ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

Ms. Swetha R.
Department of Electronics and Communication Engineering
PES University

CONTENT:Part-1

1. *Perceptron*
    1. *Introduction-Linearly Separable*
    2. *Rosenblatt Algorithm with example*
    3. *Perceptron Convergence Theorem*
2. *Single Layer Perceptron*

    DrawBack: Xor Logic Gate
3. *Multilayer Perceptron*
    1. *Backpropogation Algorithm*
    2. *Example: XOR Logic Gate*

Back-Propogation Algorithm (BPA)

Backward Pass:

Total instantaneous energy in the error is

$$E = \frac{1}{2}\sum_{l=1}^{m_2} e_l^2(k)$$

But in our case m_2 is 1

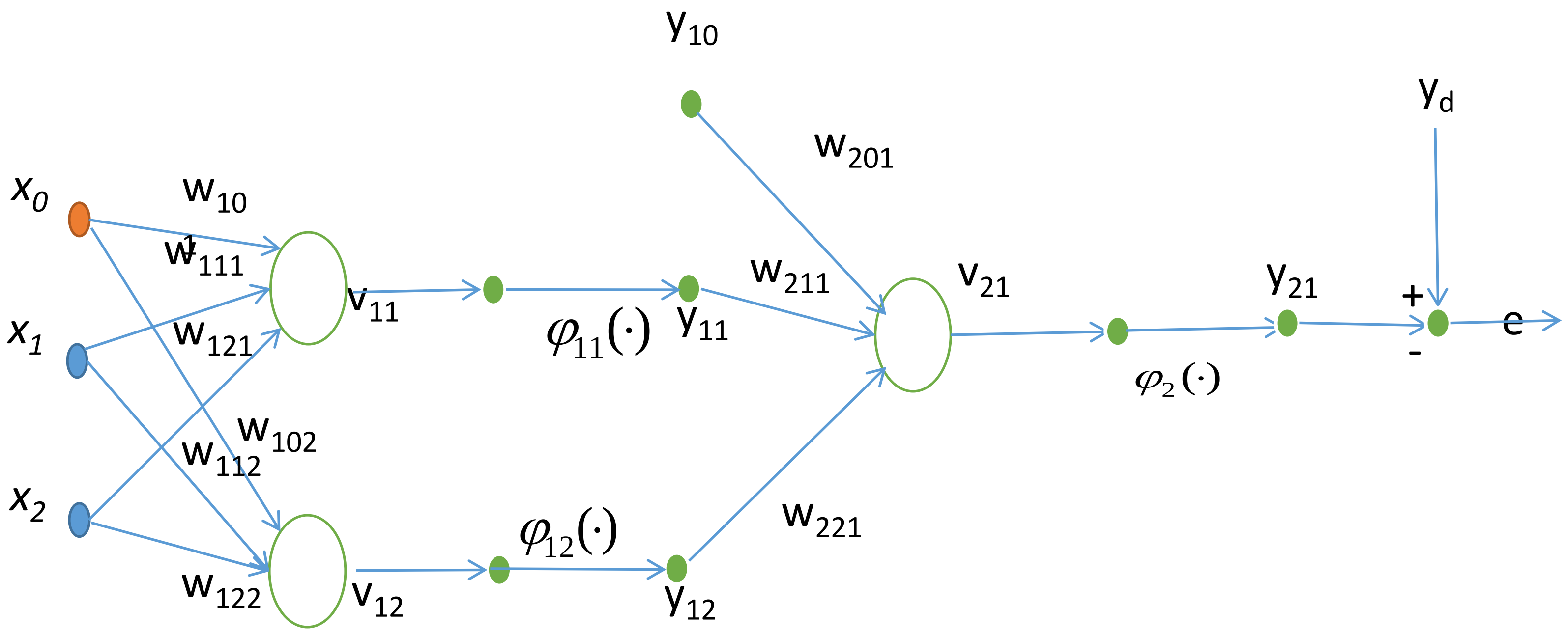
$$E = \frac{1}{2} e_1^2(k)$$

This error is now propogates backward as follows to update synaptic weights of all layers

