JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program

605-447/625-438

Dr. Mark Fleischer

Module 5.1: The Feed-forward, Back Propagation Algorithm

This Sub-Module Covers …

- Extends the Perceptron Delta Function to handle multi-layer networks.
- We will derive the feed-forward, back-propagation algorithm.
  - Similar in spirit the the Perceptron Delta Function.
  - Calculus based optimization technique.
- Next video we go through a computational example.

The Perceptron Delta Function

Using the Chain Rule, we get:

$$\frac{\partial E_j}{\partial w_{ij}} = \frac{\partial E_j}{\partial e_j} \times \frac{\partial e_j}{\partial y_j} \times \frac{\partial y_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}}$$

Factors: 1, 2, 3, 4

The Perceptron Delta Function

$$\frac{\partial E_j}{\partial w_{ij}} = \frac{\partial E_j}{\partial e_j} \times \frac{\partial e_j}{\partial y_j} \times \frac{\partial y_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial \omega_{ij}} = -e_j[1 - y_j]y_j x_i$$

From input *i* to output *j.*

The Perceptron Delta Function

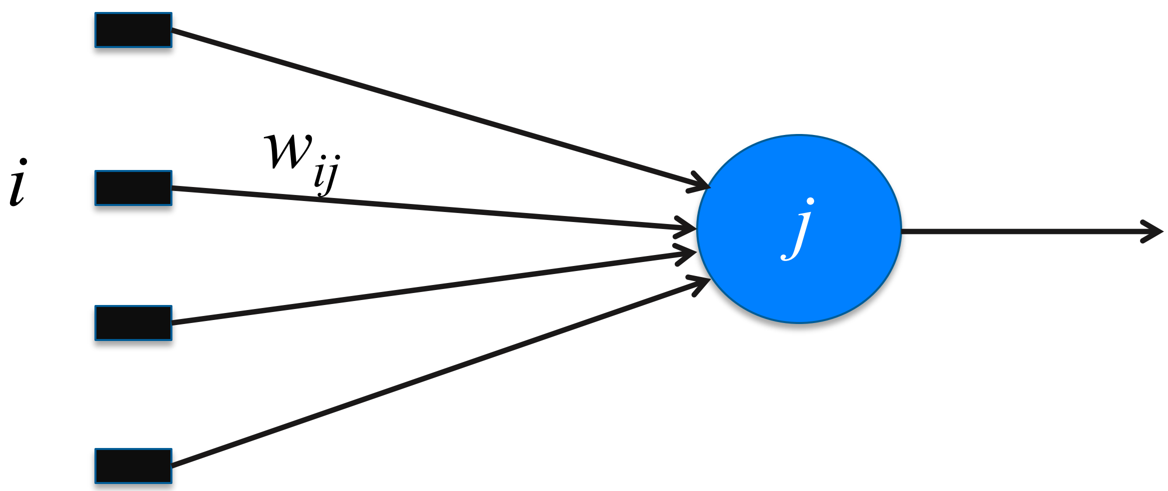
$$\frac{\partial E}{\partial \omega_{ij}} = -e_j[1 - y_j]y_j x_i$$

and letting $\delta_j = \quad e_j[1 - y_j]y_j$ then

$$\Delta\omega_{ij} = \eta \frac{\partial E}{\partial \omega_{ij}} = \eta \delta_j x_i.$$

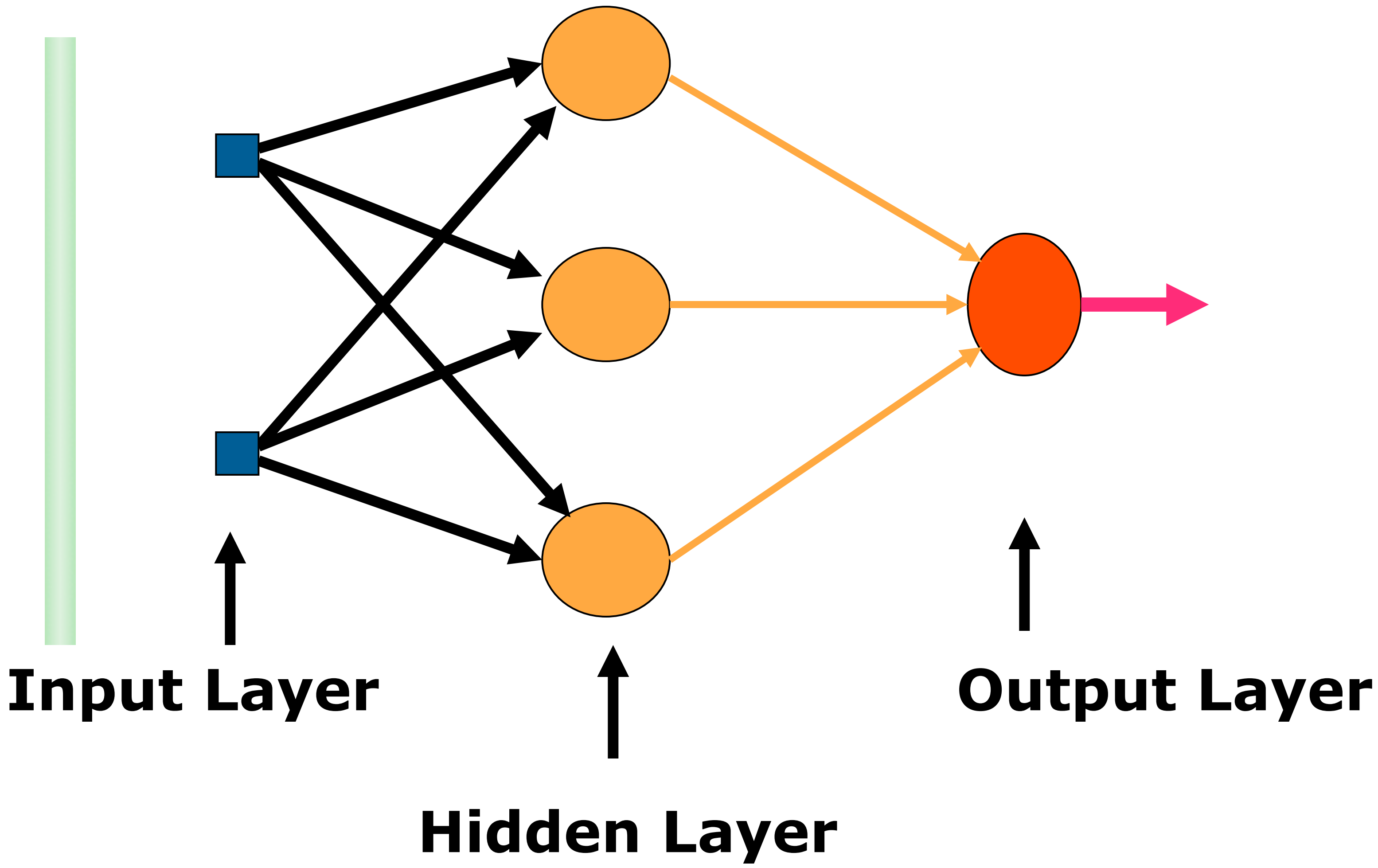
From input *i* to output *j*.

