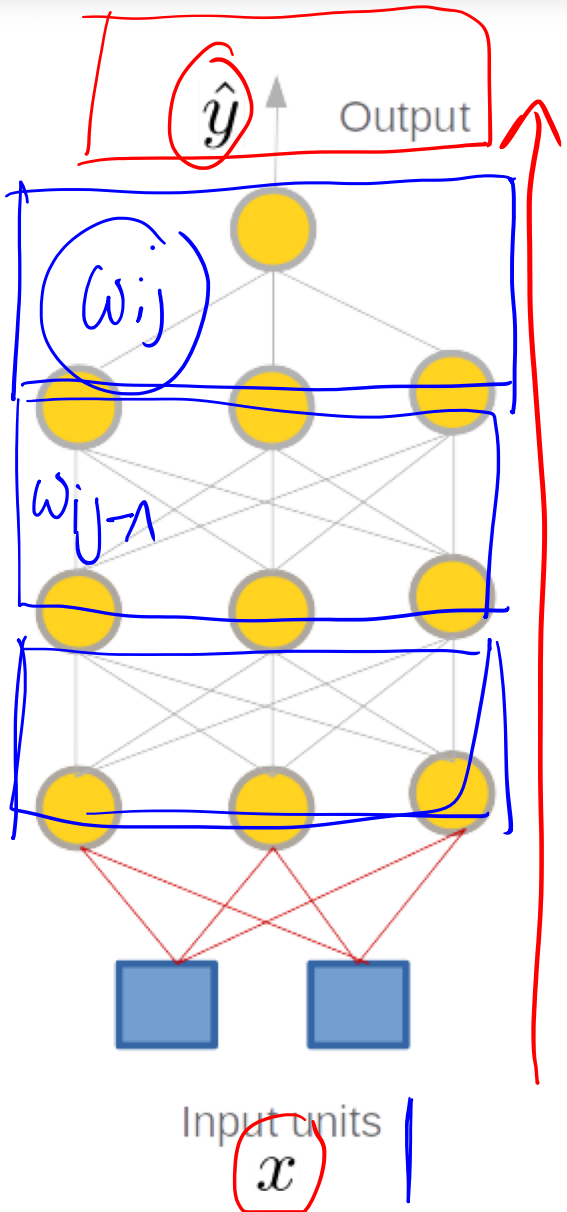
Back Propagation in Depth

Heni Ben Amor, Ph.D.
Assistant Professor
Arizona State University

ASU Ira A. Fulton Schools of Engineering
Arizona State University

Intuition of Back Propagation



1. Calculate Error at Final Layer

$$E = \frac{1}{2}(y - \hat{y})^2$$

2. Go back through layers and calculate changes to all weights

3. Repeat step 2 for all previous layers

BP = Gradient Descent

- Calculate gradient of network
- Update weights according to gradient descent

Gradient

$$\nabla E = \left(\frac{\partial E}{w_{1,1}}, \frac{\partial E}{w_{1,2}}, \ldots, \frac{\partial E}{w_{l,k}} \right)$$