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

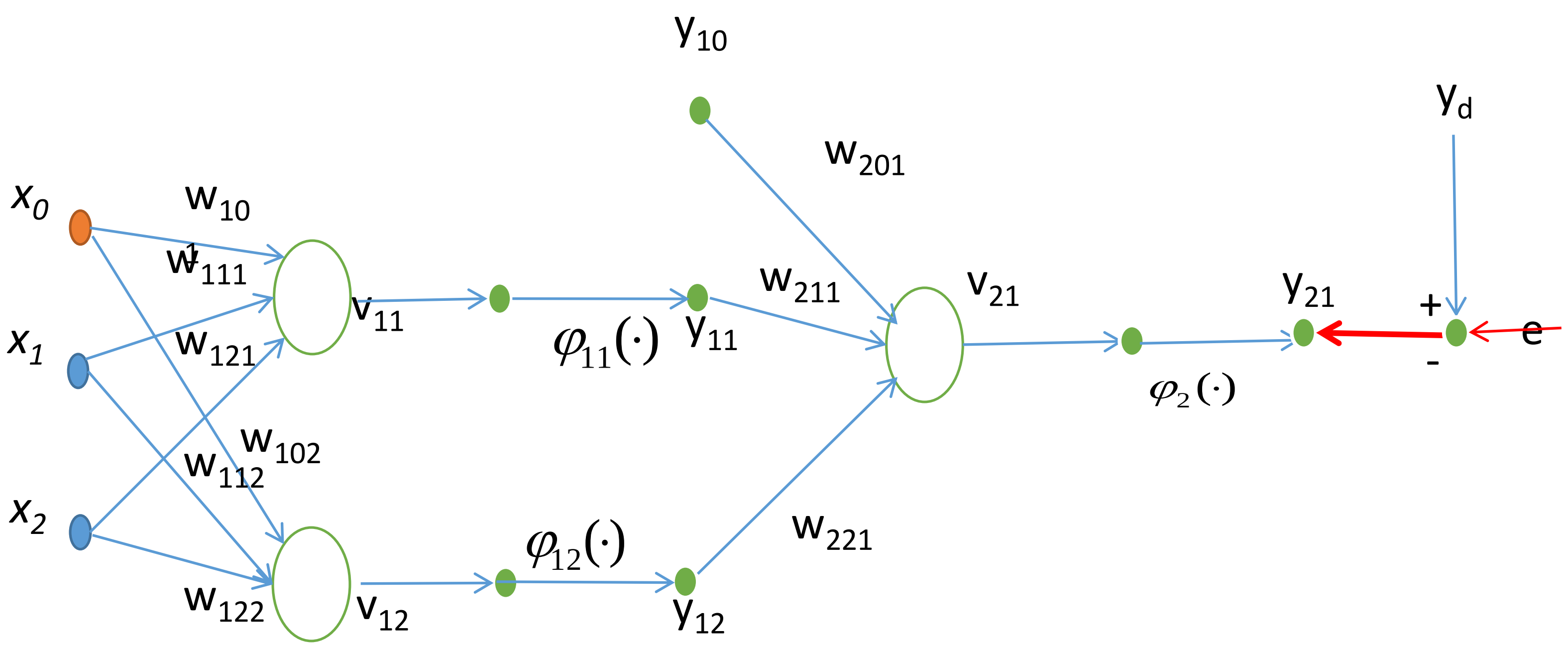


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

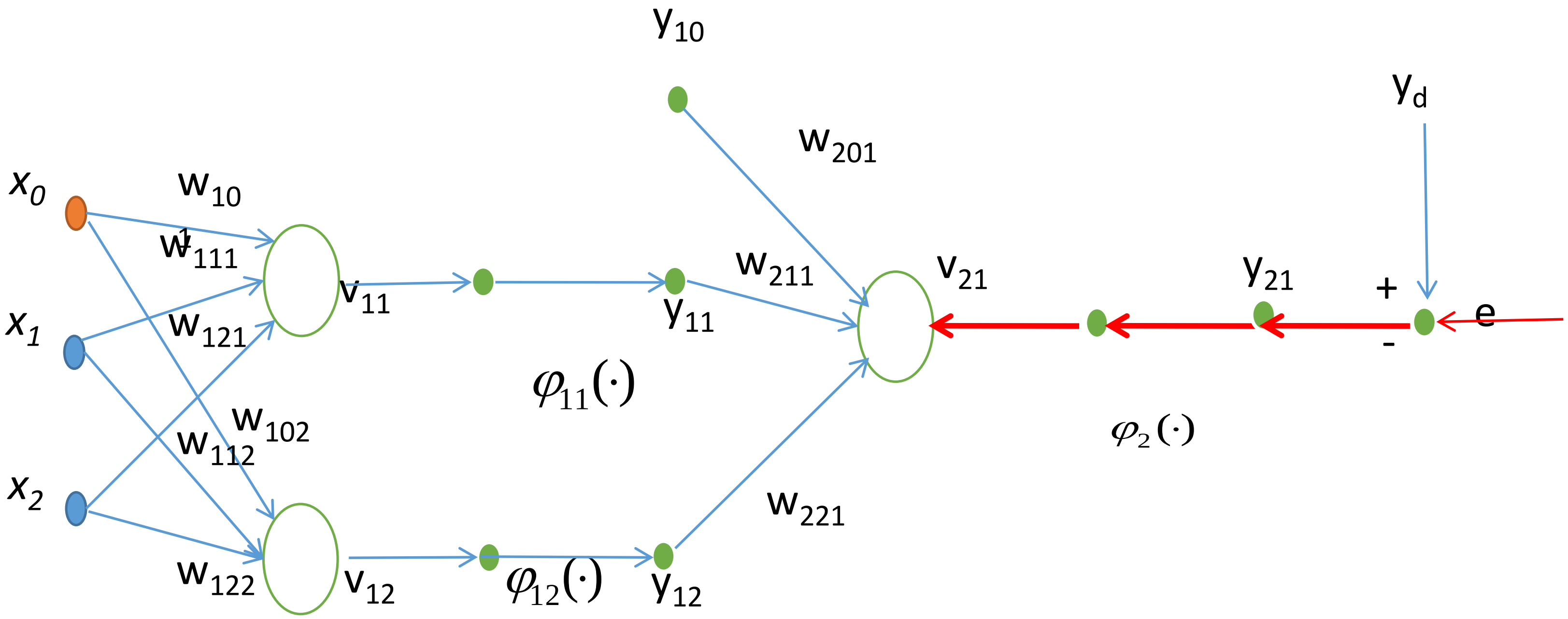


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Back-Propogation Algorithm (BPA)

Backward Pass:

Delta Rule:

$$w_{plj}(k+1) = w_{plj}(k) + \Delta w_{plj}(k)$$

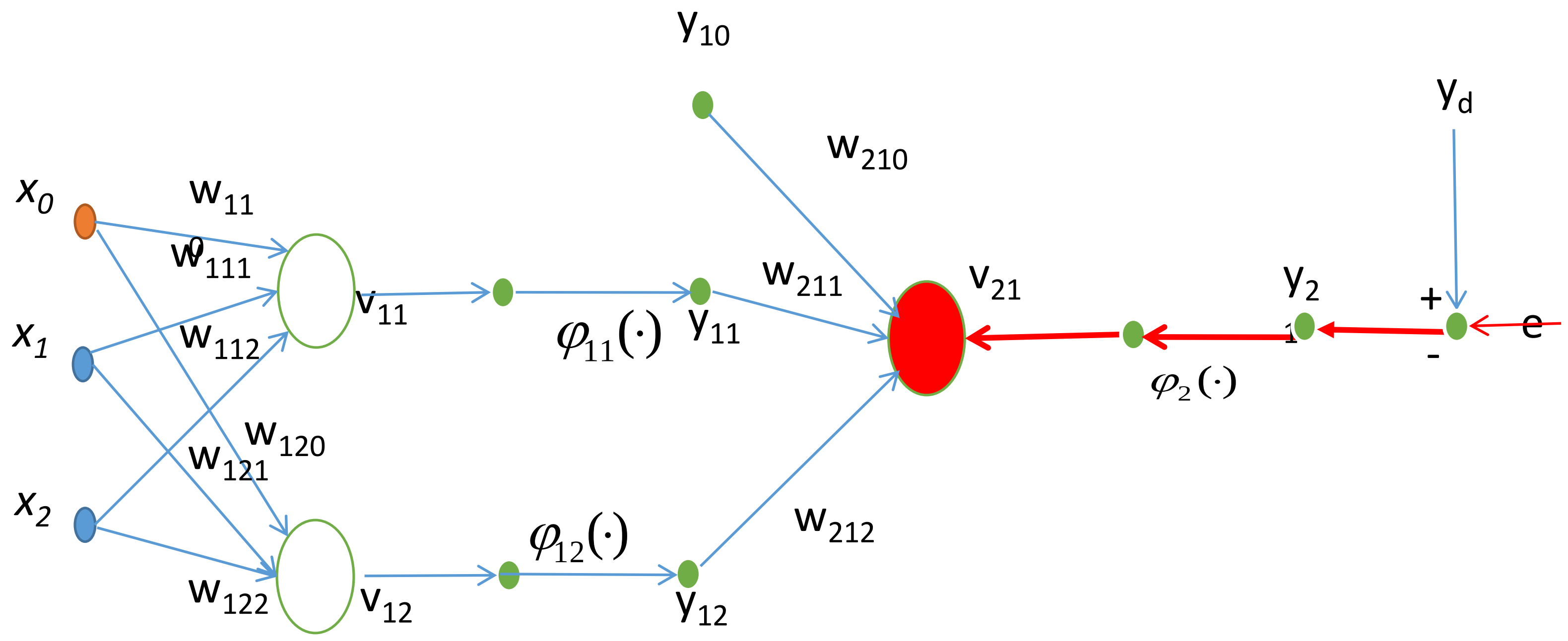
$$\Delta w_{plj}(k) = -\eta \frac{\partial E}{\partial w_{plj}(k)}$$