The Perceptron



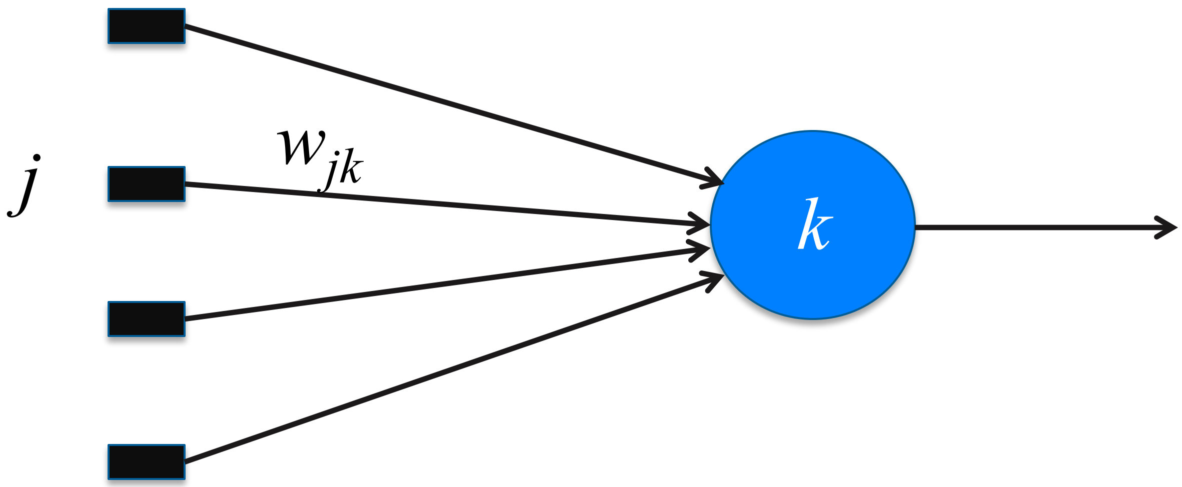
$$\frac{\partial E_j}{\partial w_{ij}}$$

A Multi-layered Network



The Perceptron

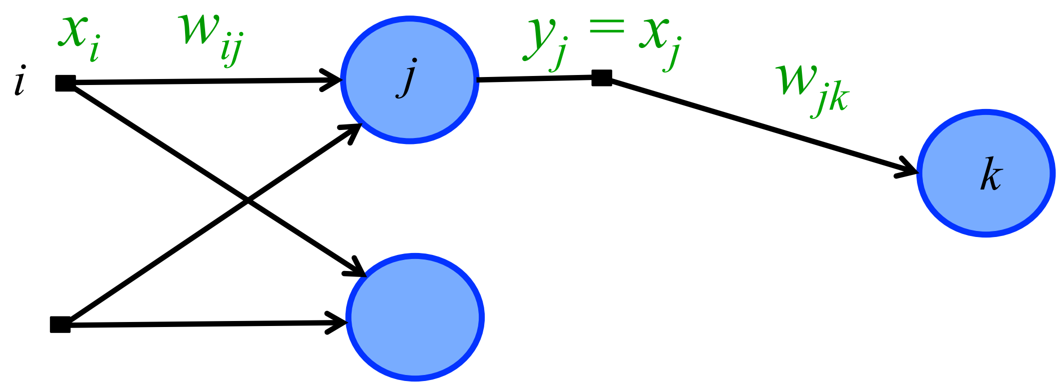
New naming conventions for the output nodes.



$$\frac{\partial E_k}{\partial w_{jk}}$$

The Feed-forward Back-propagation Algorithm

Notational Conventions for a multi-layered network



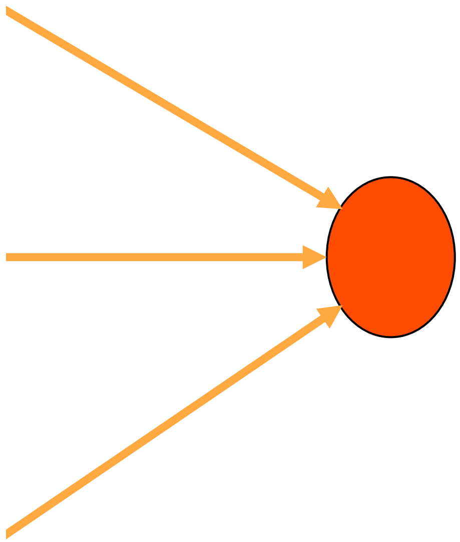
The Gradient Vector

was just $\left(\frac{\partial E_k}{\partial w_{1k}}, \frac{\partial E_k}{\partial w_{2k}}, \ldots, \frac{\partial E_k}{\partial w_{nk}}\right)$

but in a multi-layer network, it becomes

$$\left(\underbrace{\frac{\partial E_k}{\partial w_{1k}}, \frac{\partial E_k}{\partial w_{2k}}, \ldots, \frac{\partial E_k}{\partial w_{nk}}}_{\text{Output Layer Node}}, \underbrace{\frac{\partial E_k}{\partial w_{1j_1}}, \frac{\partial E_k}{\partial w_{2j_1}}, \ldots, \frac{\partial E_k}{\partial w_{mj_1}}}_{\text{Hidden Layer Node 1}}, \underbrace{\frac{\partial E_k}{\partial w_{1j_2}}, \frac{\partial E_k}{\partial w_{2j_2}}, \ldots, \frac{\partial E_k}{\partial w_{mj_2}}}_{\text{Hidden Layer Node 2}}, \ldots\right)$$

A Multi-layered Network



The Feed-forward Back-Propagation Algorithm

From this

$$\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial e_k} \times \frac{\partial e_k}{\partial y_k} \times \frac{\partial y_k}{\partial A_k} \times \frac{\partial A_k}{\partial w_{jk}}$$

to this

$$\frac{\partial E_k}{\partial w_{ij}} = \frac{\partial E_k}{\partial x_j} \times \frac{\partial x_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}}$$

The Feed-forward Back-Propagation Algorithm

$$\frac{\partial E_k}{\partial w_{ij}} = \frac{\partial E_k}{\partial x_j} \times \frac{\partial x_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}}$$

Factors: 1, 2, 3

The Feed-forward Back-Propagation Algorithm

Factor 1:

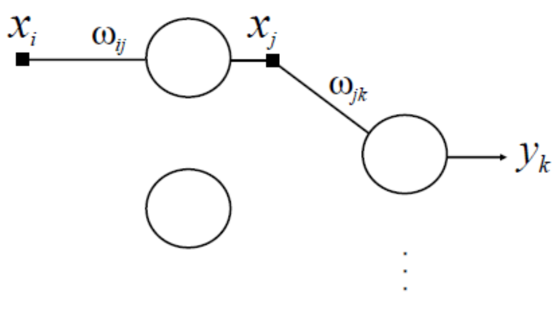
$$\frac{\partial E_k}{\partial x_j} = \frac{\partial}{\partial x_j}\frac{1}{2}\sum_k e_k^2$$

$$= \frac{1}{2}\sum_k \frac{\partial}{\partial x_j} e_k^2$$

$$= \sum_k e_k \frac{\partial e_k}{\partial x_j}$$

$$= \sum_k e_k \frac{\partial e_k}{\partial A_k} \cdot \frac{\partial A_k}{\partial x_j}$$

