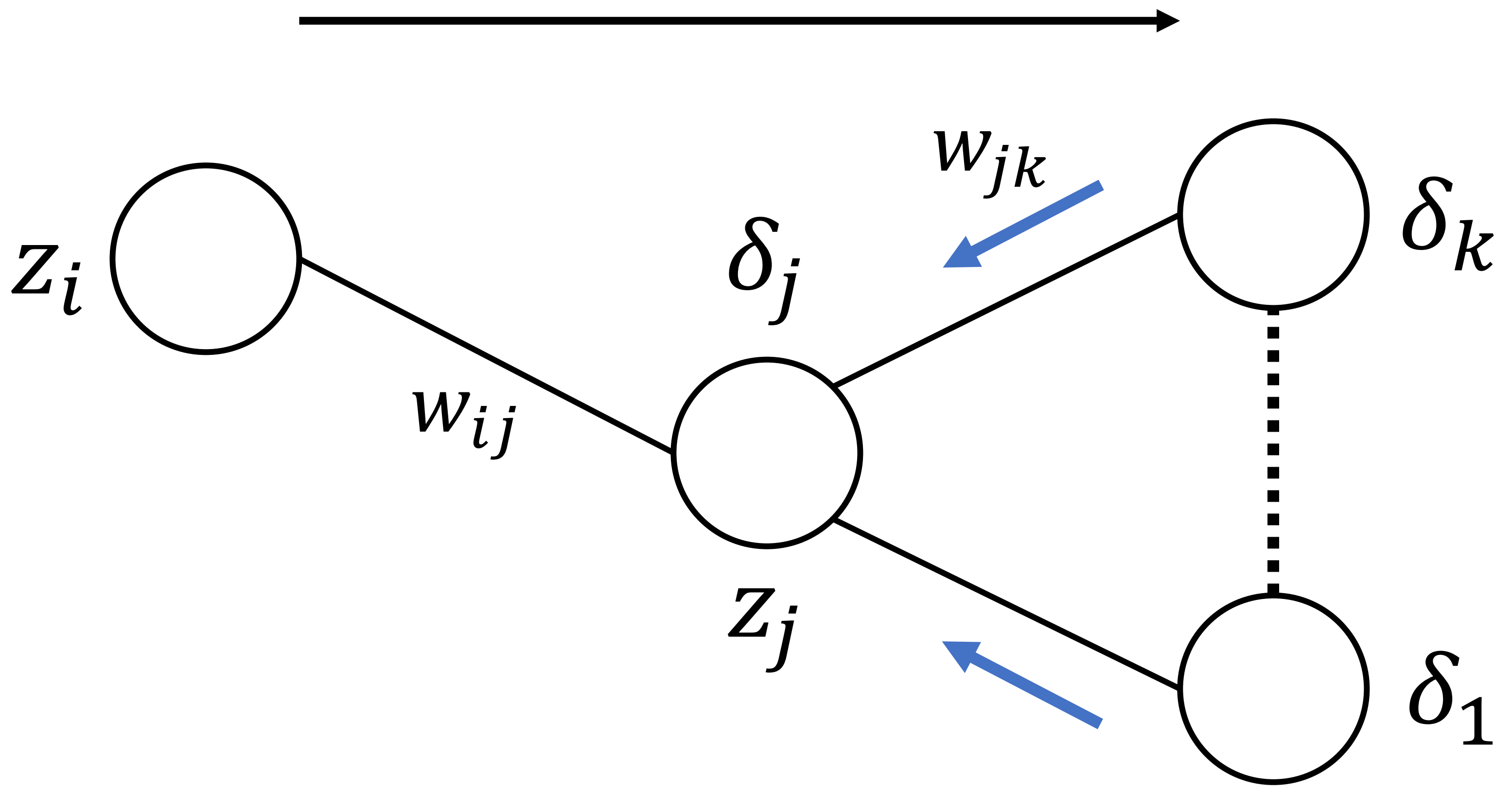
$$\delta_j^{(1)} = (z_j - z_j^2) w_{j1}^{(1)} \delta_1^{(2)}$$

- $\delta_0^{(1)} = 0,\ \delta_1^{(1)} = 0.33,\ \delta_2^{(1)} = 0.33,\ \delta_3^{(1)} = 0.33$