Layer 2: Output Layer

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

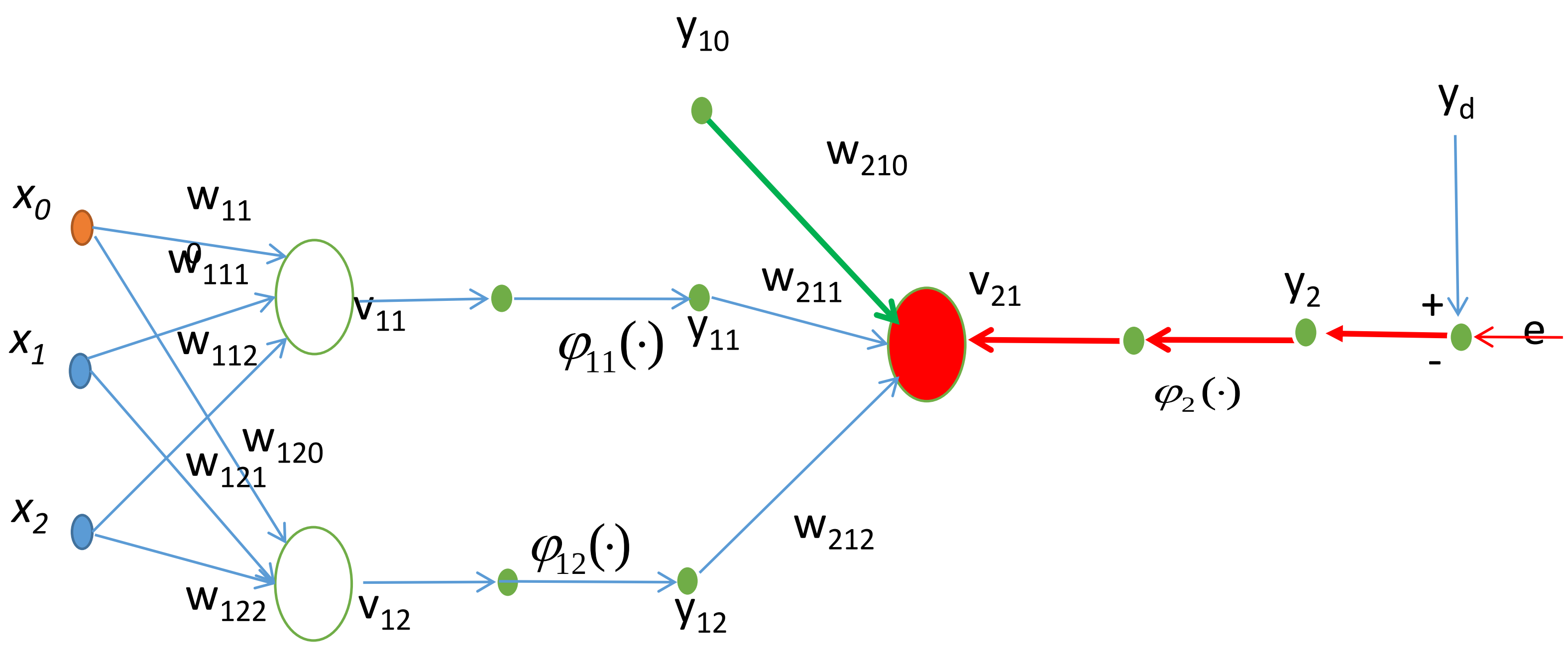


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

Delta Rule:

$$w_{plj}(k+1) = w_{plj}(k) + \Delta w_{plj}(k)$$

$$\Delta w_{plj}(k) = -\eta \frac{\partial E}{\partial w_{plj}(k)}$$

Layer 2: Output Layer

$$\frac{\partial E}{\partial w_{210}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{210}}$$

We know that

$$E = \frac{1}{2} e_1^2(k)$$

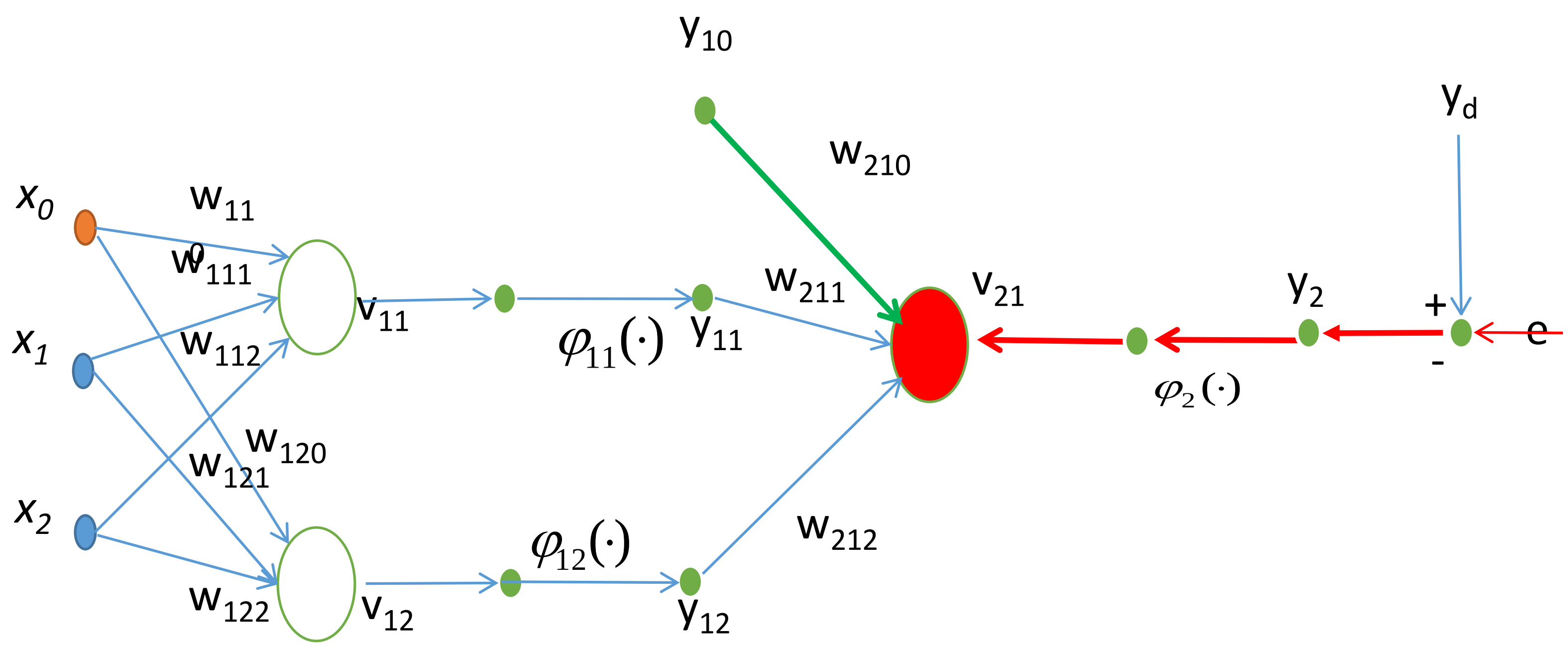
$$e_1 = y_d - y_{21}$$

$$\frac{\partial E}{\partial w_{210}} = e_1 \cdot (-1) \cdot \varphi_2{}'(v_2(k)) \cdot y_{10}$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