The Feed-forward Back-propagation Algorithm



$$\sum_k \boxed{e_k \frac{\partial e_k}{\partial A_k}} \cdot \boxed{\frac{\partial A_k}{\partial x_j}}$$

$$\begin{aligned} e_k &= d_k - y_k \\ &= d_k - f_k(A_k) \end{aligned}$$

$$\therefore \frac{\partial e_k}{\partial A_k} = -f_k'(A_k)$$

Since $A_k = \sum_{j=1}^{M} x_j w_{jk}$

then $$\frac{\partial A_k}{\partial x_j} = \frac{\partial \sum_{j=1}^{M} x_j w_{jk}}{\partial x_j}$$

$$= w_{jk}$$

The Feed-forward Back-propagation Algorithm

Factor 1:

$$\frac{\partial E_k}{\partial x_j} = \frac{\partial}{\partial x_j}\frac{1}{2}\sum_k e_k^2$$

$$= \frac{1}{2}\sum_k \frac{\partial}{\partial x_j} e_k^2$$

$$= \sum_k e_k \frac{\partial e_k}{\partial x_j}$$

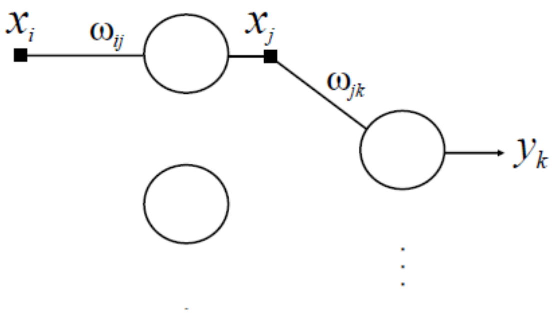
$$= \sum_k e_k \frac{\partial e_k}{\partial A_k} \cdot \frac{\partial A_k}{\partial x_j}$$

$$\frac{\partial E_k}{\partial x_j} = -\sum_k e_k f_k'(A_k) w_{jk}$$

$$= \sum_k \delta_k w_{jk}$$

FFBP: Factors 2 and 3

$$\frac{\partial x_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}} = [1 - x_j] x_j x_i$$



$$A_j = \sum_i w_{ij} x_i$$

FFBP --- All Together Now

$$\frac{\partial E_k}{\partial w_{ij}} = \frac{\partial E_k}{\partial x_j} \times \frac{\partial x_j}{\partial A_j} \times \frac{\partial A_j}{\partial w_{ij}}$$

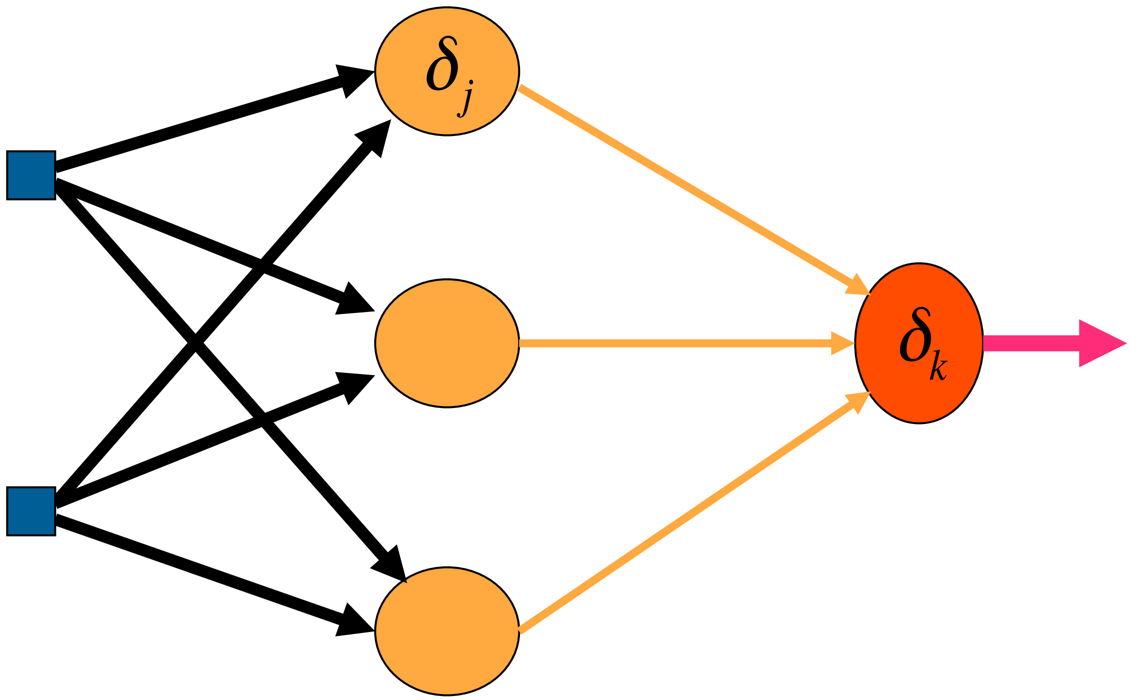
$$\sum_k \delta_k w_{jk} \quad \times \quad [1 - x_j]x_j \quad \times \quad x_i$$

FFBP --- All Together Now

$$\frac{\partial E}{\partial w_{ij}} = \boxed{[1 - x_j]x_j\left(\sum_k \delta_k w_{jk}\right)}x_i = \delta_j x_i$$

$$\Delta w_{ij} = \eta \delta_j x_i$$

A Multi-Layered Network



Introduction to Neural Networks

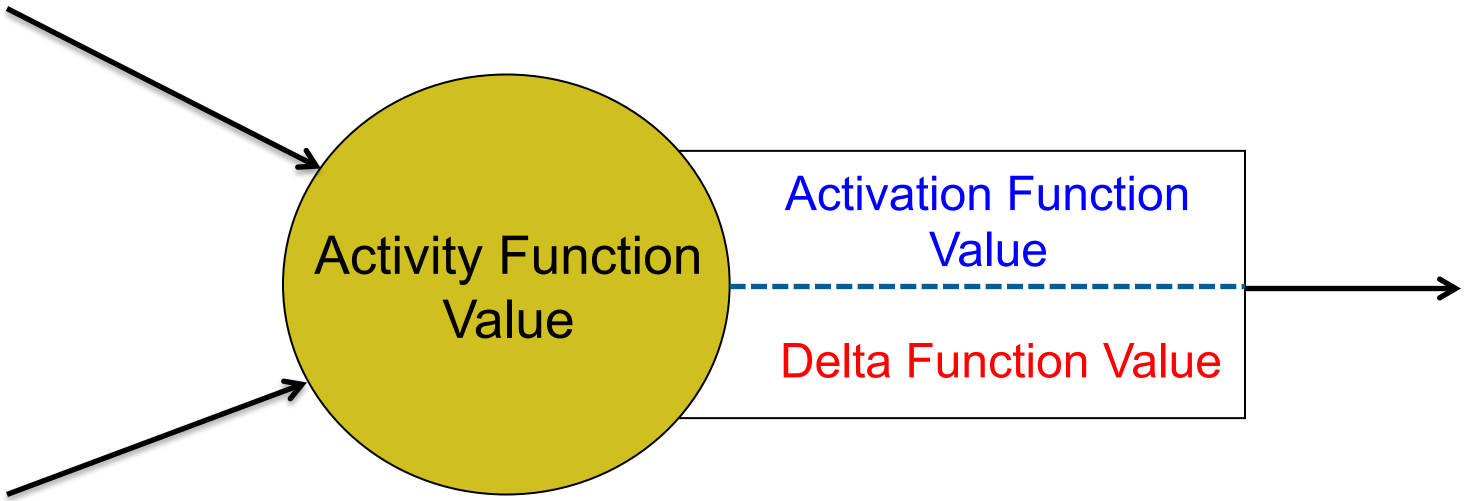
Johns Hopkins University
Engineering for Professionals Program
605-447/625-438
Dr. Mark Fleischer

Module 5.2: An Example of the FFBP

This Sub-Module Covers …

- An example of applying the FFBP algorithm to a simple, multi-layer network.
- It will demonstrate the structure and behavior of the FFBP.

