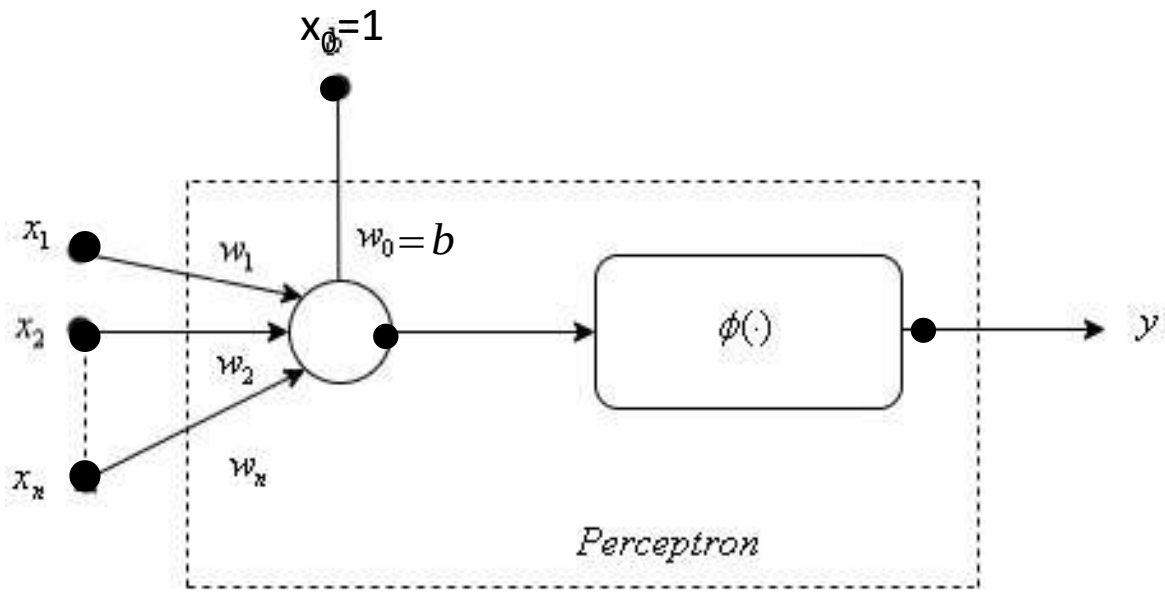
ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

Perceptron

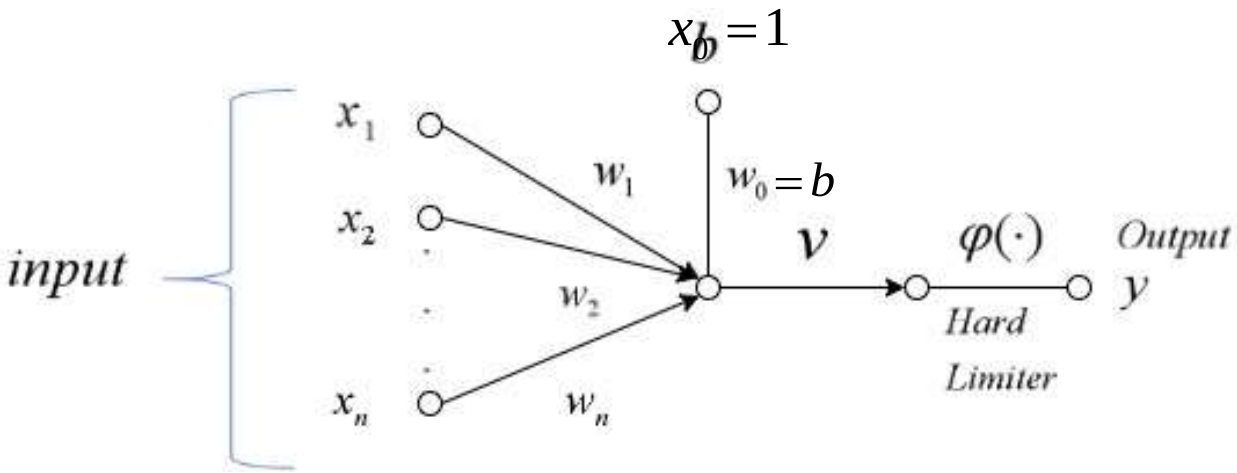
- **Perceptron:** It is a non-linear neuron model.
- The graphical representation of the perceptron is shown below:



Artificial Neural Network

Perceptron

- The signal flow graph of the perceptron is shown below:



Perceptron

- The summing node of the neuron model computes the linear combination of the inputs applied to its synspses and also incorporates an externally applies bias
- The resulting sum is applied to a hard limiter
- Accordingly neuron produces output

Perceptron

- From the model, the induced local field/hard limit input is

$$v = \sum_{i=1}^{n} w_i x_i + b$$

- The hardlimit function is defined as follows

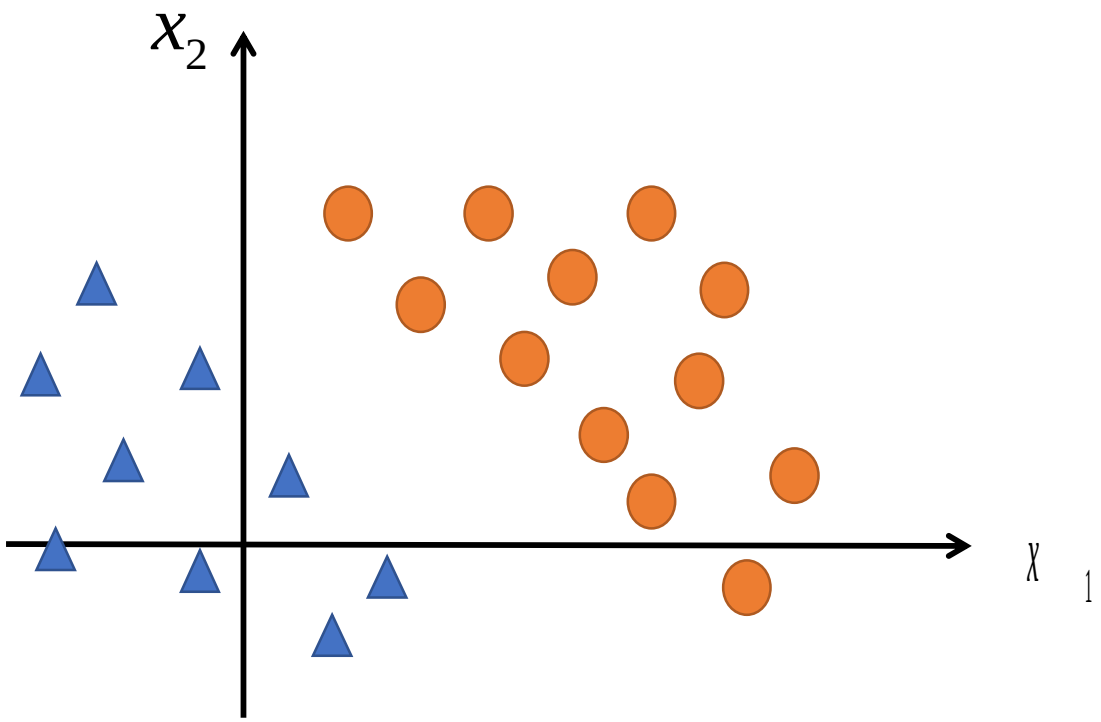
$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

Perceptron

- When do we use this perceptron ?
- Let us consider, we have 2 sets which is represented as in 2D- Plane as shown in the below figure