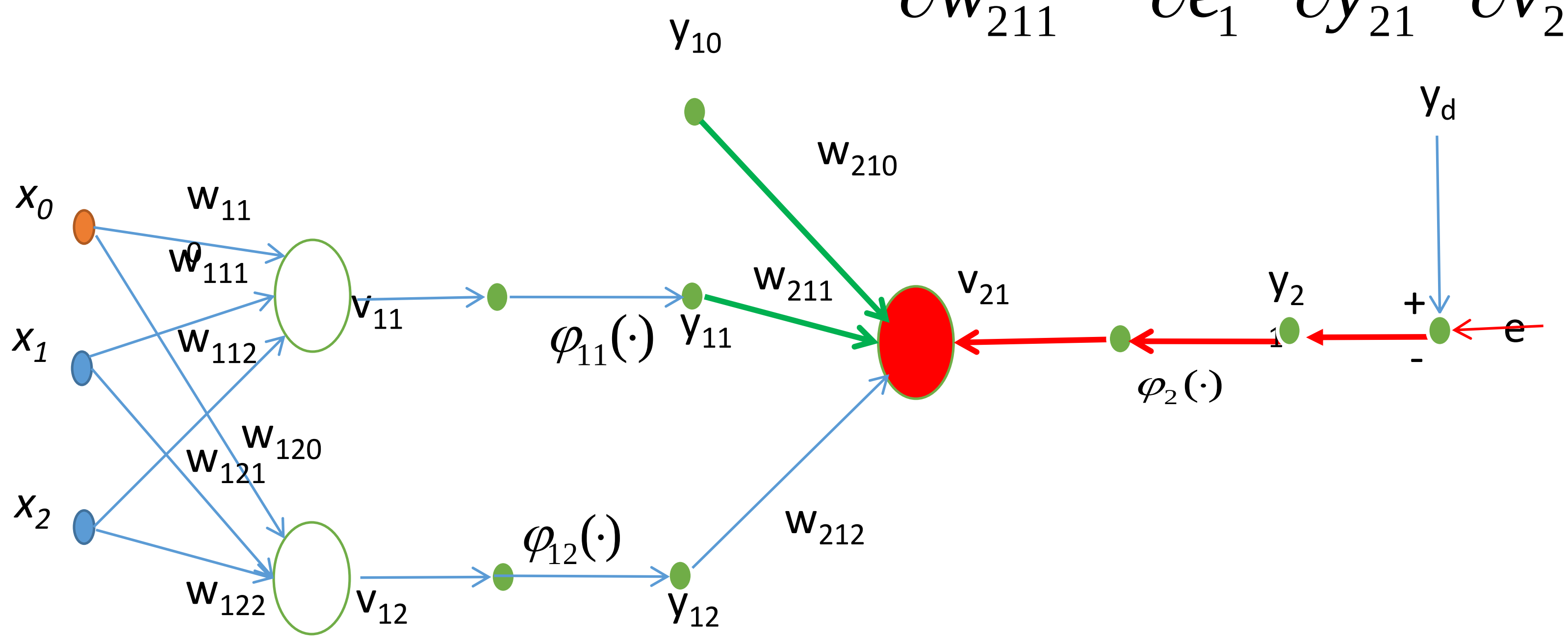
Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

$$\frac{\partial E}{\partial w_{211}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{211}}$$

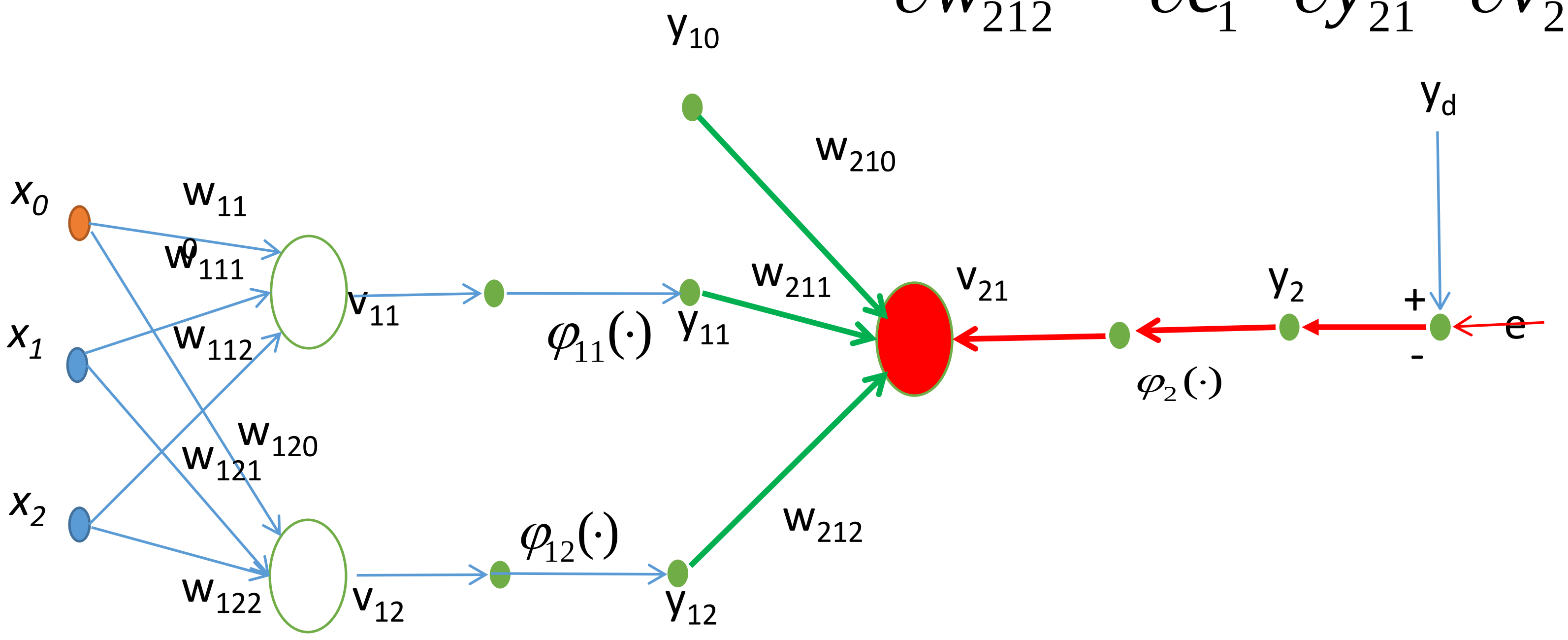


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Back-Propogation Algorithm (BPA)

Backward Pass:

$$\frac{\partial E}{\partial w_{212}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{212}}$$



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Back-Propogation Algorithm (BPA)

Backward Pass:

In general: $$\frac{\partial E}{\partial w_{2lj}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{2l}} \cdot \frac{\partial y_{2l}}{\partial v_{2l}} \cdot \frac{\partial v_{2l}}{\partial w_{2lj}}$$

Now Define, Local gradient of output layer as

$$\delta_2(k) = -\frac{\partial E}{\partial v_{2l}}$$

$$\delta_{2l}(k) = (e_1 \cdot \varphi_2'(v_2(k))) = \delta_2(k)$$

Back-Propogation Algorithm (BPA)