Error backpropagation algorithm

- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Backprop: example

- Evaluate the derivatives

$$\frac{\partial E_n}{\partial w_{jk}^{(2)}} = \delta_k z_j \rightarrow \begin{cases} \dfrac{\partial E_n}{\partial w_{01}^{(2)}} = \delta_1^{(2)} z_0 = 3.14 \times 1 = 3.14 \\ \dfrac{\partial E_n}{\partial w_{11}^{(2)}} = \delta_1^{(2)} z_1 = 3.14 \times 0.88 = 2.76 \\ \dfrac{\partial E_n}{\partial w_{21}^{(2)}} = \delta_1^{(2)} z_2 = 3.14 \times 0.88 = 2.76 \\ \dfrac{\partial E_n}{\partial w_{31}^{(2)}} = \delta_1^{(2)} z_3 = 3.14 \times 0.88 = 2.76 \end{cases}$$

Backprop: example

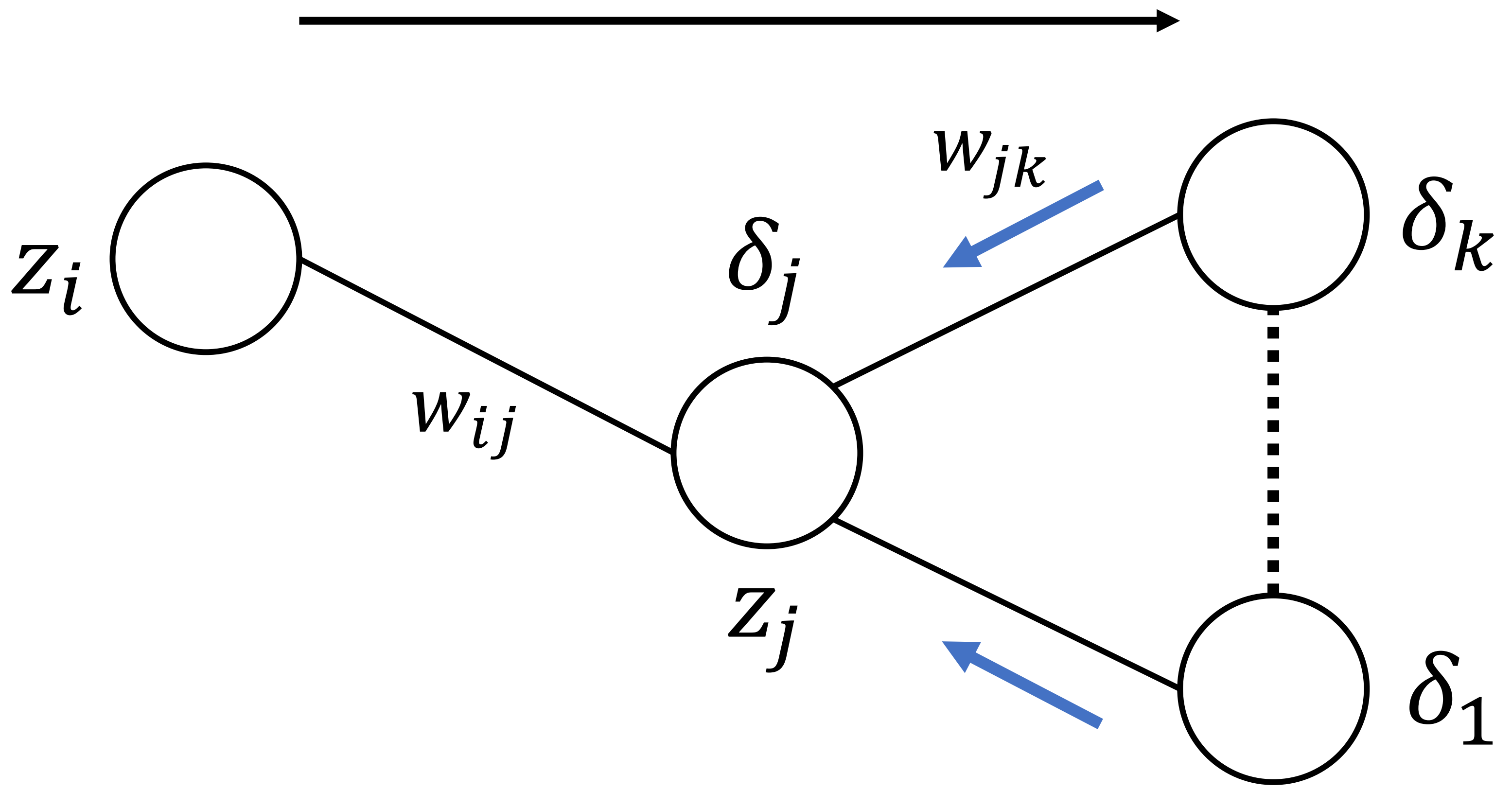
- Evaluate the derivatives

$$\frac{\partial E_n}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} x_i \rightarrow \begin{cases} \frac{\partial E_n}{\partial w_{01}^{(1)}} = \delta_1^{(1)} x_0, \frac{\partial E_n}{\partial w_{11}^{(1)}} = \delta_1^{(1)} x_1, \frac{\partial E_n}{\partial w_{21}^{(1)}} = \delta_1^{(1)} x_2 \\ \frac{\partial E_n}{\partial w_{02}^{(1)}} = \delta_2^{(1)} x_0, \frac{\partial