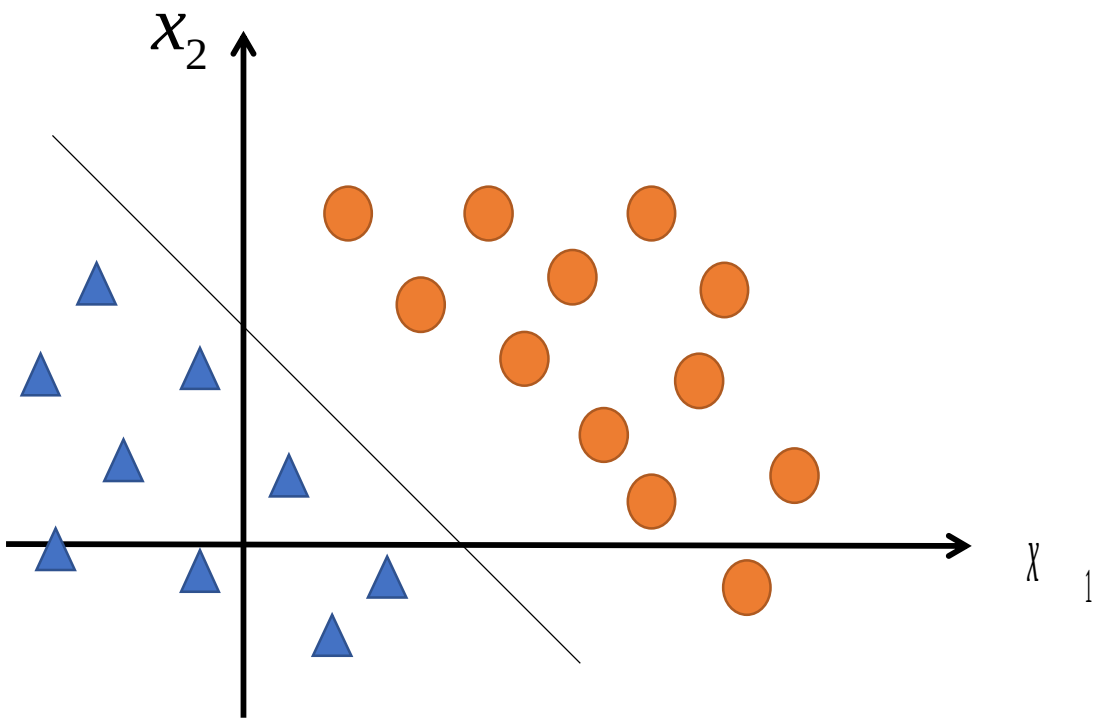
Perceptron

- When do we use this perceptron ?
- Let us consider, we have 2 sets which is represented as in 2D- Plane as shown in the below figure



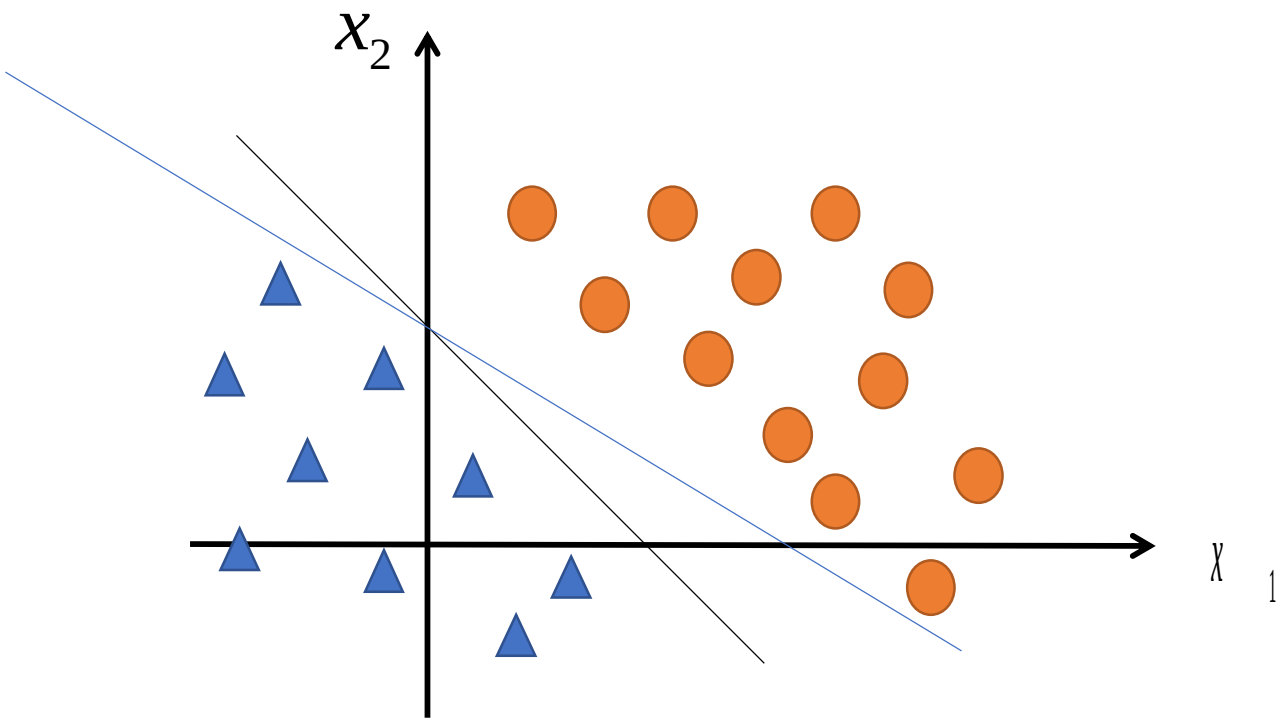
Perceptron

- When do we use this perceptron ?
- Let us consider, we have 2 sets which is represented as in 2D- Plane as shown in the below figure



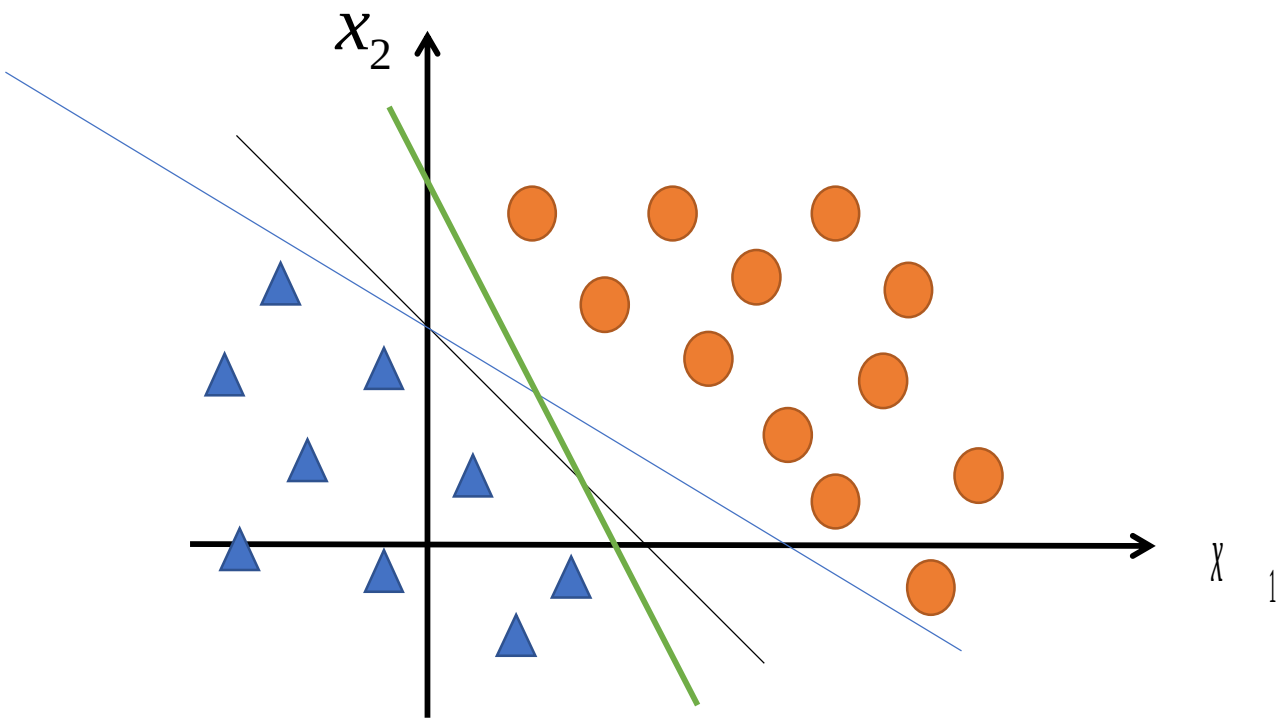
Perceptron

- When do we use this perceptron ?
- Let us consider, we have 2 sets which is represented as in 2D- Plane as shown in the below figure