Backward Pass:

$$\Delta W_2 = \begin{pmatrix} \Delta w_{210} & \Delta w_{211} & \Delta w_{211} \end{pmatrix}$$

$$= \eta_2 \begin{pmatrix} \delta_{21}(k) y_{20}(k) & \delta_{21}(k) y_{21}(k) & \delta_{21}(k) y_{22}(k) \end{pmatrix}$$

$$= \eta_2 \delta_2(k) y_1^T(k)$$

Therefore,

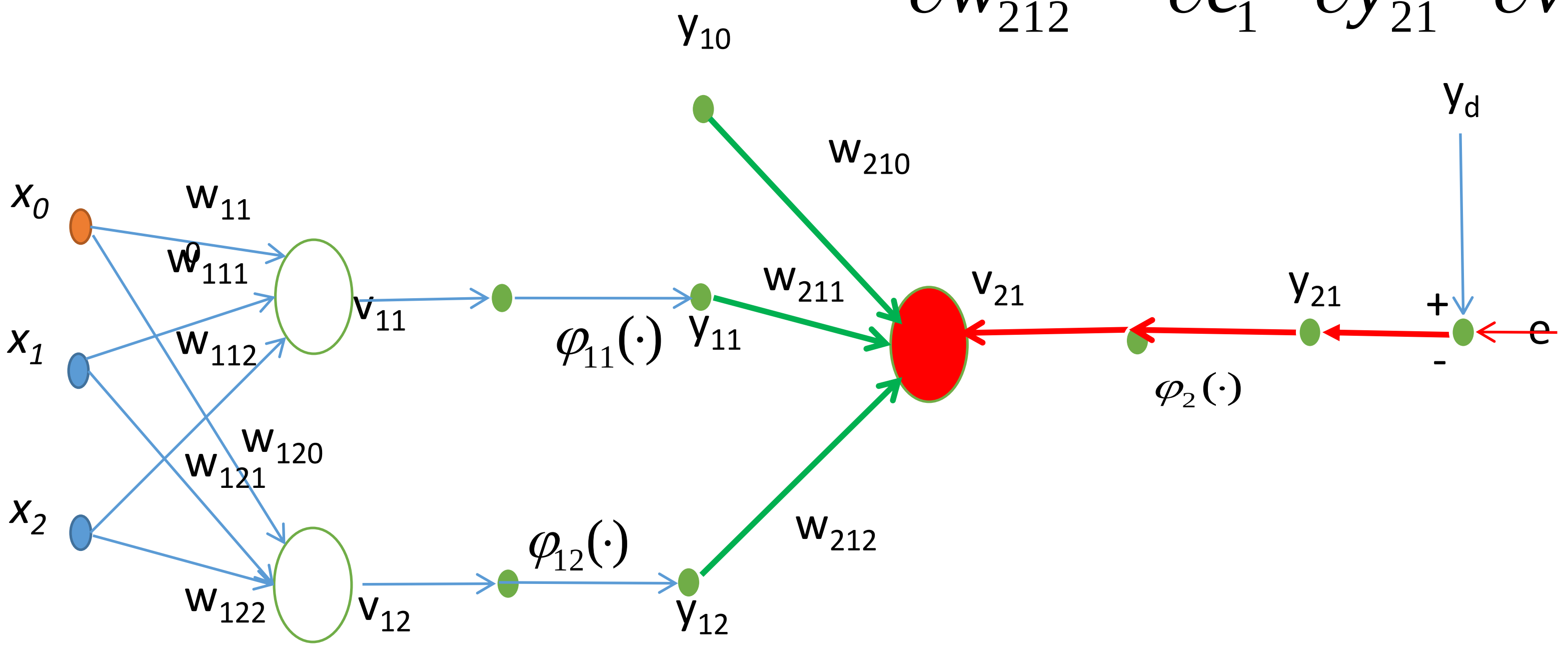
$$W_2(k+1) = W_2(k) + \Delta W_2(k)$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

$$\frac{\partial E}{\partial w_{212}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial w_{212}}$$