Conventions for Our Perceptron

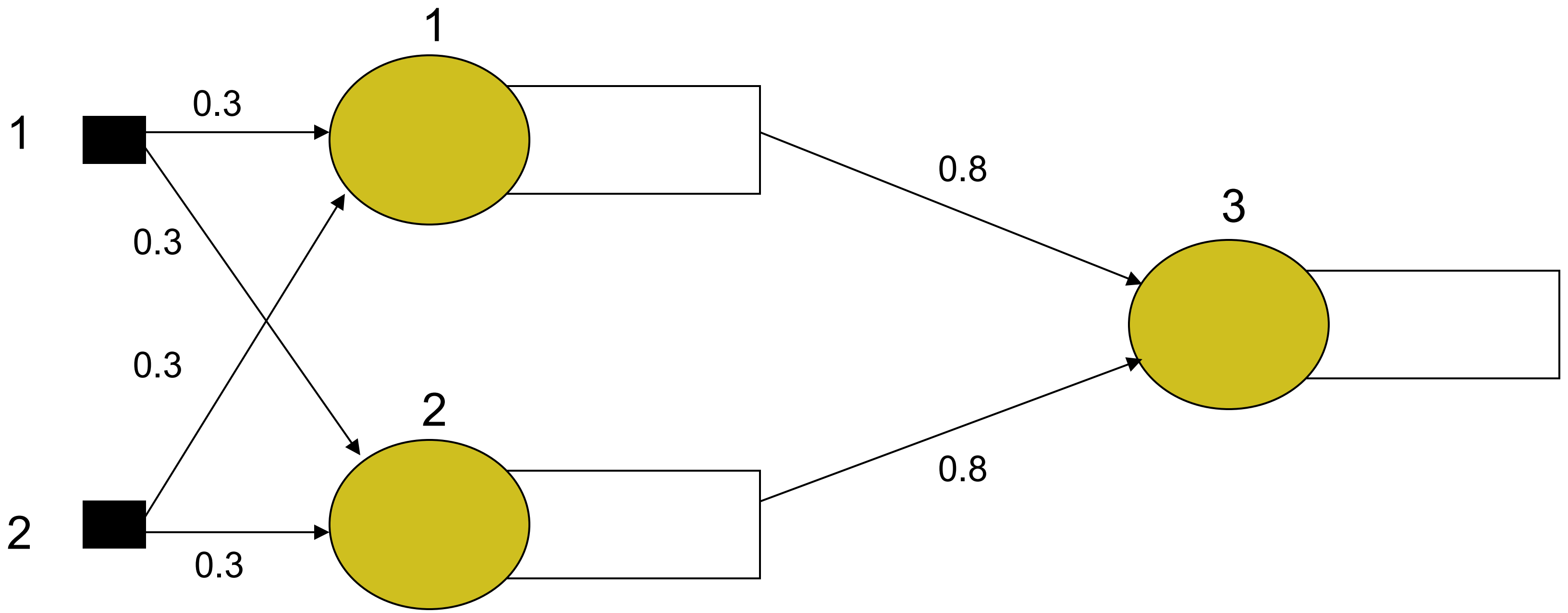


An FFBP Example

- A simple two input, single output, two layer network
- Inputs are 1 and 2: I = {1,2}
- Want to train the network so the inputs produce an output value of 0.7. Thus, {1,2} → {0.7}
- Arbitrarily set initial weights for all links.
- For convenience, we'll set the biases all to 0.
- Set all the weights of the hidden layer to 0.3.
- Set all the weights in the output layer to 0.8.
- Set the value of $\eta = 1.0$ (that's convenient!).