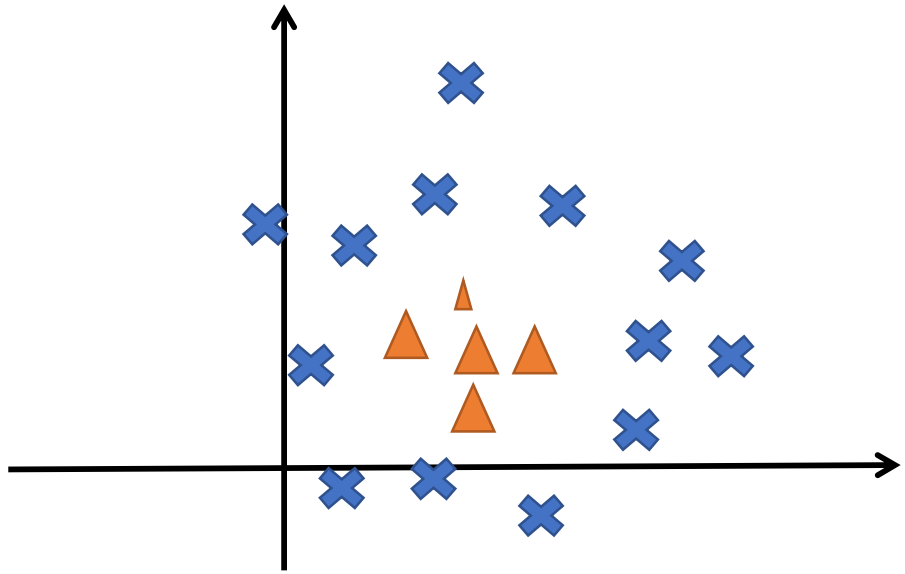
Perceptron

- When do we use this perceptron ?
- Let us consider, we have 2 sets which is represented as in 2D- Plane as shown in the below figure

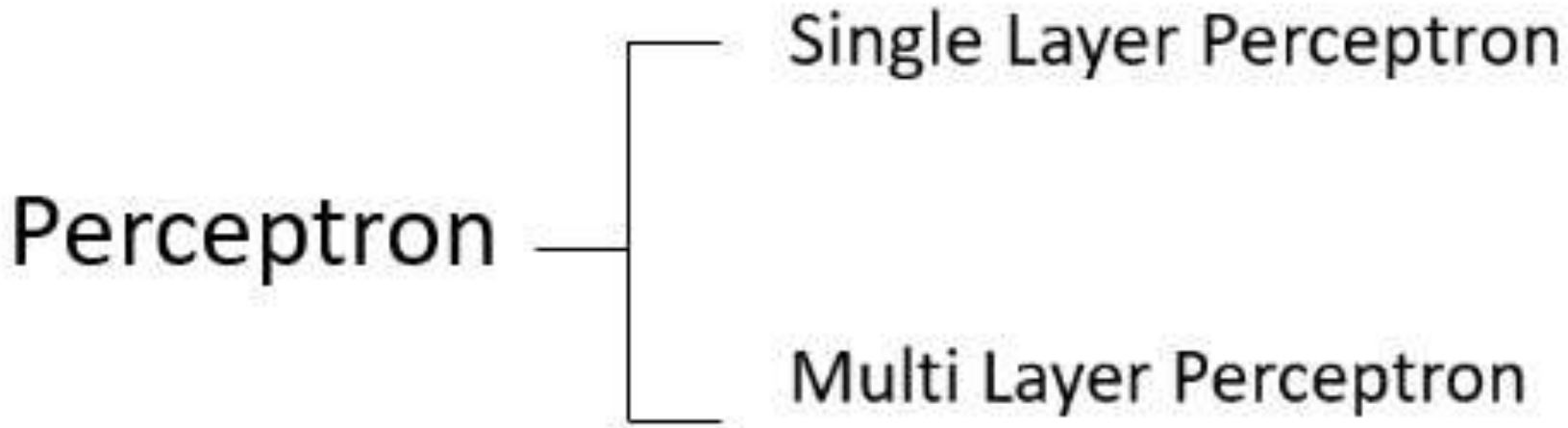


- Therefore, if any 2 sets can be separated by line then it is referred as Linearly Separable sets otherwise it will be non-linear Separable Sets

- Therefore, if any 2 sets can be separated by line then it is referred as Linearly Separable sets otherwise it will be non-linear Separable Sets



Perceptron


Perceptron
Single Layer Perceptron
Multi Layer Perceptron

- Goal of the Single Layer Perceptron
  - Classify externally applied input into 2 classes
  - if v is greater than or equal to 0, then input X belongs to C_1 otherwise, input X belongs to C_2.

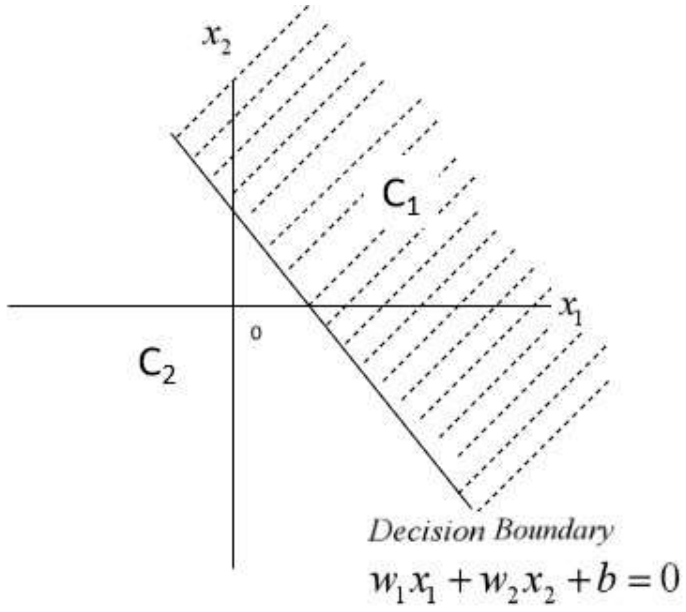


fig 1: Illustration of the hyperplane

- Training the Single-Layer Perceptron:
- Rosenblatt's Algorithm/Perceptron Learning Rule:
  - It is a Supervised Learning.
  - This algorithm, learn the pattern/classes in finite number of steps.
  - It can be applied only for the linearly-separable classes

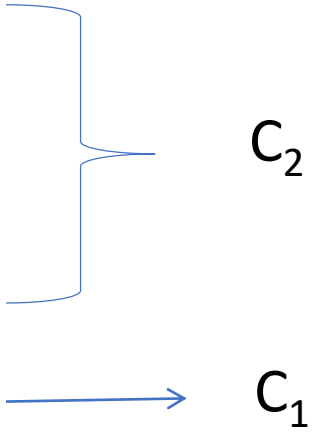
Artificial Neural Network

Single Layer Perceptron

- Single layer perceptron:

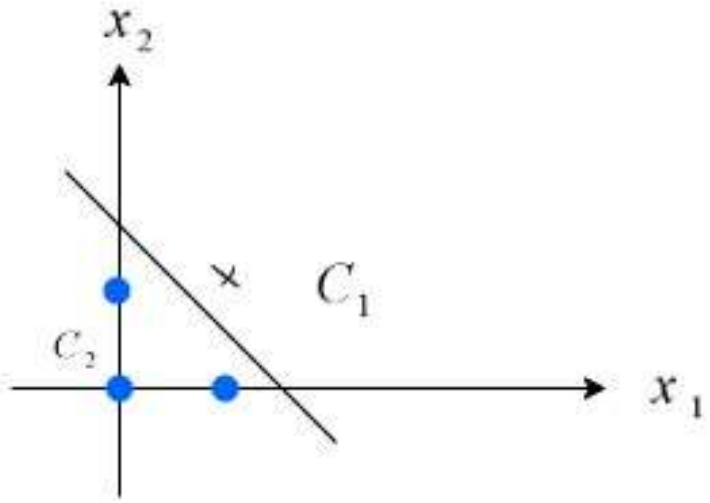
Consider 2 input AND Logic Gate

x	y	z
0	0	0
1	0	0
0	1	0
1	1	1



Single Layer Peceptron

- Lets design the 2 input AND gate using single layer perceptron using Rosennblatt's Algorithm



Rosenblatt's algorithm:

Let W(1) be any initial choice of the weight vector and X(k) be any sequence in $C_1 \cup C_2$

At the kth stage, Let W(k) be the weight vector

If X(k) is correctly classified, then no changes, i.e no updation of wieghts

Otherwise, updation in weights as follows

$$W(k+1) = W(k) + \begin{cases} -\eta X(k) & W^T(k)X(k) \geq 0 \;\&\; X(k) \in C_2 \\ \eta X(k) & W^T(k)X(k) < 0 \;\&\; X(k) \in C_1 \end{cases}$$

Reference

- S. Haykin, (2003), “Neural Networks: A Comprehensive Foundation”, 2nd edition, Prentice Hall of India.