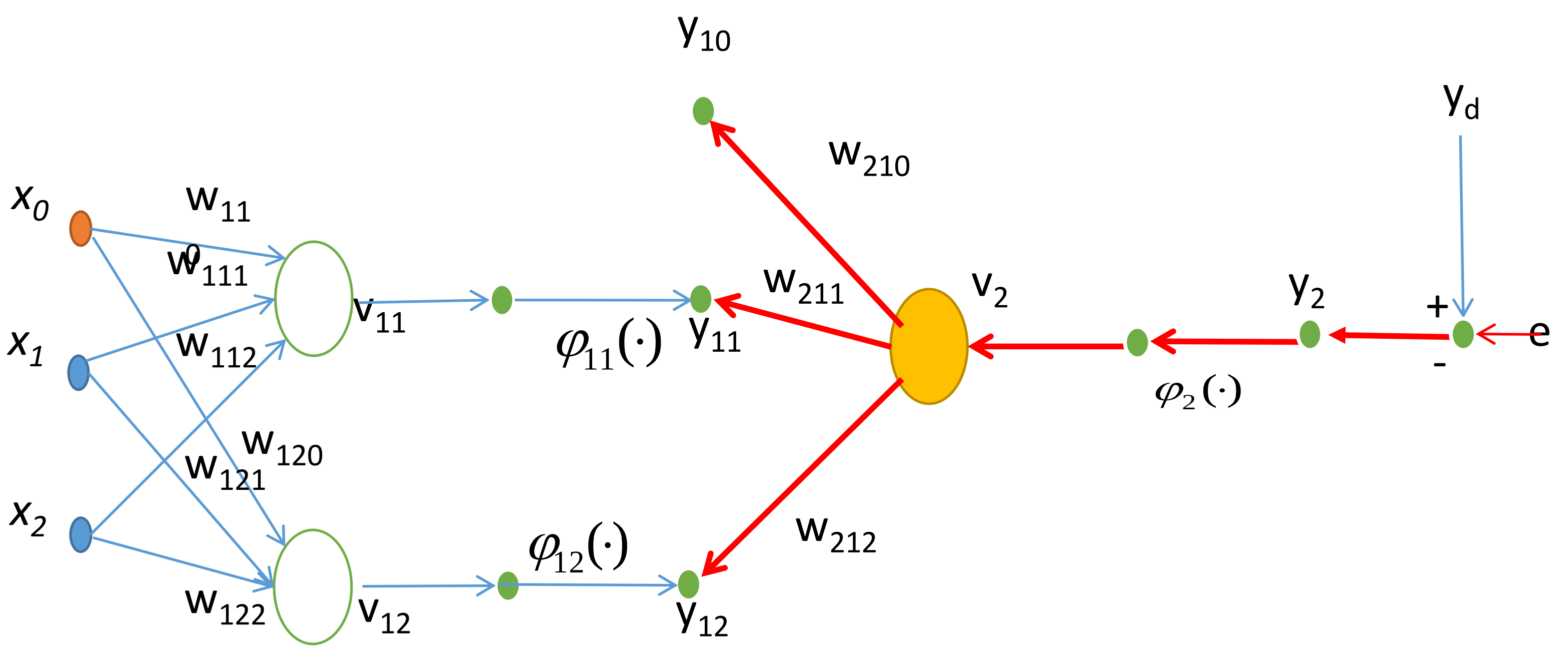


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

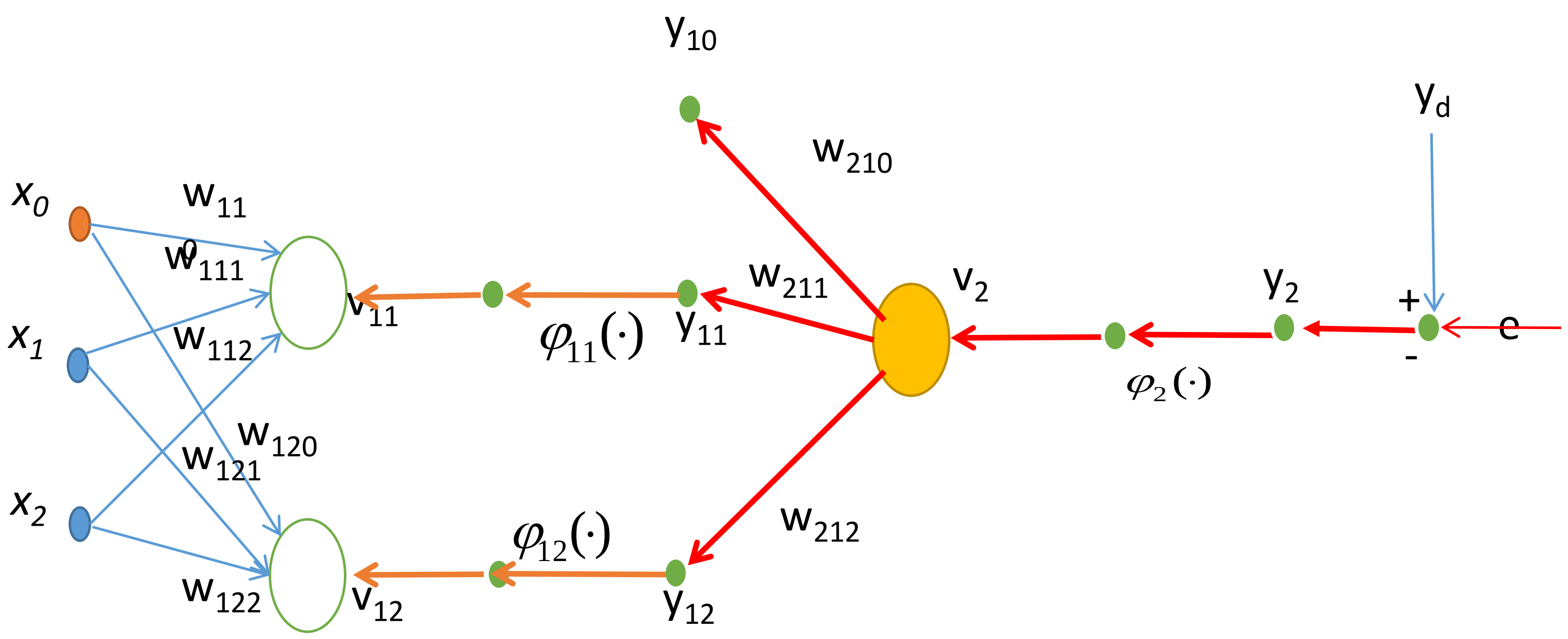


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

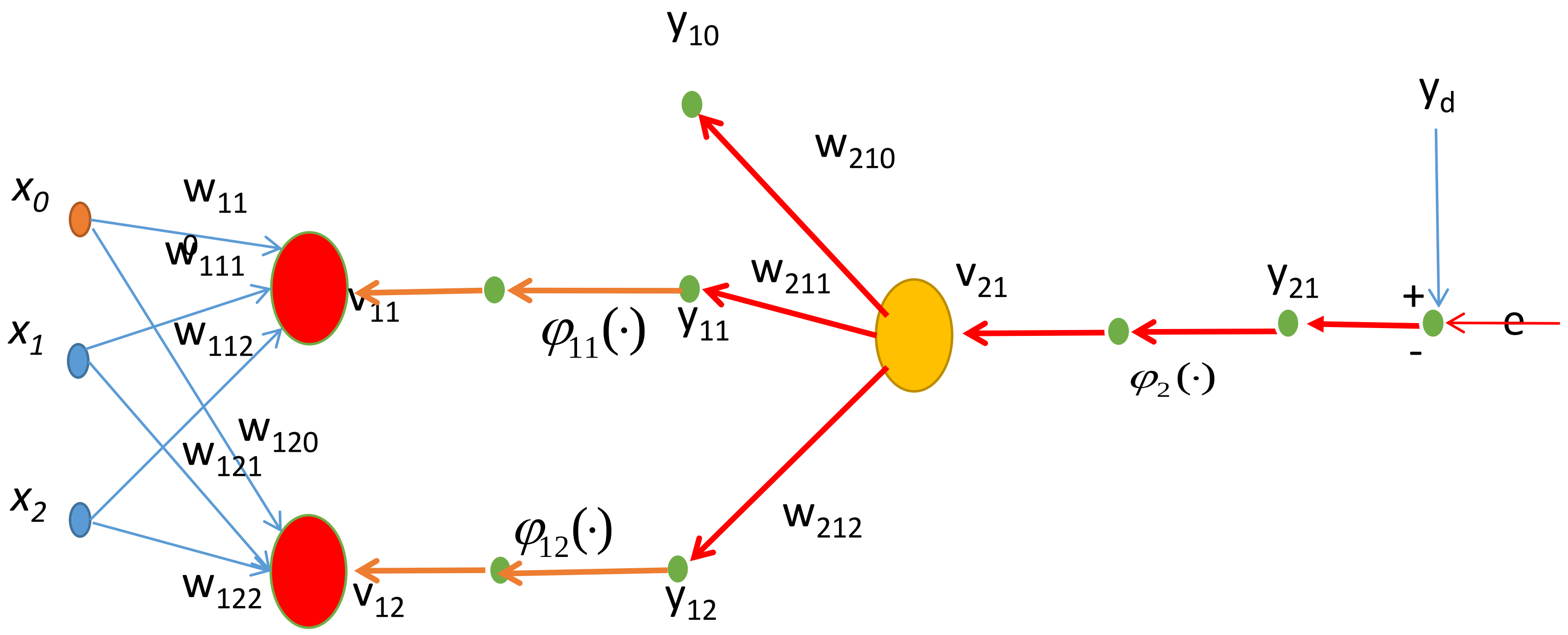
Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Backward Pass:

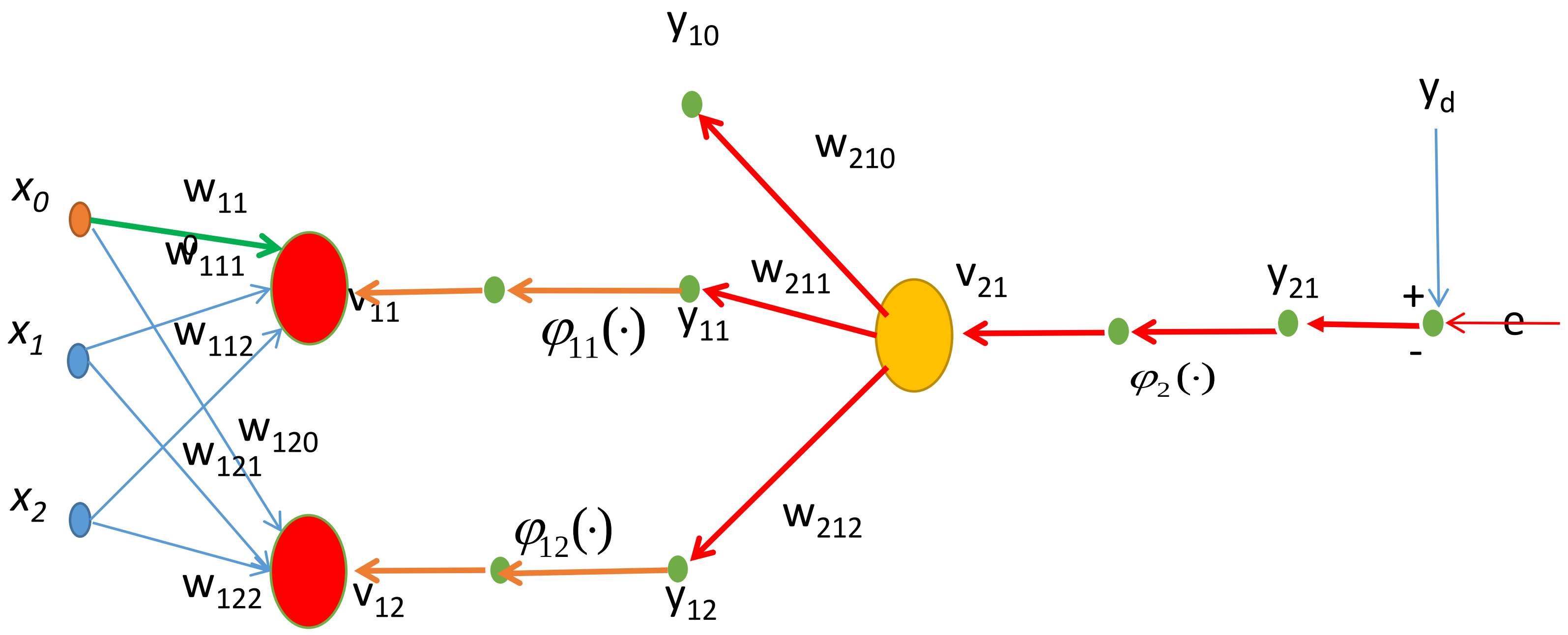


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Back-Propogation Algorithm (BPA)

Layer 1: 1st Hidden layer

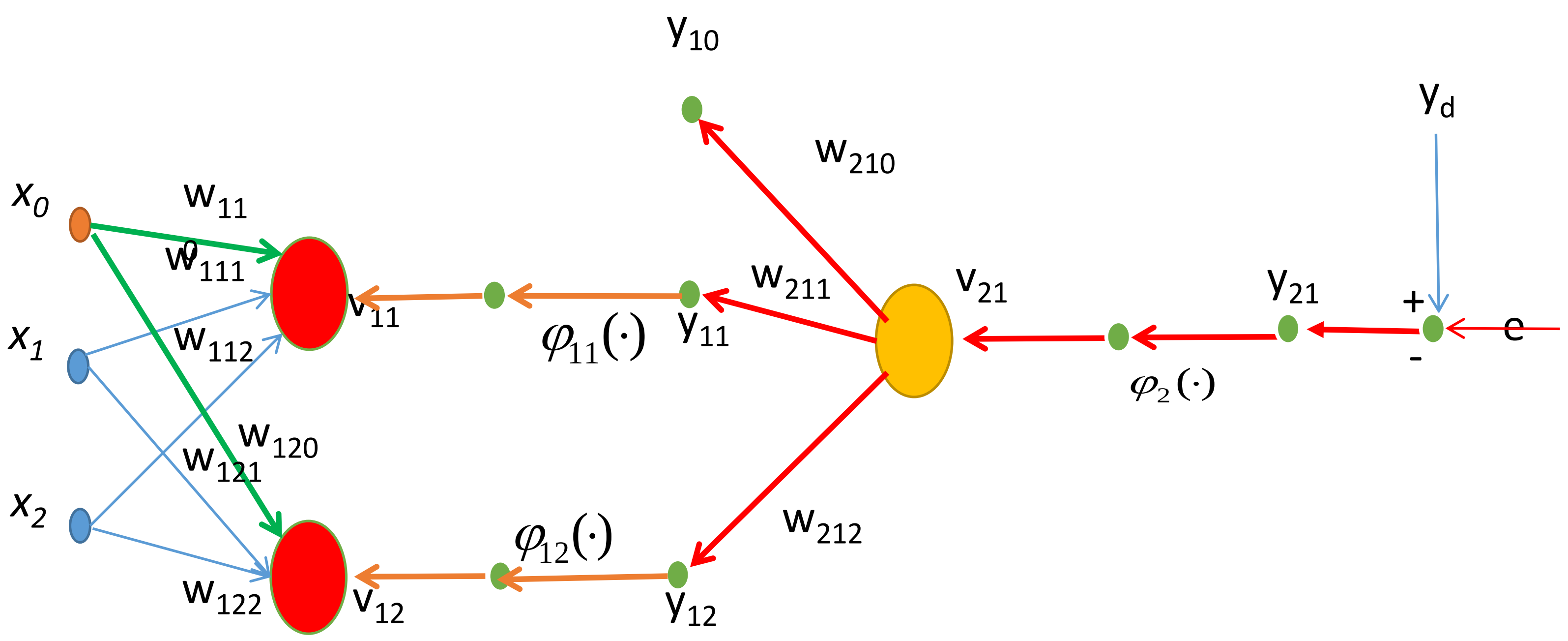
$$\frac{\partial E}{\partial w_{110}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial v_{11}} \cdot \frac{\partial v_{11}}{\partial w_{110}}$$

$$= e_1 \cdot (-1) \cdot \varphi_2^{'}(v_2(k)) \cdot w_{211} \cdot \varphi_{11}^{'}(v_{11}(k)) \cdot y_{00}(k) \qquad \text{................(a)}$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Layer 1: 1st Hidden layer

$$\frac{\partial E}{\partial w_{110}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial v_{11}} \cdot \frac{\partial v_{11}}{\partial w_{110}}$$

$$= e_1 \cdot (-1) \cdot \varphi_2^{'}(v_2(k)) \cdot w_{211} \cdot \varphi_{11}^{'}(v_{11}(k)) \cdot y_{00}(k) \qquad \text{................(a)}$$

$$\frac{\partial E}{\partial w_{120}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{12}} \cdot \frac{\partial y_{12}}{\partial v_{12}} \cdot \frac{\partial v_{12}}{\partial w_{120}}$$

$$= e_1 \cdot (-1) \cdot \varphi_2^{'}(v_2(k)) \cdot w_{211} \cdot \varphi_{12}^{'}(v_{12}(k)) \cdot y_{00}(k)$$

................(b)

Layer 1: 1st Hidden layer

- Comparing equation (a) and (b), we get the generalised equation for the 1st hidden layer as follows

$$\frac{\partial E}{\partial w_{1ji}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{1j}} \cdot \frac{\partial y_{1j}}{\partial v_{1j}} \cdot \frac{\partial v_{1j}}{\partial w_{1ji}}$$