nabla

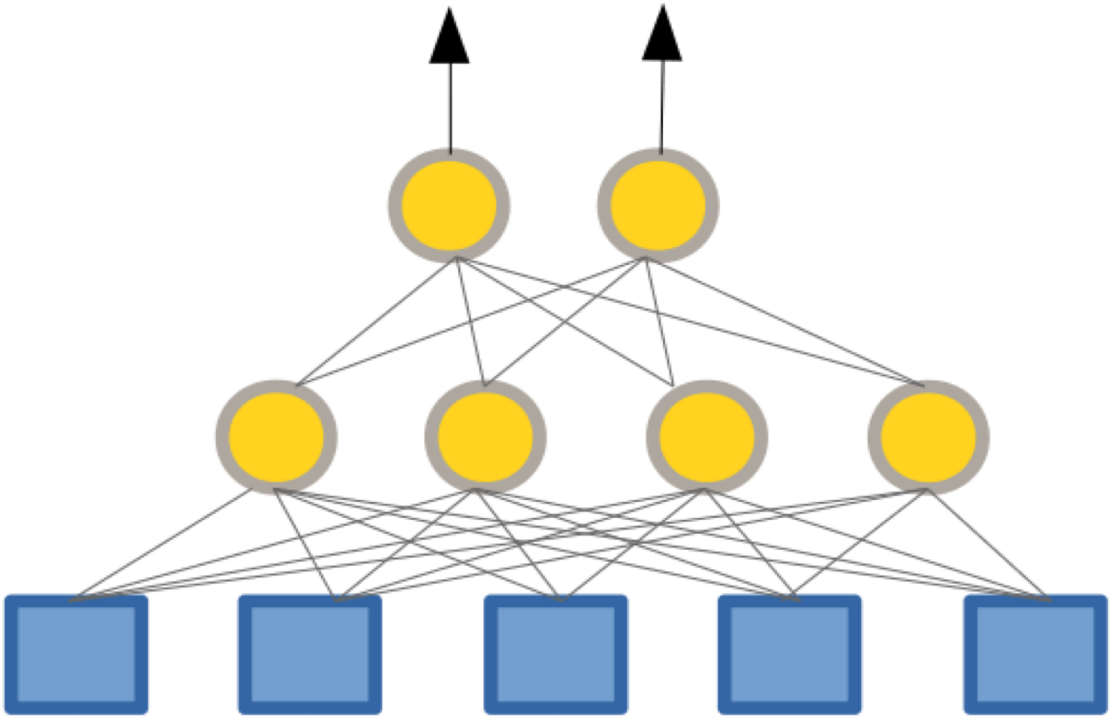
- Update equation

$$w_{i,j} \leftarrow w_{i,j} - \alpha \frac{\partial E}{w_{i,j}}$$

Learning Rate

Gradient

What is the Gradient of a Network?



Gradient Calculation

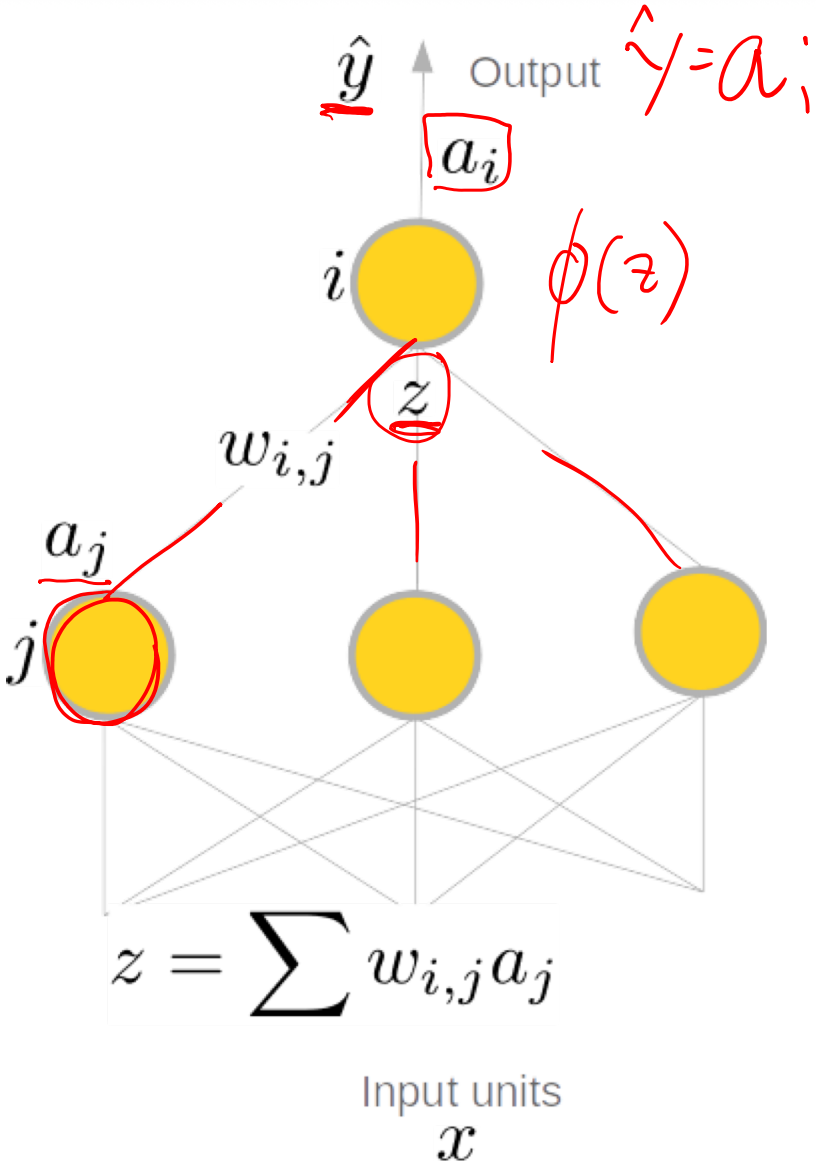
- How to get partial derivatives:

$$\nabla E = \left(\frac{\partial E}{w_1}, \frac{\partial E}{w_2}, \cdots, \frac{\partial E}{w_l} \right)$$