E_n}{\partial w_{12}^{(1)}} = \delta_2^{(1)} x_1, \frac{\partial E_n}{\partial w_{22}^{(1)}} = \delta_2^{(1)} x_2 \\ \frac{\partial E_n}{\partial w_{03}^{(1)}} = \delta_3^{(1)} x_0, \frac{\partial E_n}{\partial w_{13}^{(1)}} = \delta_3^{(1)} x_1, \frac{\partial E_n}{\partial w_{23}^{(1)}} = \delta_3^{(1)} x_2 \end{cases}$$

$$\begin{cases} \frac{\partial E_n}{\partial w_{01}^{(1)}} = 0.33, \frac{\partial E_n}{\partial w_{11}^{(1)}} = 0.165, \frac{\partial E_n}{\partial w_{21}^{(1)}} = 0.165 \\ \frac{\partial E_n}{\partial w_{02}^{(1)}} = 0.33, \frac{\partial E_n}{\partial w_{12}^{(1)}} = 0.165, \frac{\partial E_n}{\partial w_{22}^{(1)}} = 0.165 \\ \frac{\partial E_n}{\partial w_{03}^{(1)}} = 0.33, \frac{\partial E_n}{\partial w_{13}^{(1)}} = 0.165, \frac{\partial E_n}{\partial w_{23}^{(1)}} = 0.165 \end{cases}$$

Error backpropagation algorithm

- Initialize the weights
- Apply input vector $\mathbf{x}_n$
- Evaluate the δ_k for all the output units
- Backpropagate the δ to obtain δ_k for each hidden unit
- Evaluate the required derivatives
- Update the weights