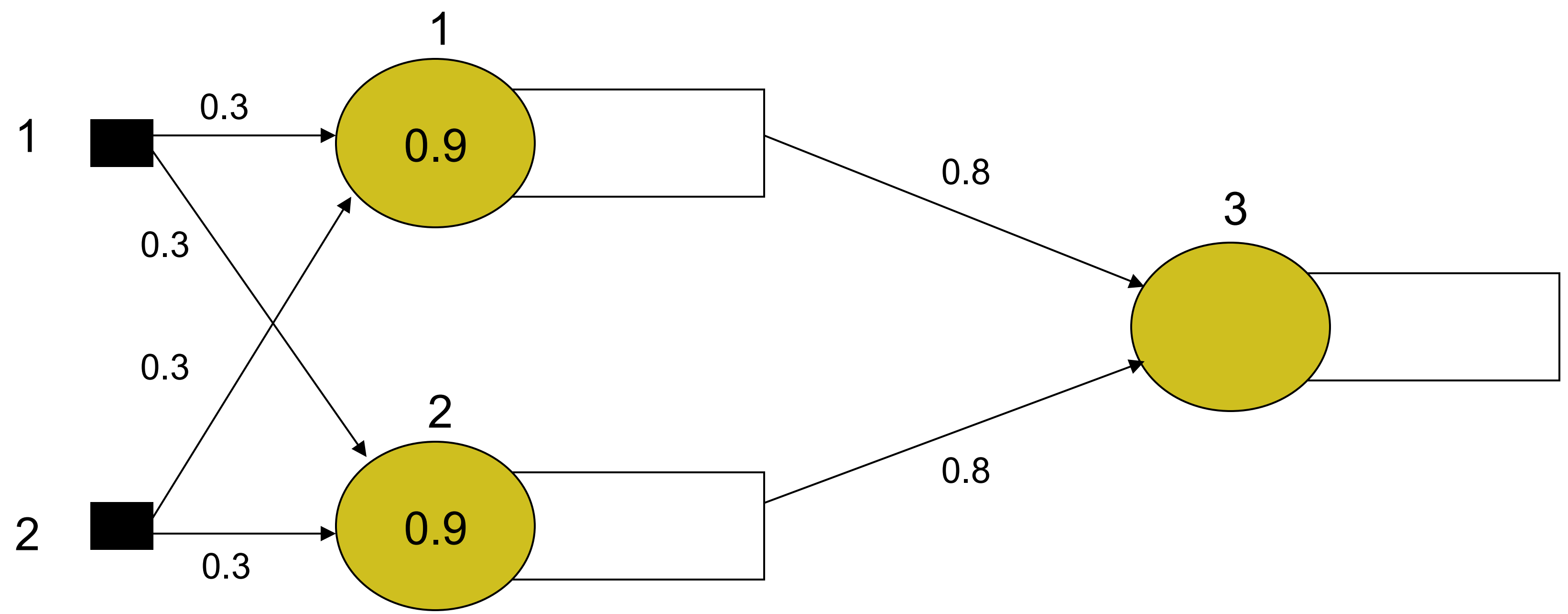
Topology of Net

After Initialization of Weights

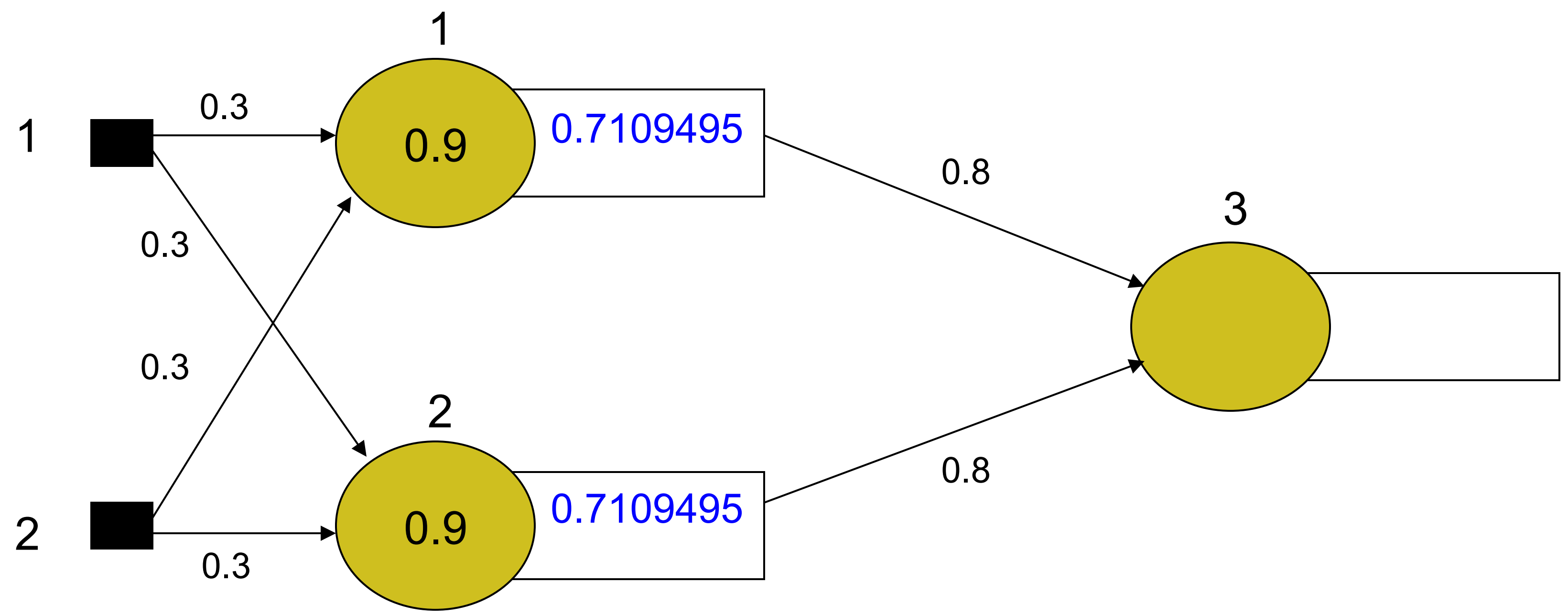


Topology of Net

Feed-forward Epoch

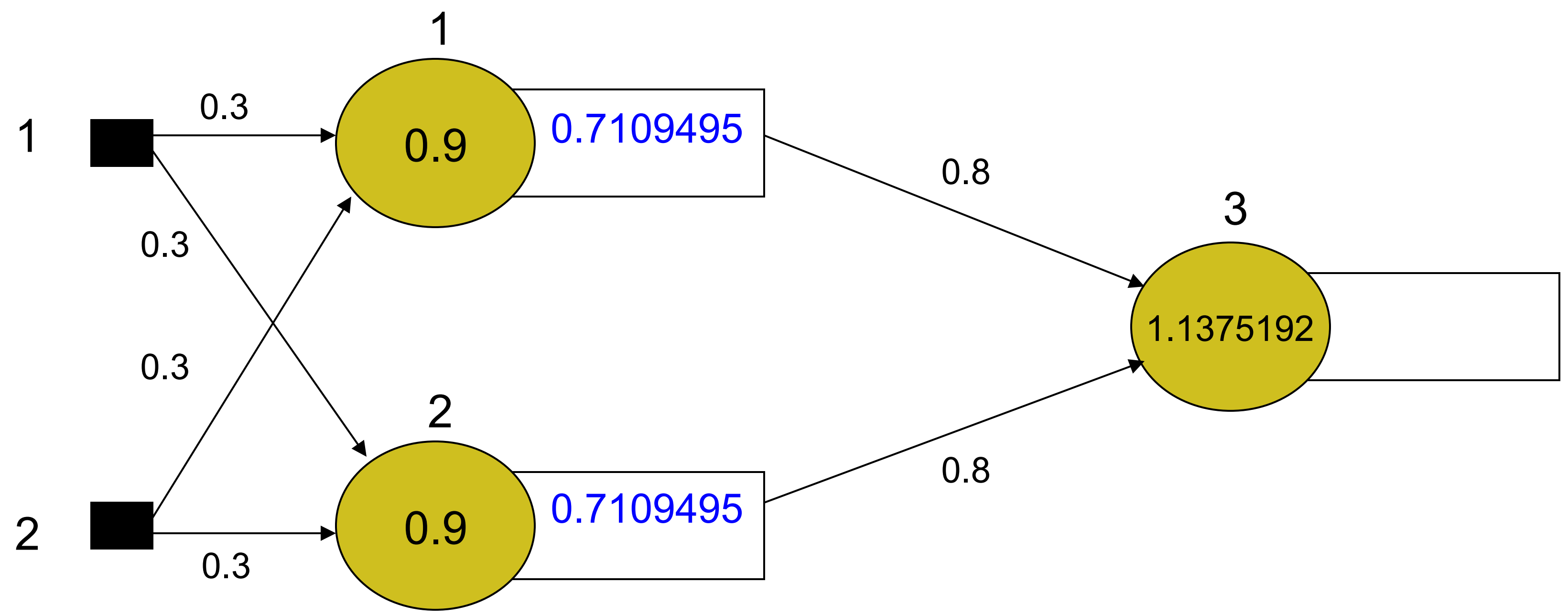


Topology of Net
Feed-forward Epoch



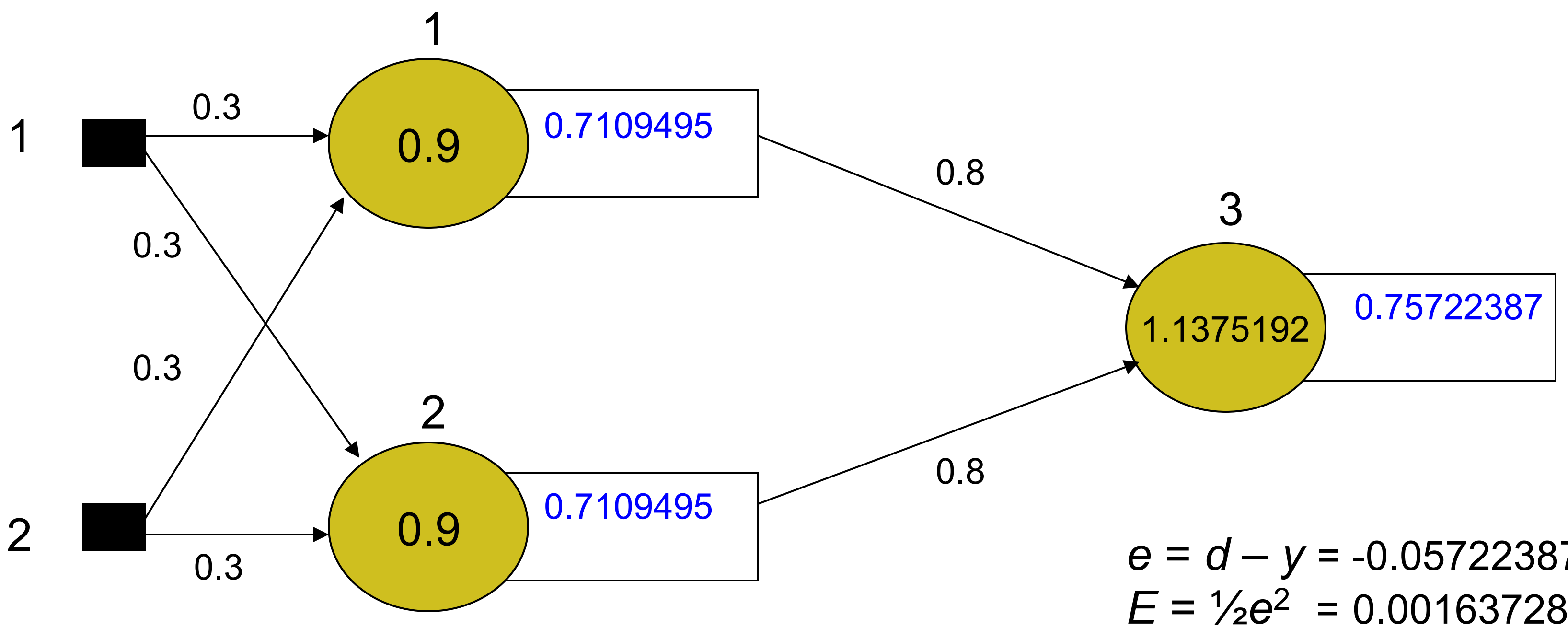
Topology of Net

Feed-forward Epoch



Topology of Net

Feed-forward Epoch



$e = d - y = -0.05722387$

$E = \frac{1}{2}e^2 = 0.001637286$