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

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

Single Layer Perceptron

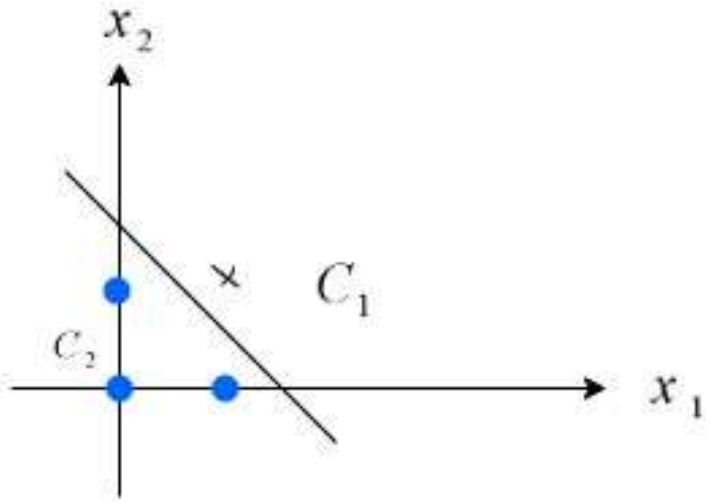
- Single layer perceptron:

Consider 2 input AND Logic Gate

x	y	z	
0	0	0	
1	0	0	C_2
0	1	0	
1	1	1	C_1

Single Layer Peceptron

- Lets design the 2 input AND gate using single layer perceptron using Rosennblatt's Algorithm



Rosenblatt's algorithm:

Let W(1) be any initial choice of the weight vector and X(k) be any sequence in $C_1 \cup C_2$

At the kth stage, Let W(k) be the weight vector

If X(k) is correctly classified, then no changes, i.e no updation of wieghts

Otherwise, updation in weights as follows

$$W(k+1) = W(k) + \begin{cases} -\eta X(k) & W^T(k)X(k) \geq 0 \;\&\; X(k) \in C_2 \\ \eta X(k) & W^T(k)X(k) < 0 \;\&\; X(k) \in C_1 \end{cases}$$

Rosenblatt's algorithm:

Let W(1)=[0.1 1 1.1]',
X(1)=[1 0 0]
Learning rate=0.1

Iteration 1: n=1

$$v = W^T(1)X(1)$$

$$v = 0.1$$

$$\Rightarrow X(1) \in C_1$$

But X(1) belongs to C_2

Therefore, update the weights as follows:

$$W(2) = W(1) - \eta X(1)$$

$$W(2) = \begin{pmatrix} 0.1 \\ 1 \\ 1.1 \end{pmatrix} - (0.1)\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1.1 \end{pmatrix}$$

Rosenblatt's algorithm:

Iteration 2: n=2

$$v = W^T(2)X(2)$$

$$v = 1.1$$

$$\Rightarrow X(1) \in C_1$$

But X(2) belongs to C_2

Therefore, update the weights as follows:

$$W(3) = W(2) - \eta X(2)$$

$$W(3) = \begin{pmatrix} 0 \\ 1 \\ 1.1 \end{pmatrix} - (0.1)\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.1 \\ 1 \\ 1 \end{pmatrix}$$

Rosenblatt's algorithm:

Iteration 3: n=3

$$v = W^T(3)X(3)$$

$$v = 0.9$$

$$\Rightarrow X(3) \in C_1$$

But X(3) belongs to C_2

Therefore, update the weights as follows:

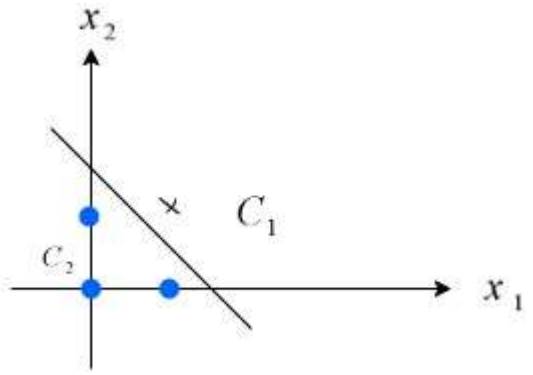
$$W(4) = W(3) - \eta X(3)$$

$$W(4) = \begin{pmatrix} -0.1 \\ 1 \\ 1 \end{pmatrix} - (0.1)\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.2 \\ 0.9 \\ 1 \end{pmatrix}$$

This procedure will continue till perceptron learns all the patterns

- Let the Choice of W=[-0.75 0.25 0.5]'

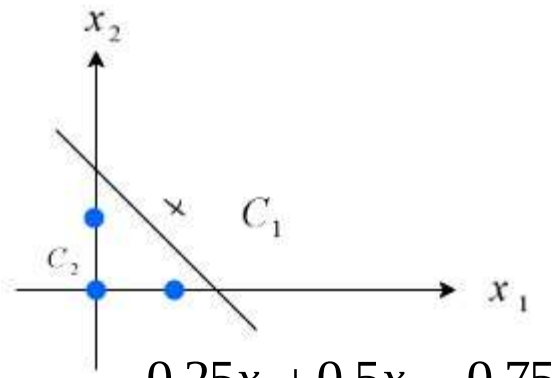
x_0	x_1	x_2	v	$\varphi(v)$	y
1	0	0	-0.75	0	0
1	0	1	-0.25	0	0
1	1	0	-0.5	0	0
1	1	1	0	1	1



$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

- Let the Choice of W=[-0.75 0.25 0.5]'

x_0	x_1	x_2	v	$\varphi(v)$	y
1	0	0	-0.75	0	0
1	0	1	-0.25	0	0
1	1	0	-0.5	0	0
1	1	1	0	1	1



$$0.25x_1 + 0.5x_2 - 0.75 = 0$$

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

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*

Perceptron

Perceptron Convergence Theorem:

If there is a weight vector W* such that

$$\varphi\left(W^{*}X(k)\right)=y(k)$$

then for any starting vector W, the perceptron learning rule will converge to weight vector W* that gives the correct response for all training patterns and it will do so in finite number of steps

Perceptron Convergence Theorem

Assumptions:

- Inputs to the perceptron originate from 2 linearly separable classes.
- W(0)=0
- Learning rate is 1 and it remains constant

Perceptron Convergence Theorem

PES UNIVERSITY ONLINE

Proof:

- Let
  - C_1 and C_2 be 2 different classes,
  - $H_1 \subset C_1$ and $H_2 \subset C_2$
- Define $X(n) = [+1, x_1(n), x_2(n), \cdots, x_m(n)]^T$

  $W(n) = [b(n), w_1(n), w_2(n), \cdots, w_m(n)]^T$
- Require a 'W' such that,

$$w \in R^{m+1},$$

$$v(x) = W^T X \geq 0 \forall X \in H_1$$

$$v(x) = W^T X < 0 \forall X \in H_2$$

Perceptron Convergence Theorem

For $k = 0$

$$v(0) = W^T(0)X(0) = 0$$

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

if $X(0) \in H_1$, then $W(1) = W(0)$

otherwise $x(1) \in H_2$, then $W(2) = W(1) - X(1) = -X(1)$

To understand in easier way, i assume that the perceptron incorrectly classifies the vectors x(1),X(2)... Therefore,

$$W^T(k)X(k) < 0 \ for \quad k = 1,2,...and$$

$$the \quad input \quad vector \quad X(k) \in H_1$$

Perceptron Convergence Theorem

For $k = 1$

$$v(1) = W^T(1)X(1) < 0$$

since $X(1) \in H_1$ $\quad W(2) = W(1) + X(1) = X(1)$

For $k = 2$

$$v(2) = W^T(2)X(2) < 0$$

$X(2) \in H_1$ and

$$W(3) = W(2) + X(2) = X(1) + X(2)$$

At the kth stage we obtain

$$W(k+1) = X(1) + X(2) + ... + X(k)$$

Since the Classes C1 and C2 are assumed to be linearly separable, there exists a solution W_0.