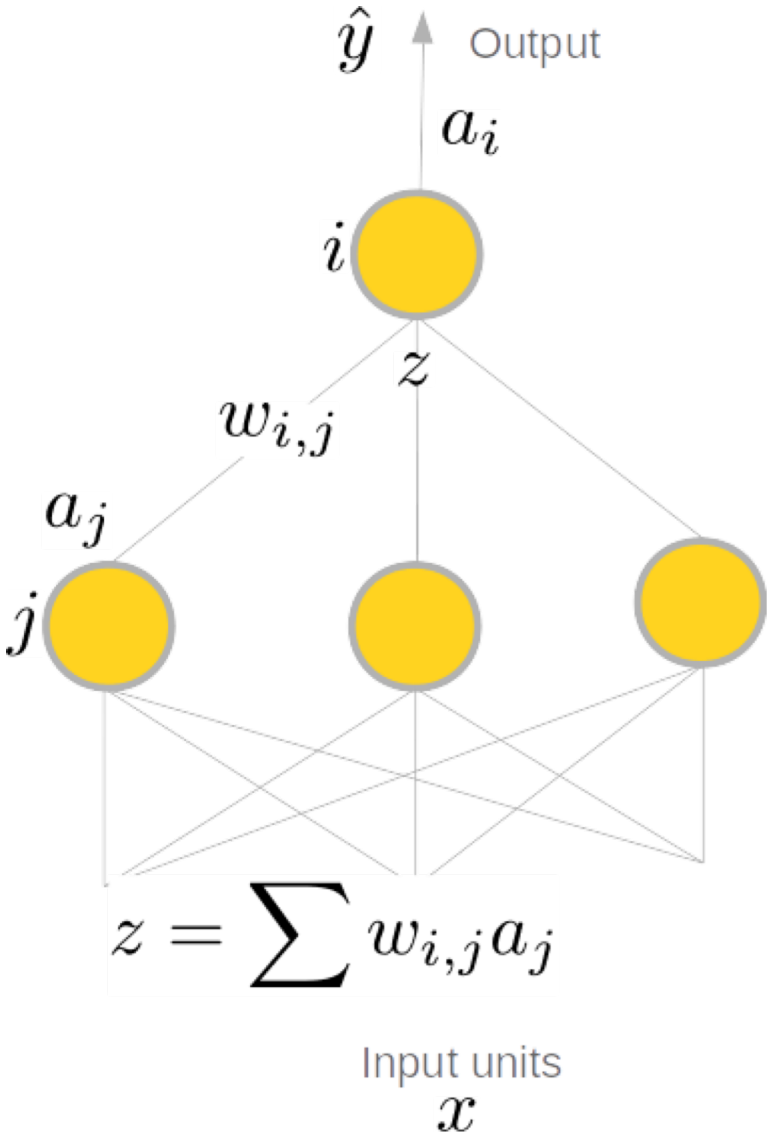
- Apply **chain rule**:

$$D\{f(g(x))\} = f'(g(x))g'(x)$$

In Depth Look at Back Propagation



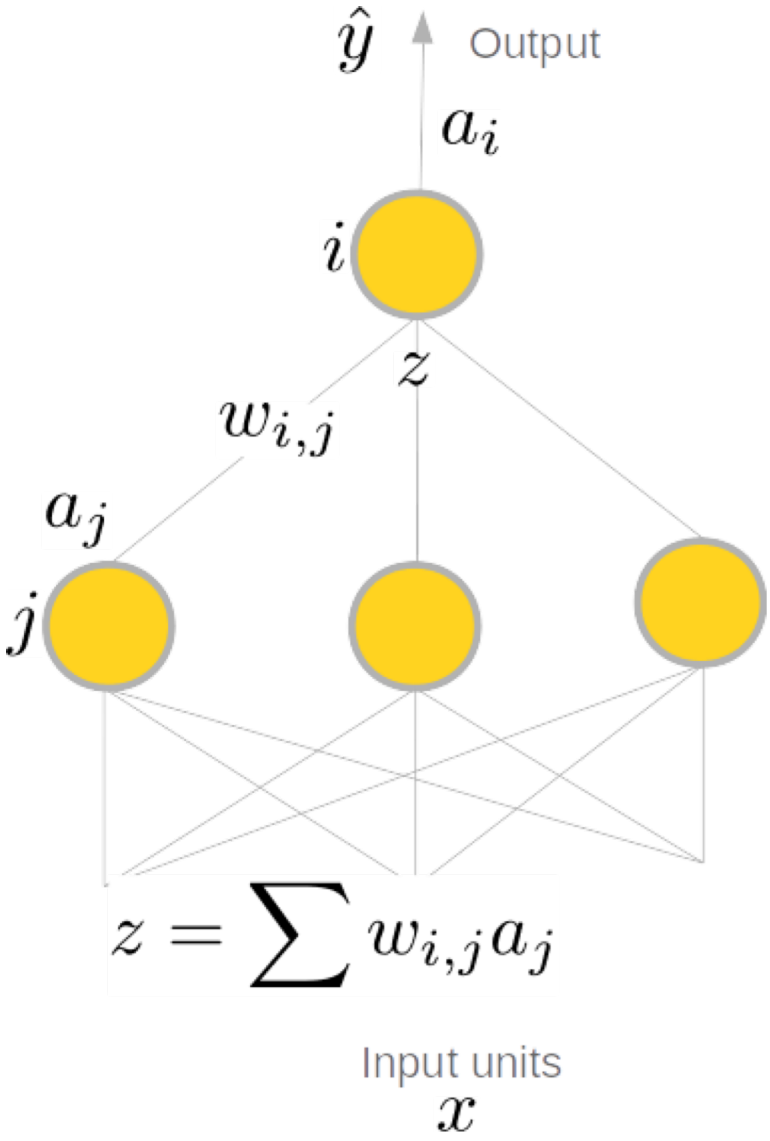
In Depth Look at Back Propagation



$$E = \frac{1}{2}(y - \hat{y})^2$$

discrepancy

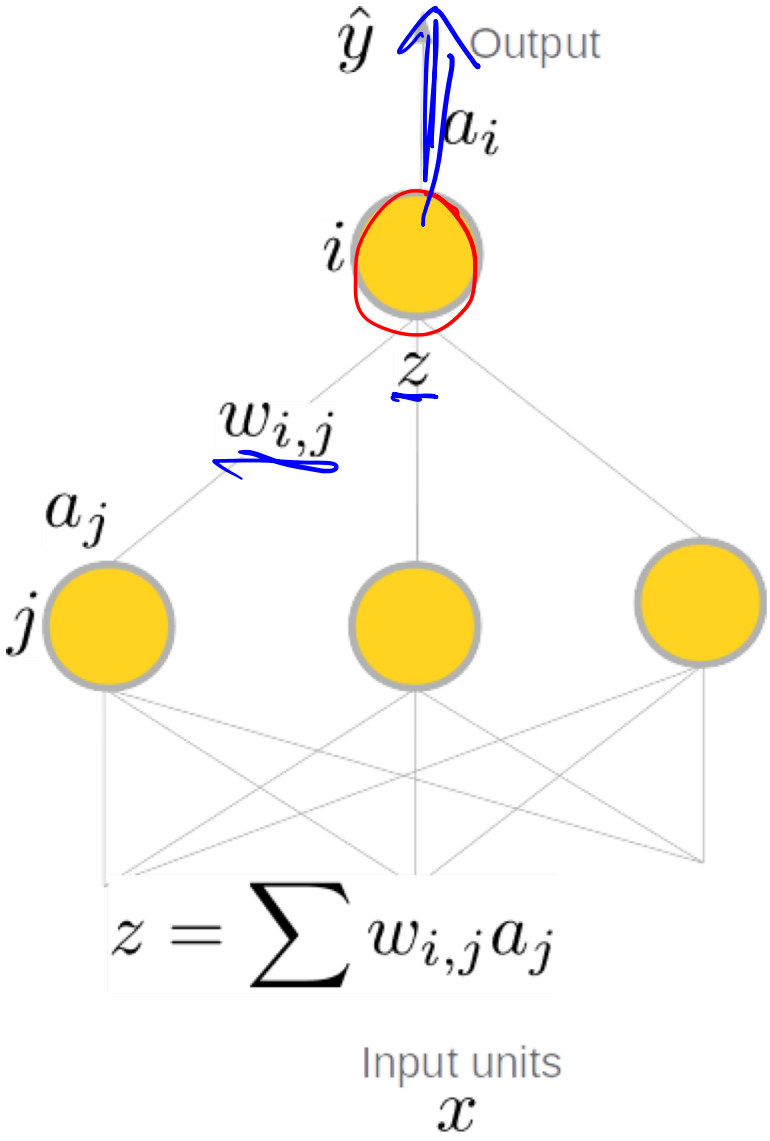
In Depth Look at Back Propagation