Backprop: example

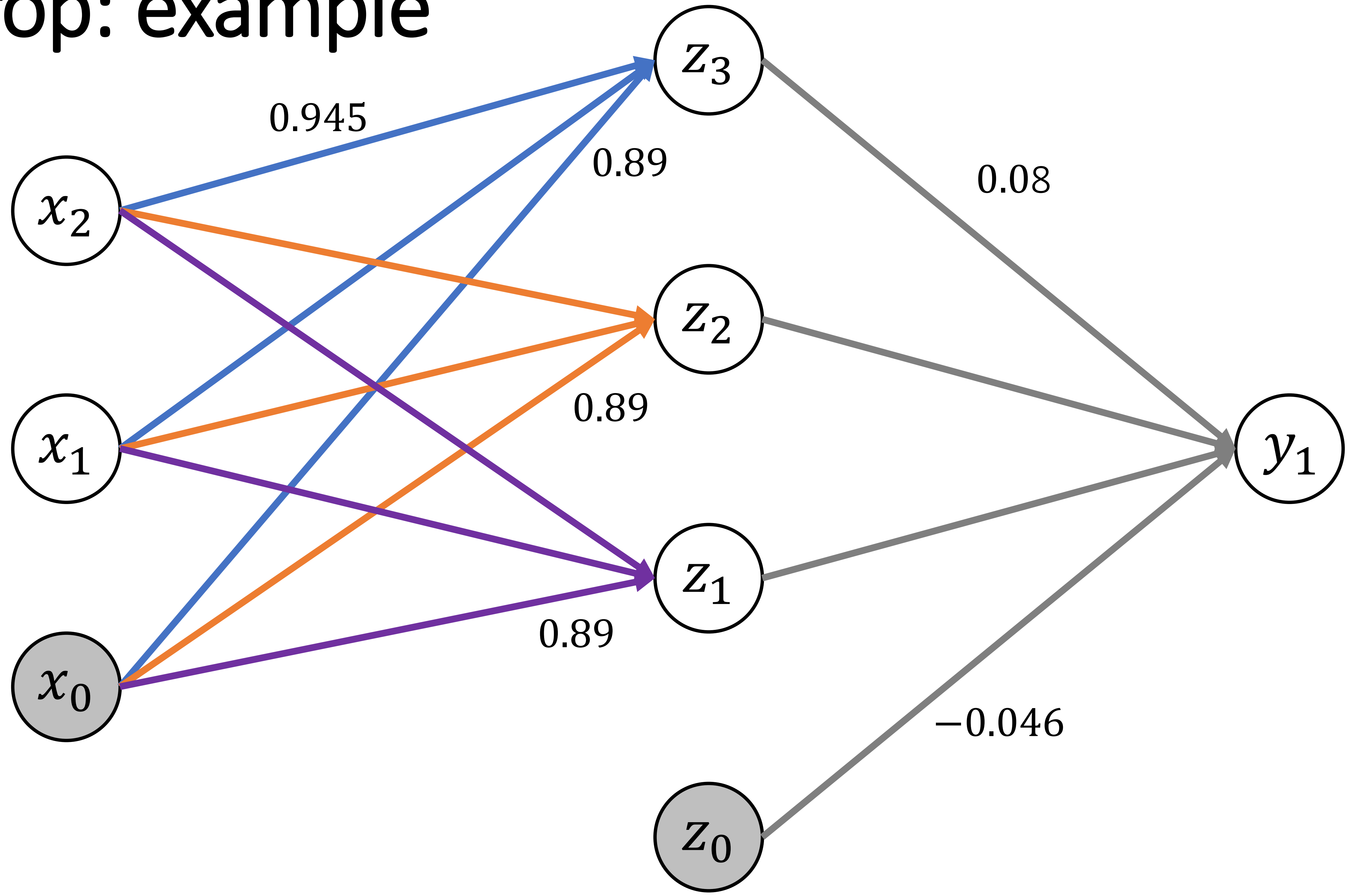
- Use the update rule and the calculated error derivatives

$$w^{new} = w^{old} - \eta \nabla E_n\left(w^{old}\right)$$

- Assuming an $\eta = \frac{1}{3}$ the new weights are calculated for the network, e.g. weight $w_{\mathbf{01}}^{(1)}$ (weight from $x_{\mathbf{0}}$ to $z_{\mathbf{1}}$ on the first layer).