Then for a fixed solution W_0, we may define a positive

$$\alpha = \min_{X(i) \in H_1} W_0^T X(i)$$

Perceptron

$$W(k+1) = X(1) + X(2) + ... + X(k)$$

Multiply W_0 on both the side of the equation

$$W_0^T W(k+1) = W_0^T X(1) + W_0^T X(2) + ... + W_0^T X(k)$$

$$W_0^T W(k+1) \geq k\alpha \quad \text{.....(1)}$$

Perceptron

Let $x, y \in R^m$, then $\left|x^T y\right| \leq \|x\|^2 \|y\|^2$

- The above inequality is referred as Cauchy-Schwarz inequality

Therefore, $\left|W_0{}^T W(k+1)\right| \leq \|W_0\|^2 \|W(k+1)\|^2$

$$\|W(k+1)\|^2 \geq \frac{k^2 \alpha^2}{\|W_0\|^2} \qquad \cdots\cdots(a)$$

Perceptron

Under the initial assumption

$$W(k+1) = W(k) + X(k) \qquad for \quad k = 1,2...$$

By taking the squared Euclidean norm of both sides of above equation

$$\|W(k+1)\|^2 = \|W(k)\|^2 + \|X(k)\|^2 + 2W^T(k)X(k)$$

$$\|W(k+1)\|^2 \leq \|W(k)\|^2 + \|X(k)\|^2$$

$$\|W(k+1)\|^2 - \|W(k)\|^2 \leq \|X(k)\|^2$$

$$\|W(k+1)\|^2 \leq \sum_{X(i)\in C_1} \|X(k)\|^2$$

Let

$$\beta = \max_{X(k)\in C_1} \|X(k)\|^2$$

$$\Rightarrow \|W(k+1)\|^2 \leq k\beta \qquad \cdots\cdots(b)$$

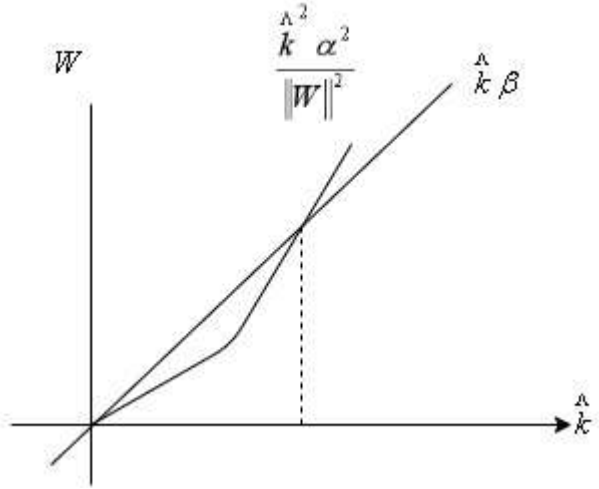
Compare (a) and (b)

$$\frac{k^2\alpha^2}{\|W_0\|^2} \leq \|W(k+1)\|^2 \leq k\beta$$

Artificial Neural Network

Perceptron

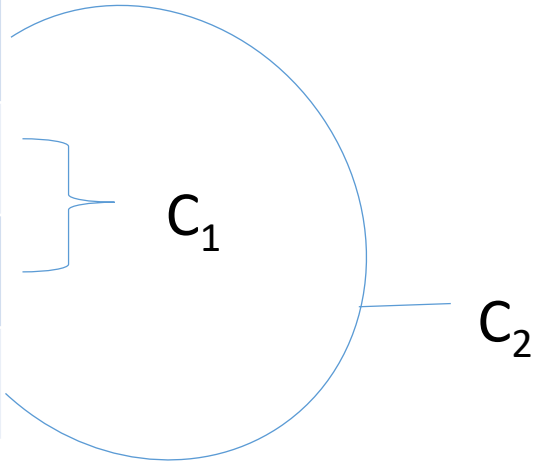
$$k_{\max} = \frac{\beta \|W\|^2}{\alpha^2}$$



Single Layer Perceptron

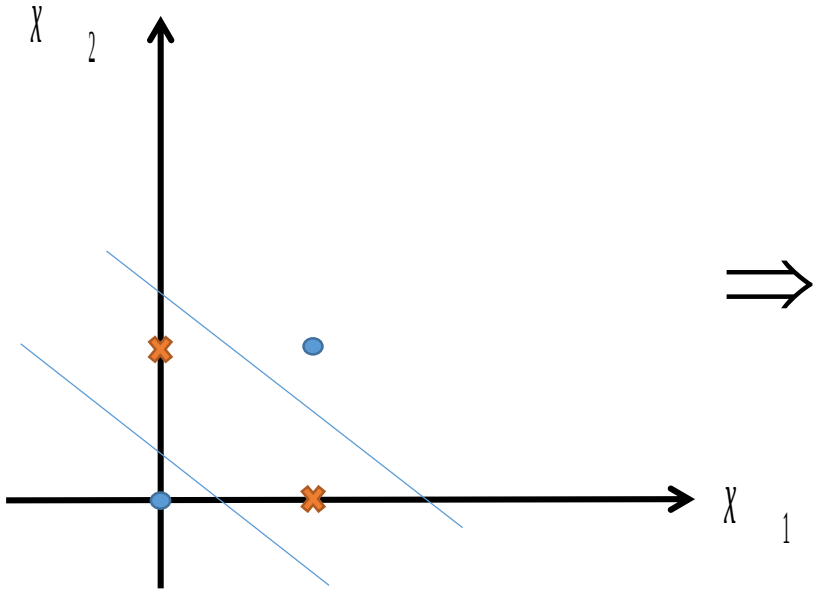
- Now lets consider 2 input XOR logic gate.
- Is it possible to design the gate using Single layer perceptron?

x	y	z
0	0	0
1	0	1
0	1	1
1	1	0



Artificial Neural Network-Perceptron

Single Layer Perceptron



$\Rightarrow$

- Therefore, Single-layer Perceptron cannot be used in this case.
- This problem can be solved using Multi-layer Perceptron

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

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*

Perceptron

Perceptron Convergence Theorem:

If there is a weight vector W* such that

$$\varphi\left(W^{*}X(k)\right)=y(k)$$

then for any starting vector W, the perceptron learning rule will converge to weight vector W* that gives the correct response for all training patterns and it will do so in finite number of steps

Perceptron Convergence Theorem

Assumptions:

- Inputs to the perceptron originate from 2 linearly separable classes.
- W(0)=0
- Learning rate is 1 and it remains constant

Perceptron Convergence Theorem

PES UNIVERSITY ONLINE

Proof:

- Let
  - C_1 and C_2 be 2 different classes,
  - $H_1 \subset C_1$ and $H_2 \subset C_2$
- Define $X(n) = [+1, x_1(n), x_2(n), \cdots, x_m(n)]^T$

  $W(n) = [b(n), w_1(n), w_2(n), \cdots, w_m(n)]^T$
- Require a 'W' such that,

$$w \in R^{m+1},$$

$$v(x) = W^T X \geq 0 \forall X \in H_1$$

$$v(x) = W^T X < 0 \forall X \in H_2$$

Perceptron Convergence Theorem

For $k = 0$

$$v(0) = W^T(0)X(0) = 0$$

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

if $X(0) \in H_1$, then $W(1) = W(0)$

otherwise $x(1) \in H_2$, then $W(2) = W(1) - X(1) = -X(1)$

To understand in easier way, i assume that the perceptron incorrectly classifies the vectors x(1),X(2)... Therefore,

$$W^T(k)X(k) < 0 \ for \quad k = 1,2,...and$$

$$the \quad input \quad vector \quad X(k) \in H_1$$

Perceptron Convergence Theorem

For $k = 1$

$$v(1) = W^T(1)X(1) < 0$$

since $X(1) \in H_1$ $\quad W(2) = W(1) + X(1) = X(1)$

For $k = 2$

$$v(2) = W^T(2)X(2) < 0$$

$X(2) \in H_1$ and

$$W(3) = W(2) + X(2) = X(1) + X(2)$$

At the kth stage we obtain

$$W(k+1) = X(1) + X(2) + ... + X(k)$$

Since the Classes C1 and C2 are assumed to be linearly separable, there exists a solution W_0.