$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_{i,j}}$$

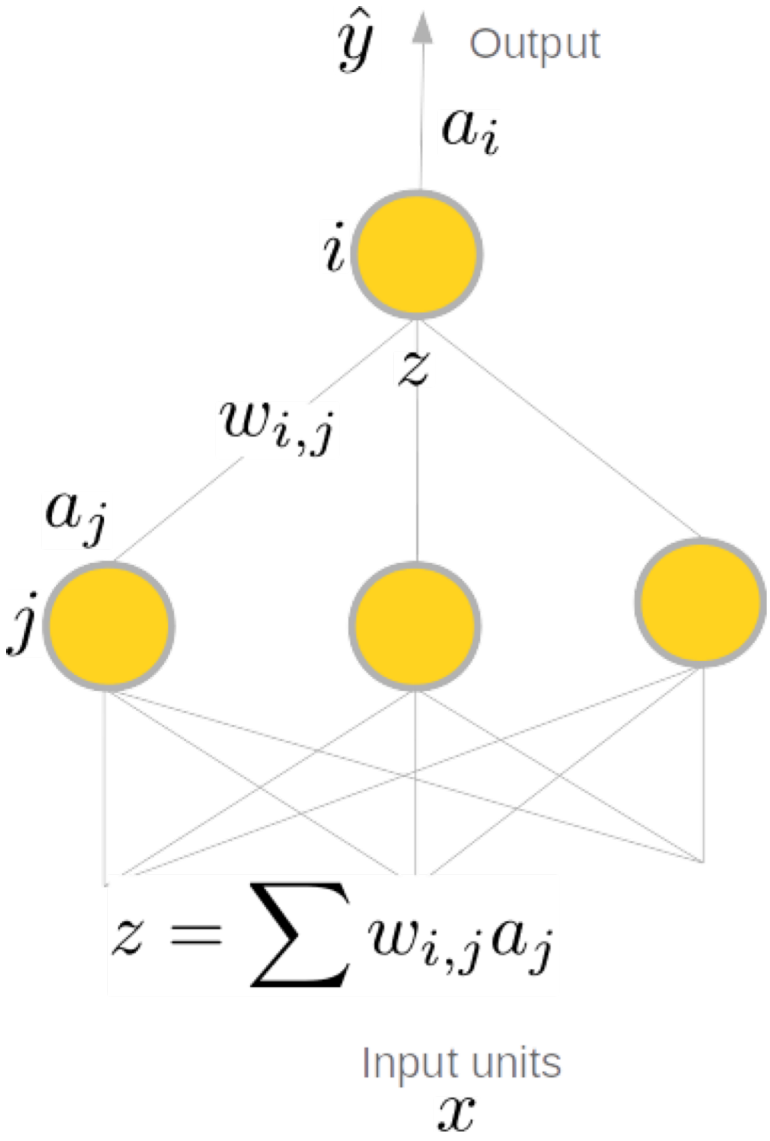
In Depth Look at Back Propagation



$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

In Depth Look at Back Propagation

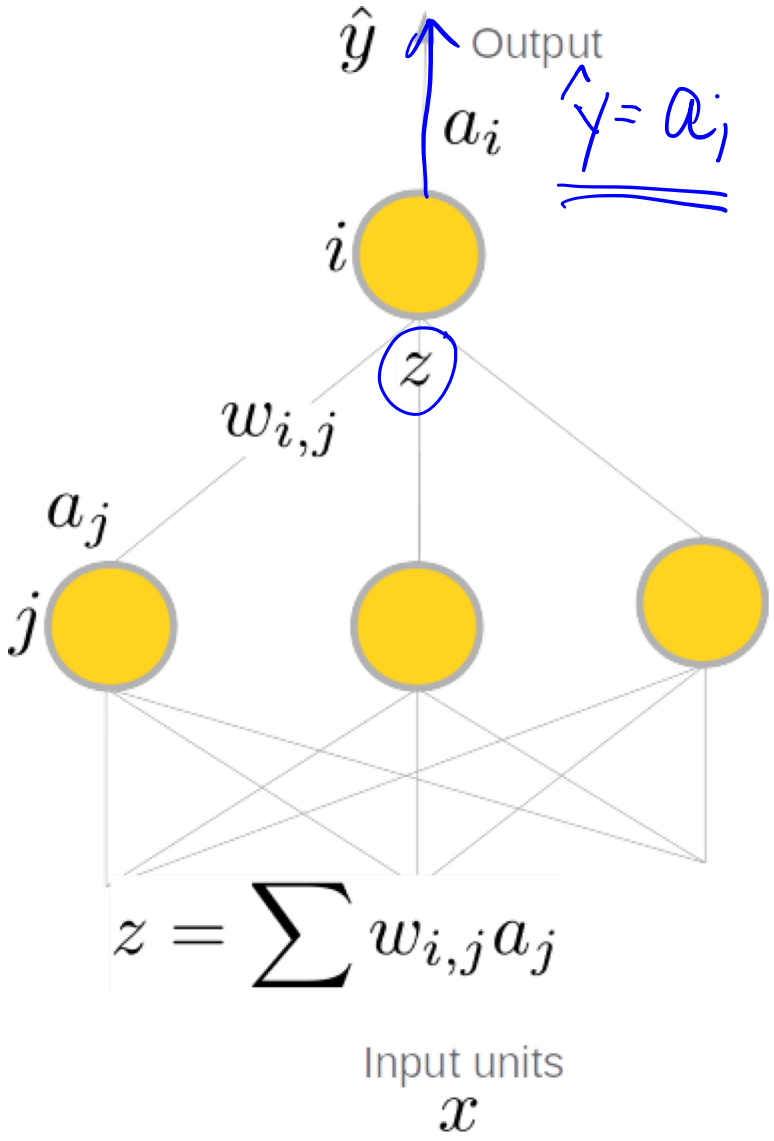