Calculate the Deltas for the Output Layer

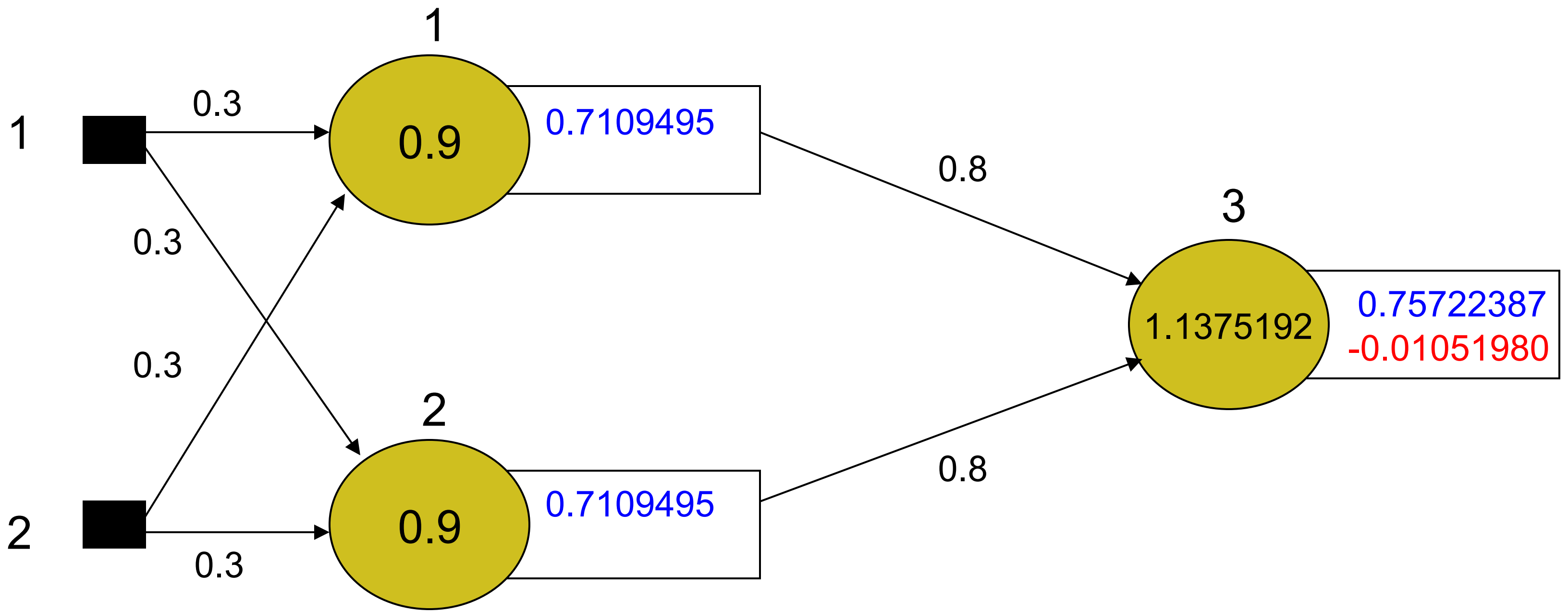
$$w_{jk}(t+1) = w_{jk}(t) - \eta\left(-e_k[1-y_k]y_k x_j\right)$$

$$w_{jk}(t+1) = w_{jk}(t) + \eta\left(\delta_k x_j\right)$$

$$\delta_k = e_k\left[1-y_k\right]y_k$$

Delta (node 3) = (0.7 - 0.757224) (1 - 0.757224) 0.757224
= -0.0105198

Back Propagation Epoch



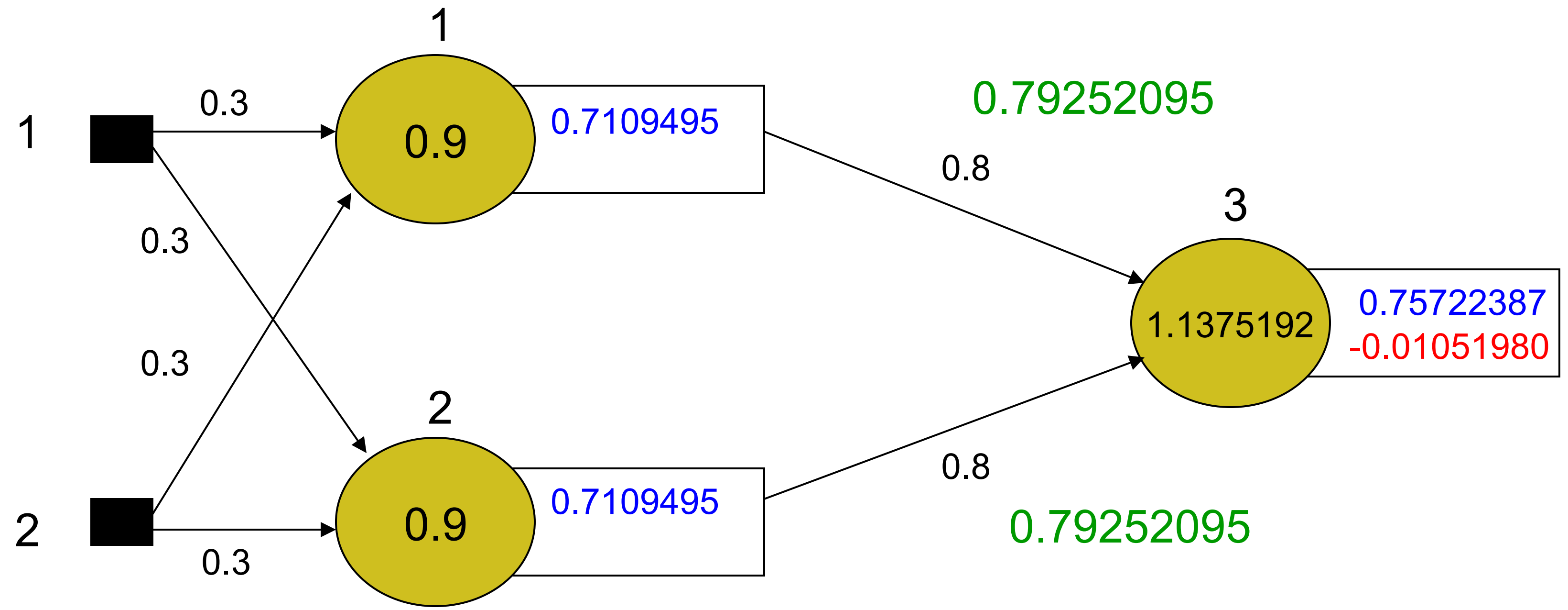
Calculate the New Weights for the Output Layer