Then for a fixed solution W_0, we may define a positive

$$\alpha = \min_{X(i) \in H_1} W_0^T X(i)$$

Perceptron

$$W(k+1) = X(1) + X(2) + ... + X(k)$$

Multiply W_0 on both the side of the equation

$$W_0^T W(k+1) = W_0^T X(1) + W_0^T X(2) + ... + W_0^T X(k)$$

$$W_0^T W(k+1) \geq k\alpha \quad \text{.....(1)}$$

Perceptron

Let $x, y \in R^m$, then $\left|x^T y\right| \leq \|x\|^2 \|y\|^2$

- The above inequality is referred as Cauchy-Schwarz inequality

Therefore, $\left|W_0{}^T W(k+1)\right| \leq \|W_0\|^2 \|W(k+1)\|^2$

$$\|W(k+1)\|^2 \geq \frac{k^2 \alpha^2}{\|W_0\|^2} \qquad \cdots\cdots(a)$$

Perceptron

Under the initial assumption

$$W(k+1) = W(k) + X(k) \qquad for \quad k = 1,2...$$

By taking the squared Euclidean norm of both sides of above equation

$$\|W(k+1)\|^2 = \|W(k)\|^2 + \|X(k)\|^2 + 2W^T(k)X(k)$$

$$\|W(k+1)\|^2 \leq \|W(k)\|^2 + \|X(k)\|^2$$

$$\|W(k+1)\|^2 - \|W(k)\|^2 \leq \|X(k)\|^2$$

$$\|W(k+1)\|^2 \leq \sum_{X(i)\in C_1} \|X(k)\|^2$$

Let

$$\beta = \max_{X(k)\in C_1} \|X(k)\|^2$$

$$\Rightarrow \|W(k+1)\|^2 \leq k\beta \qquad \cdots\cdots(b)$$

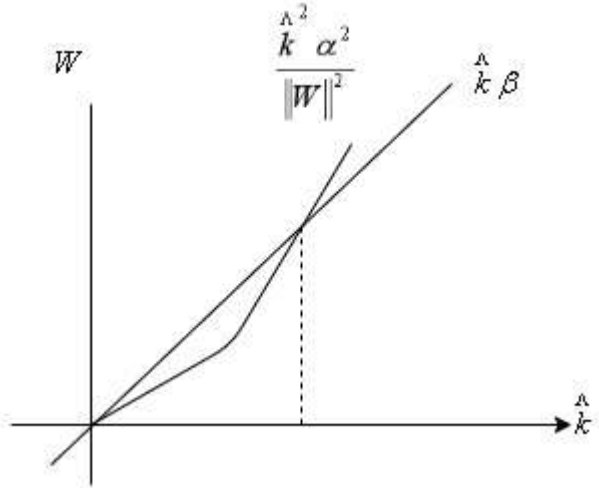
Compare (a) and (b)

$$\frac{k^2\alpha^2}{\|W_0\|^2} \leq \|W(k+1)\|^2 \leq k\beta$$

Artificial Neural Network

Perceptron

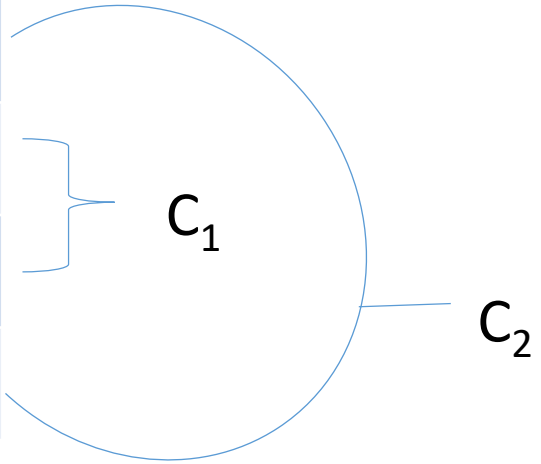
$$k_{\max} = \frac{\beta \|W\|^2}{\alpha^2}$$



Single Layer Perceptron

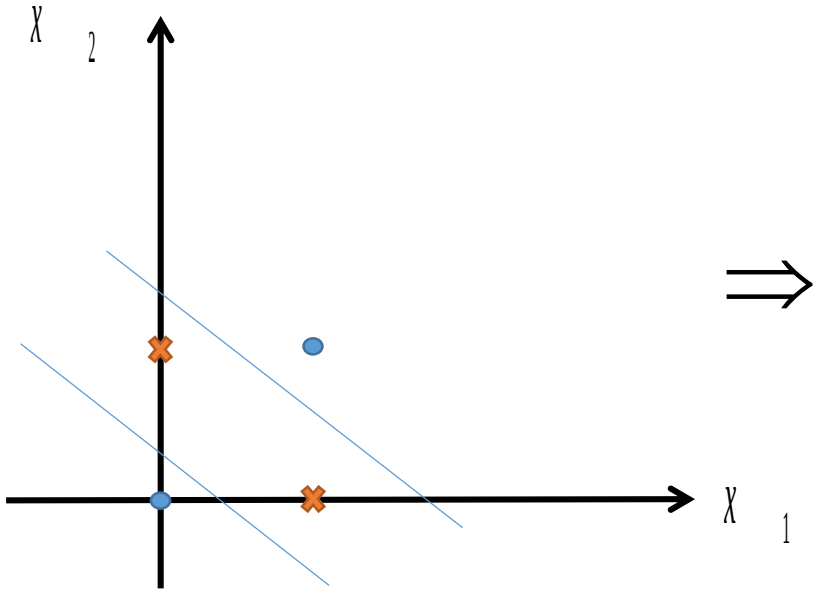
- Now lets consider 2 input XOR logic gate.
- Is it possible to design the gate using Single layer perceptron?

x	y	z
0	0	0
1	0	1
0	1	1
1	1	0



Artificial Neural Network-Perceptron

Single Layer Perceptron



$\Rightarrow$

- Therefore, Single-layer Perceptron cannot be used in this case.
- This problem can be solved using Multi-layer Perceptron

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

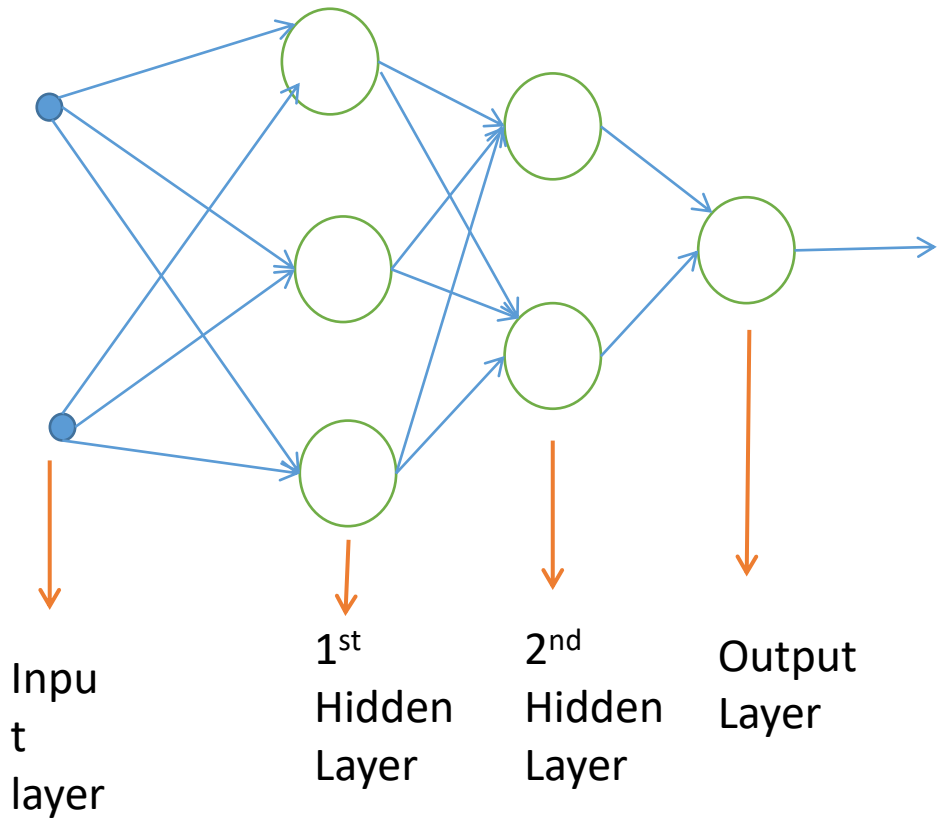
CONTENT:Part-1

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*

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)