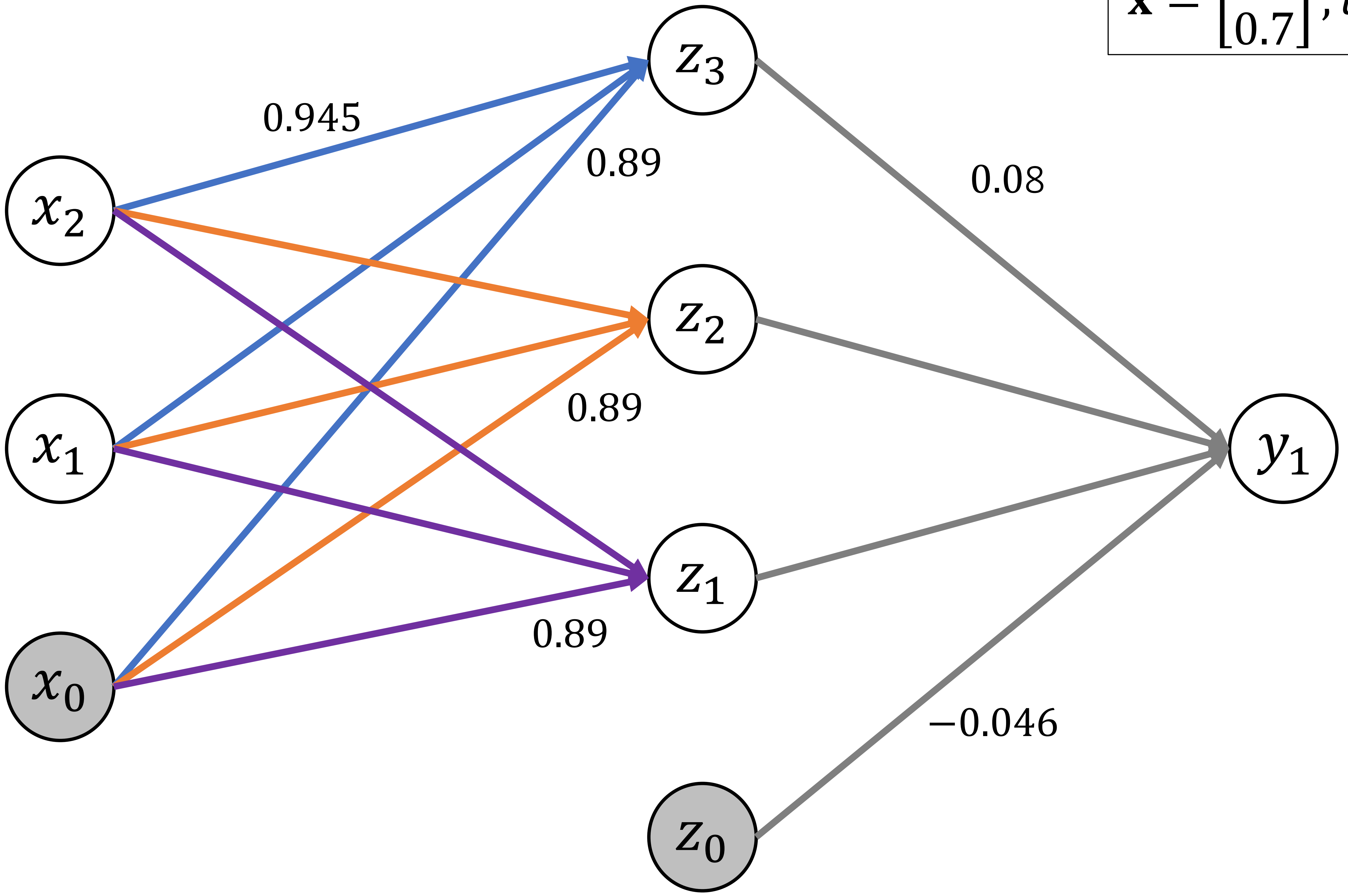
$$w_{01}^{(1),new} = w_{01}^{(1),old} - \eta \frac{\partial E_n}{\partial w_{01}^{(1),old}} = 1 - \frac{1}{3}(0.33) = 0.89$$

Backprop: example



Test

$\mathbf{x} = \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix}, t = -0.02$



Test

- Activations

$$a_1 = \left(\sum_{i=1}^{D} w_{i1}^{(1)} x_i\right) + w_{01}^{(1)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{01}^{(1)}$$

$$a_2 = \left(\sum_{i=1}^{D} w_{i2}^{(1)} x_i\right) + w_{02}^{(1)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{02}^{(1)}$$

$$a_3 = \left(\sum_{i=1}^{D} w_{i3}^{(1)} x_i\right) + w_{03}^{(1)} = w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 + w_{03}^{(1)}$$

- Hidden units

$$z_1 = h(a_1) = \frac{1}{1 + \exp(-a_1)}$$

$$z_2 = h(a_2) = \frac{1}{1 + \exp(-a_2)}$$

$$z_3 = h(a_3) = \frac{1}{1 + \exp(-a_3)}$$

Test

- Output

$$a_1^{(2)} = \left(\sum_{i=1}^{M} w_{ij}^{(2)} z_i \right) + w_{01}^{(2)} = w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2 + w_{31}^{(2)} z_3 + w_{01}^{(2)}$$

$$y_1 = \sigma\left(a_1^{(2)}\right) = a_1^{(2)}$$