$$w_{jk}(t+1) = w_{jk}(t) - \eta\left(-e_k[1-y_k]y_k x_j\right)$$

$$w_{jk}(t+1) = w_{jk}(t) + \eta\left(\delta_k x_j\right)$$

$w_{jk}\,(t+1)$ = (0.8) + 1*(-0.0105198)*0.7109495
= 0.79252095

Back Propagation Epoch



Calculate the Deltas for the Input Layer

Recall that for the input layer, the gradient vector element

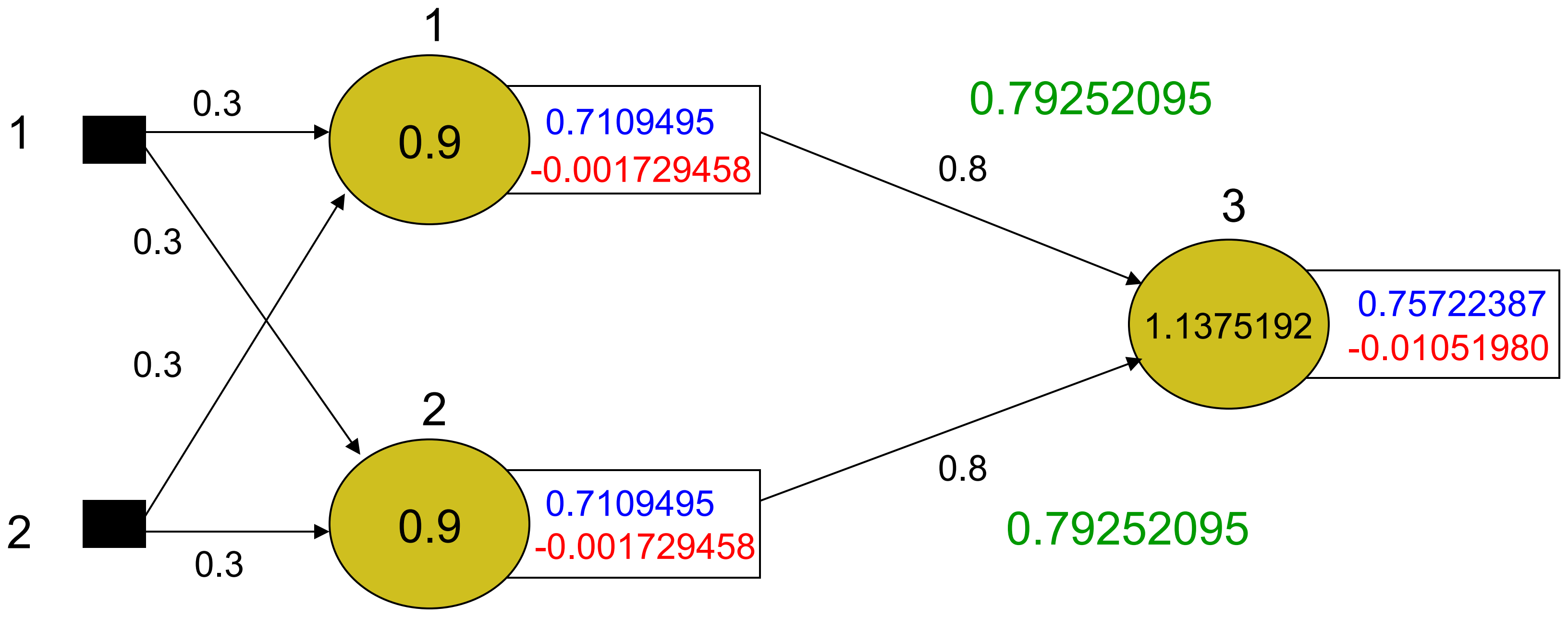
$$\frac{\partial E}{\partial w_{ij}} = -[1 - x_j]x_j\left(\sum_k \delta_k w_{jk}\right)x_i = \delta_j x_i$$

where

$$\delta_j = [1 - x_j]x_j\left(\sum_k \delta_k w_{jk}\right)$$

δ_j = (1 - 0.7109495) * 0.7109495 *(-0.01051980*0.8)
= -0.001729458

Back Propagation Epoch



Calculate the New Weights for the Hidden Layer

$$w_{ij}(t+1) = w_{ij}(t) - \eta\left(\frac{\partial E}{\partial w_{ij}}\right)$$

But now

$$\frac{\partial E}{\partial w_{ij}} = -[1 - x_j]x_j\left(\sum_k \delta_k w_{jk}\right)x_i = -\delta_j x_i$$

Thus, the updated weights for the input layer are $w_{ij}(t+1) = w_{ij}(t) + \eta\left(\delta_j x_i\right)$

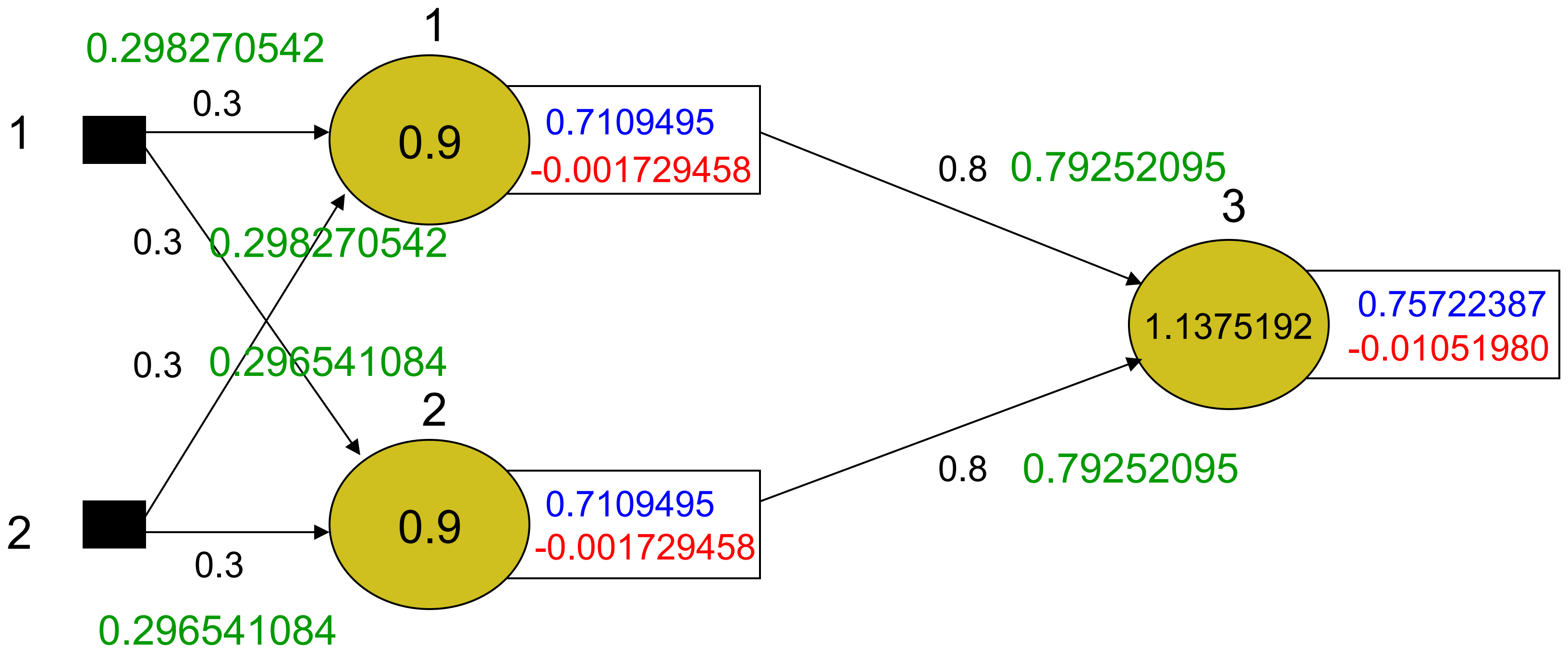
w_{11} $(t+1)$ = (0.3) + 1*(-0.001729458)*1 = 0.298270542