$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

In Depth Look at Back Propagation



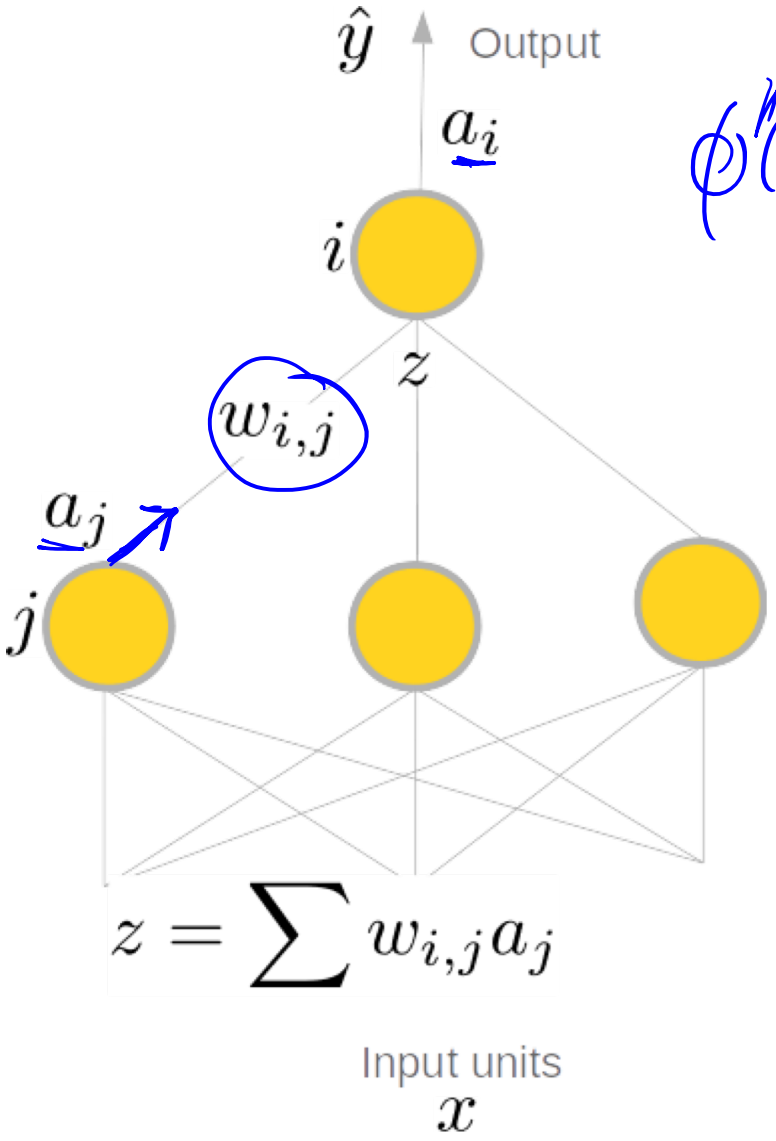
$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - a_i) \frac{\partial \phi(z)}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