Now, lets define the local gradient for the nurons in the first hidden layer

$$\delta_{2j}(k) = -\frac{\partial E}{\partial v_{2j}}$$

$$\delta_1(k) = \begin{pmatrix} \delta_{11}(k) \\ \delta_{12}(k) \end{pmatrix}$$

$$\frac{\partial E}{\partial w_{1ji}} = \underbrace{\underbrace{\frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}}}_{\delta_2(k)} \cdot \frac{\partial v_{21}}{\partial y_{1j}} \cdot \frac{\partial y_{1j}}{\partial v_{1j}}}_{\delta_1(k)} \cdot \frac{\partial v_{1j}}{\partial w_{1ji}}$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

$$\delta_1(k) = \begin{pmatrix} \delta_{11}(k) \\ \delta_{12}(k) \end{pmatrix}$$

$$\delta_{11}(k) = \delta_2(k) w_{211} \varphi_{11}(v_{11}(k))$$

$$\delta_{12}(k) = \delta_2(k) w_{212} \varphi_{12}(v_{12}(k))$$

$$\delta_1(k) = \begin{pmatrix} \delta_{11}(k) \\ \delta_{12}(k) \end{pmatrix} = \begin{pmatrix} \delta_2(k) w_{211} \\ \delta_2(k) w_{212} \end{pmatrix} \Theta \begin{pmatrix} \varphi'(v_{11}(k)) \\ \varphi'(v_{12}(k)) \end{pmatrix}$$

$$= \overline{W}_2^T(k) \delta_2(k) \Theta \varphi_2'(k)$$

where, $$\varphi_1(v_1(k)) = \begin{pmatrix} \varphi_{11}(v_{11}(k)) \\ \varphi_{12}(v_{12}(k)) \end{pmatrix}$$

$$v_1(k) = \begin{pmatrix} v_{11}(k) \\ v_{12}(k) \end{pmatrix} \qquad \overline{W}_2(k) = \begin{pmatrix} W_{211} & W_{212} \end{pmatrix}$$

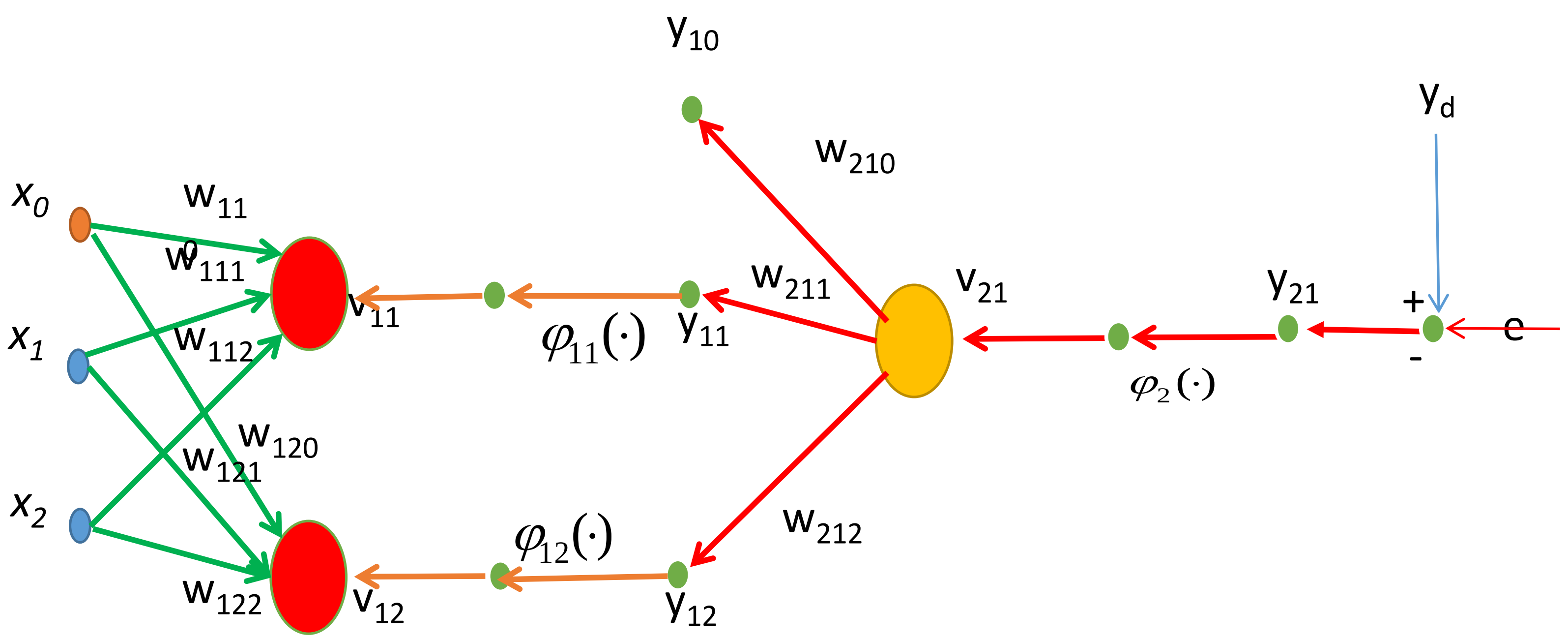
Therefore,

$$\frac{\partial E}{\partial w_{1ji}} = \delta_1(k) \cdot y_0^T(k)$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Back-Propogation Algorithm (BPA)

$$w_1(k+1) = w_1(k) + \Delta w_1(k)$$

$$\Delta w_1(k) = \begin{pmatrix} \Delta w_1(k) & \Delta w_1(k) & \Delta w_1(k) \\ \Delta w_1(k) & \Delta w_1(k) & \Delta w_1(k) \end{pmatrix}$$

$$\Delta w_{1ji} = -\eta_1 \frac{\partial E}{\partial w_{1ji}}$$

$$\Delta w_{1ji} = \eta_1 \delta_1(k) \cdot y_0^T(k)$$

$$\Delta w_1(k) = \eta_1 \begin{pmatrix} \delta_{11}(k) y_{00}(k) & \delta_{11}(k) y_{01}(k) & \delta_{11}(k) y_{02}(k) \\ \delta_{12}(k) y_{00}(k) & \delta_{12}(k) y_{01}(k) & \delta_{12}(k) y_{02}(k) \end{pmatrix}$$

$$= \eta_1 \begin{pmatrix} \delta_{11}(k) \\ \delta_{12}(k) \end{pmatrix} \Theta \begin{pmatrix} y_{00}(k) & y_{01}(k) & y_{02}(k) \end{pmatrix}$$

Hadamard Product

Summary:

Forward Computation

The induced local field for neuron j in layer l is

$$v_{lj}(k) = \sum_{i=0}^{m} w_{lji}(k) y_{(l-1)i}(k)$$

The ouput of neuron j in layer l is

$$y_{lj}(k) = \varphi_j(v_j(k))$$

If neuron j is in the input layer, set

$$y_{0j}(k) = x_j(k)$$

If neuron j is in the output layer, set

$$y_{lj}(k) = actual\_output$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Compute the error signal

$$e_j(k) = (desired\_output - actual\_output)$$

Bacward Computation

Compute the local gradients of the network defined by

$$\delta_{lj}(k) = \begin{cases} e_{lj}(k)\varphi_j^{'}(v_{lj}(k)) & for\_neuron\_j\_in\_output\_layer\_L \\ \varphi_j^{'}(v_{lj}(k))\sum_k \delta_{(l+1)nj}(n) & for\_neuron\_j\_in\_hiddenlayer\_l \end{cases}$$

Update weights:

$$w_{lji}(k+1) = w_{lji}(k) + \eta\delta_{lj}(k)y_{(l-1)i}(k)$$

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753