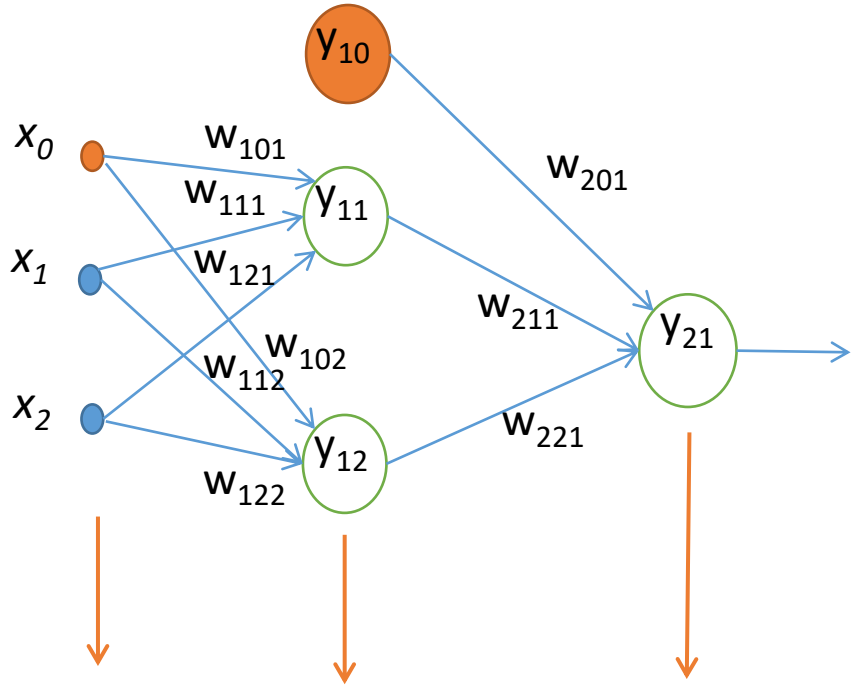
⇒

This Multilayer Perceptron is represented as 2:3:2:1

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Consider a neural network with 2:2:1 as shown below

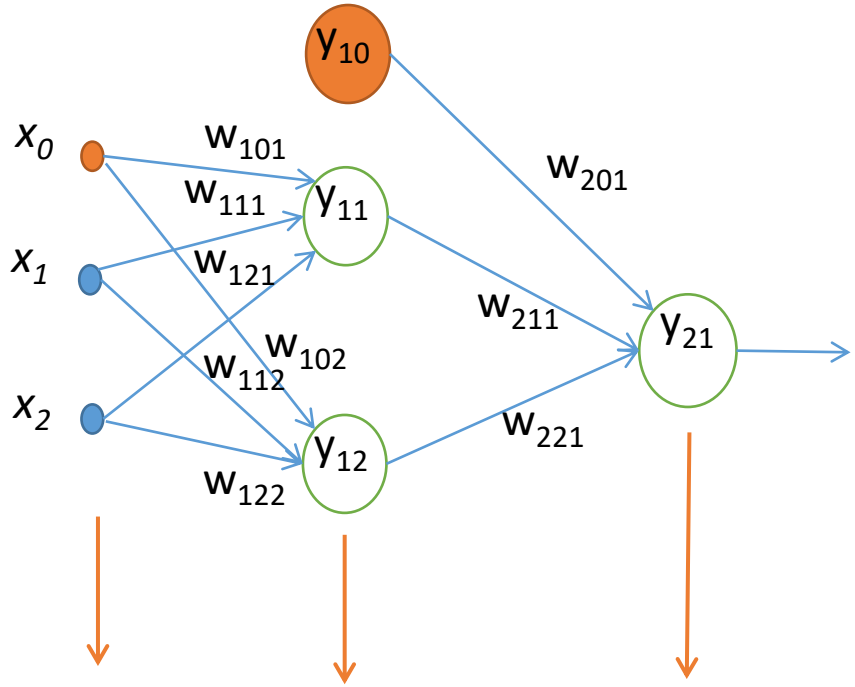


Layer	0	1	2
Index	m_0	m_1	m_2

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Consider a neural network with 2:2:1 as shown below



Layer	0	1	2
Index	m_0=2	m_1=2	m_2=1

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

- Procedure:

Step 1: Initialize the weight vector all layers

Step 2: Computation takes place at 2 stages

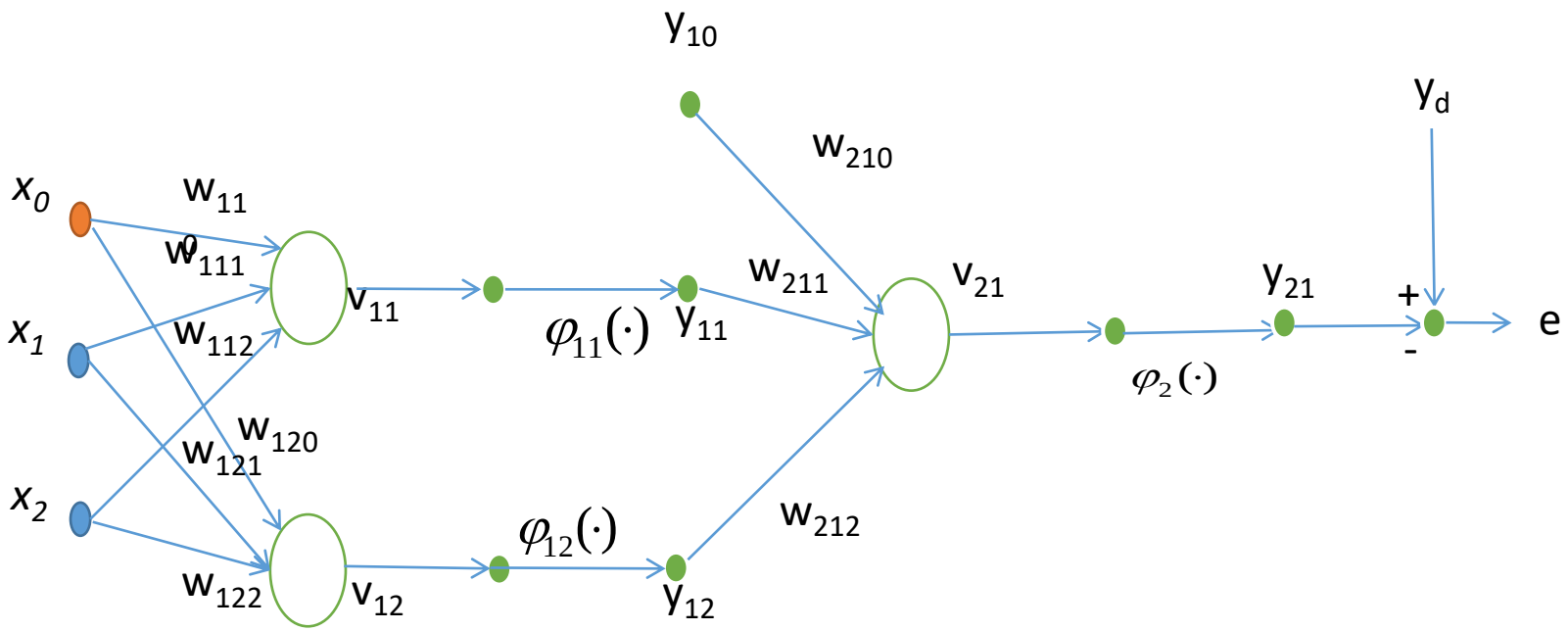
- Feedforward Pass
  - Compute output
- Backward Pass
  - Compute local gradient