In Depth Look at Back Propagation



Output error

$\phi'(z)$

$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - a_i) \frac{\partial \phi(z)}{\partial z} \frac{\partial z}{\partial w_{i,j}}$$

$$\frac{\partial E}{\partial w_{i,j}} = -(y - a_i) \phi'(z) a_j$$

Update Equation for Output Layer

$$\frac{\partial E}{\partial w_{i,j}} = -(y - a_i)\,\phi'(z)\,a_j$$

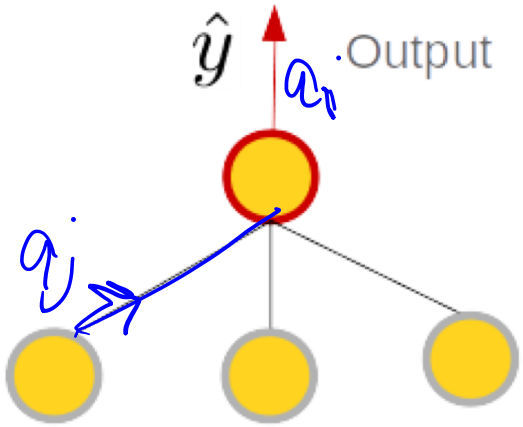
Difference to target

Derivative of Activation Function

Activation

previous layer

Output and Hidden Layer Updates in Back Propagation with Sigmoids

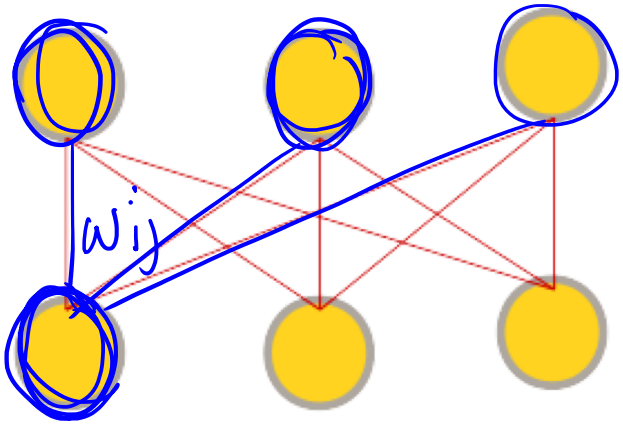


$$\frac{\partial E}{\partial w_{i,j}} = \delta_i a_j$$

w_{ij} leading into output lay

$$\delta_i = a_i(1 - a_i)(a_i - y)$$

Hidden Layer



$$\frac{\partial E}{\partial w_{i,j}} = \delta_i a_j$$

$$\delta_i = a_i(1 - a_i) \sum_{h \in succ(i)} \delta_h w_{i,h}$$

Back Propagation Algorithm

- **Initialize all weights randomly**
- **Until convergence do:**
  - Input example and calculate network output
  - For each output unit do:

$$\delta_i = a_i(1 - a_i)(a_i - y)$$

  - For each hidden unit do:

$$\delta_i = a_i(1 - a_i) \sum_{h \in succ(i)} \delta_h w_{i,h}$$

  - Update weights of connection between i and j:

$$w_{i,j} \leftarrow w_{i,j} - \alpha \delta_i a_j$$

E(NN) < threshold

previous activation

learning rate α

Summary

- Back propagation algorithm
- Generates the weights of the network
- Objective is to minimize the training error
- Perform gradient descent
- Calculate derivative of network
- Apply chain rule to get partial derivatives
- Derivatives are propagated backwards