w_{12} $(t+1)$ = (0.3) + 1*(-0.001729458)*1 = 0.298270542

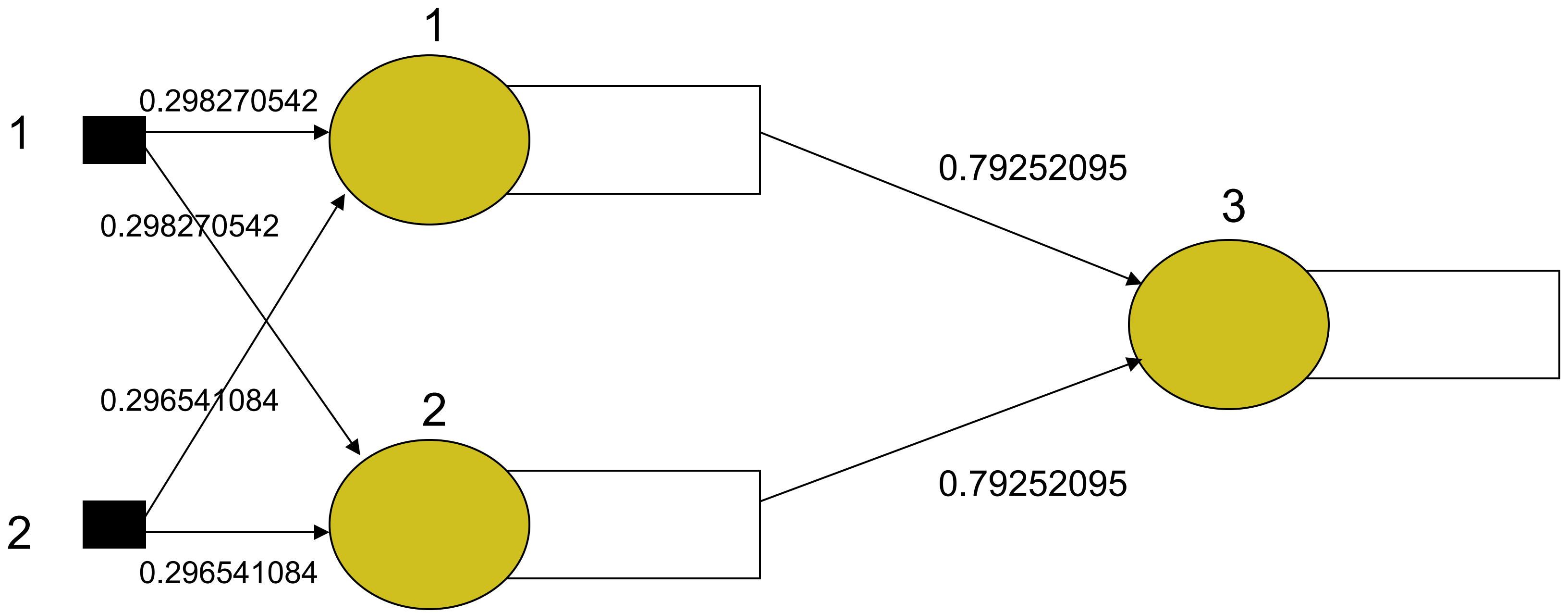
w_{21} $(t+1)$ = (0.3) + 1*(-0.001729458)*2 = 0.296541084

w_{22} $(t+1)$ = (0.3) + 1*(-0.001729458)*2 = 0.296541084

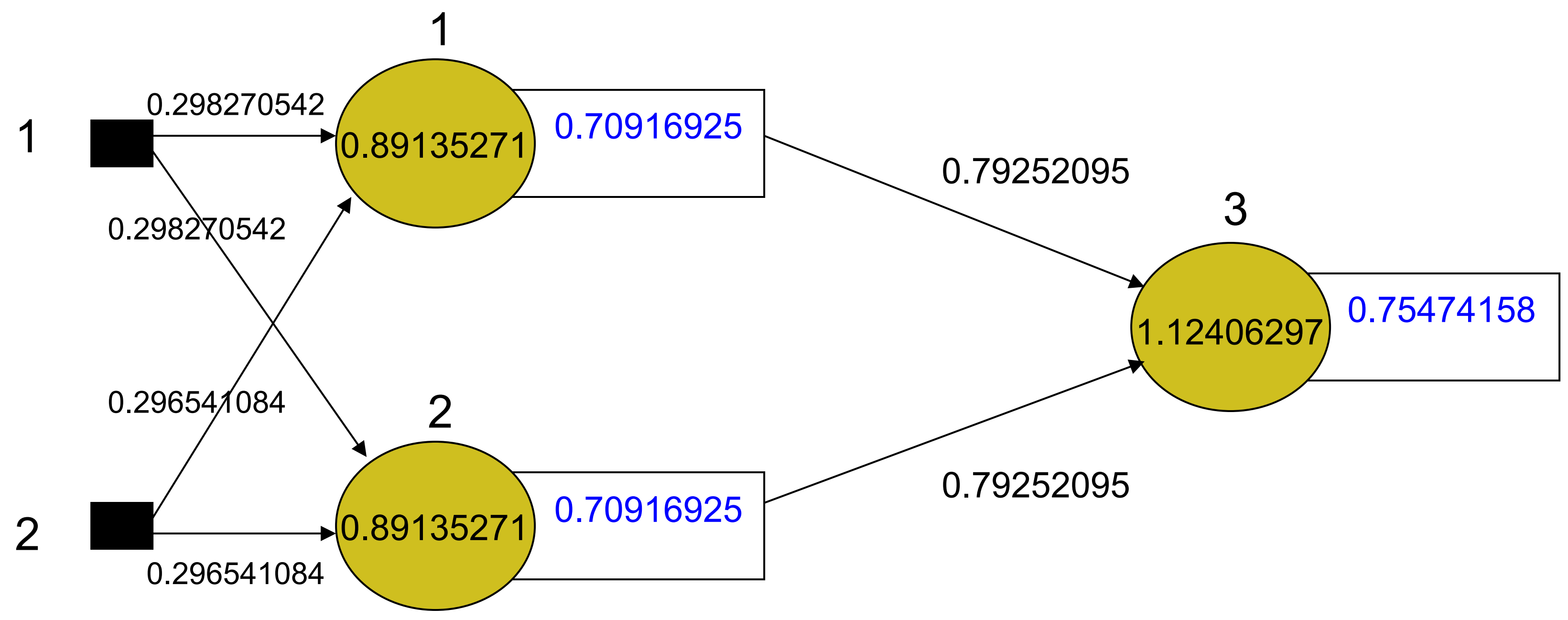
Back Propagation Epoch



Start of a New Day



2^{nd} Feed-forward Epoch



First Epoch	Second Epoch
$e = d - y = -0.05722387$	$e = d - y = -0.05474158$
$E = \frac{1}{2}e^2 = 0.001637286$	$E = \frac{1}{2}e^2 = 0.00149832$

The FFBP Summary

Feed-forward Epoch:

- We presented input values at the input layer to the hidden layer nodes.
- Computed activity and activation values in the hidden layer.
- Used those activation function values as input values for the output layer node(s).
- Computed the output error values.

Back-Propagation Epoch:

- Used the output error values in conjunction with the inputs to and the outputs of the output layer node(s) to compute the delta value(s) (gradient value(s)) for the output layer node(s).
- Used these delta value(s) to compute updated weights for the output layer node(s).
- Used these delta values in conjunction with the output and input values of the hidden layer nodes along with the original weights in the output layer to compute delta values for the hidden layer nodes.
- Used the delta values associated with the hidden layer nodes to compute updated weights for the hidden layer.