Step 3: Update the weight vectors if required

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Feedforward 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 0: Input Layer

$$X(k) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad y_0(k) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

Layer 1: 1st Hidden Layer

Input : $y_0(k) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$

Weight Vector: $W_1 = \begin{pmatrix} w_{110} & w_{111} & w_{112} \\ w_{120} & w_{121} & w_{122} \end{pmatrix}$

Induced local field:

$$v_1 = \begin{pmatrix} v_{11} \\ v_{12} \end{pmatrix} = W_1(k) y_0(k)$$

Output of activation block:

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

Output of 1st Hidden Layer:

$$\bar{y}_1 = \begin{pmatrix} \varphi_{11}(v_{11}) \\ \varphi_{12}(v_{12}) \end{pmatrix} = \begin{pmatrix} y_{11} \\ y_{12} \end{pmatrix}$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Layer 2: Output Layer

Input : $y_1(k) = \begin{pmatrix} 1 \\ - \\ y_1(k) \end{pmatrix}$

Weight Vector: $W_2 = \begin{pmatrix} w_{210} & w_{211} & w_{211} \end{pmatrix}$

Induced local field:

$$v_2(k) = w_2(k) y_1(k)$$

Output of activation block:

$$\varphi_2(v_2(k)) = \varphi_{21}(v_2(k))$$

Output:

$$y_2 = \varphi_2(v_2(k))$$

Error:

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

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

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

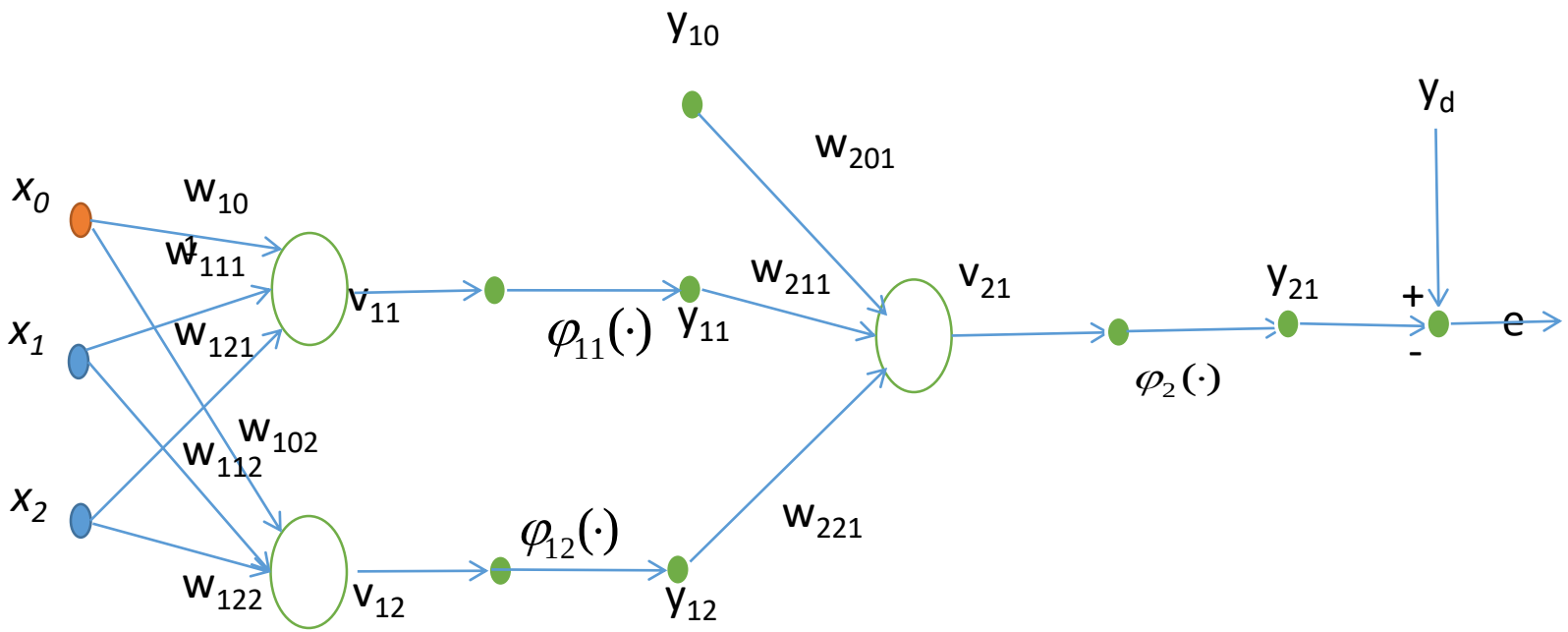
$$E = \frac{1}{2} e_1^2(k)$$

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:

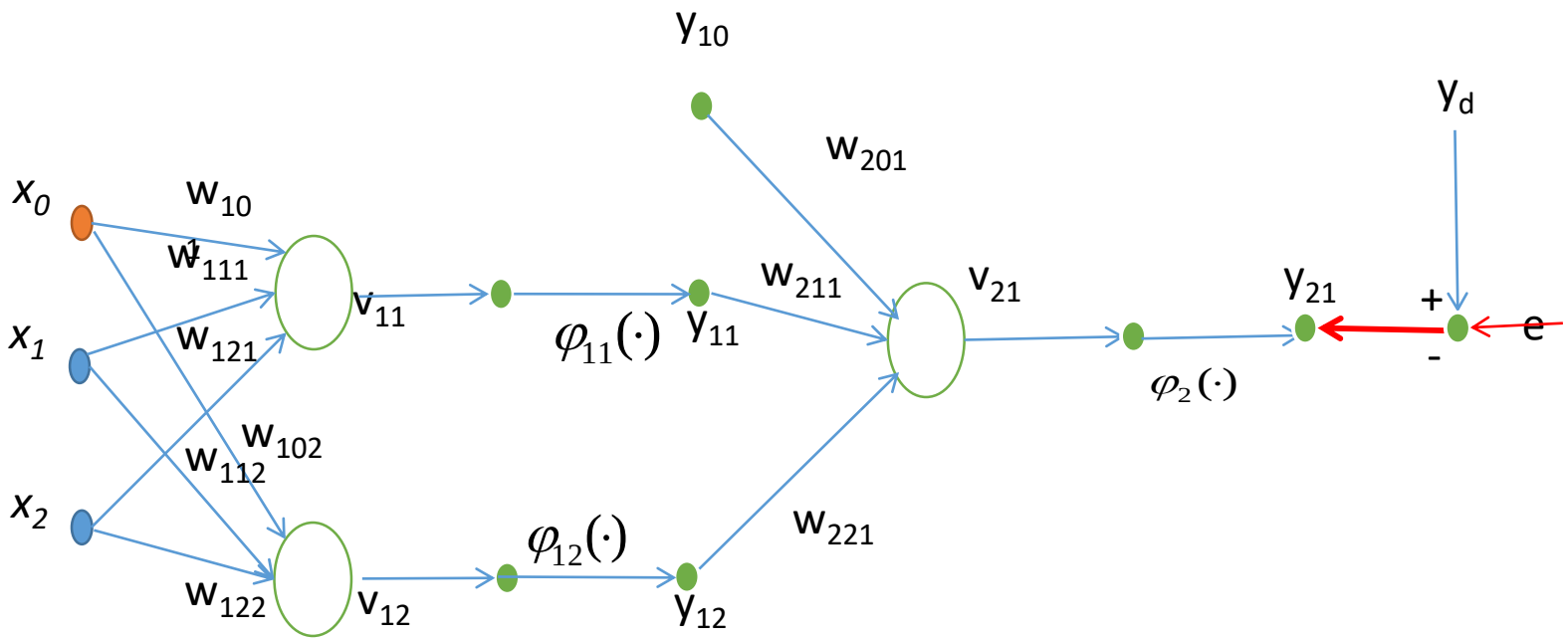


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

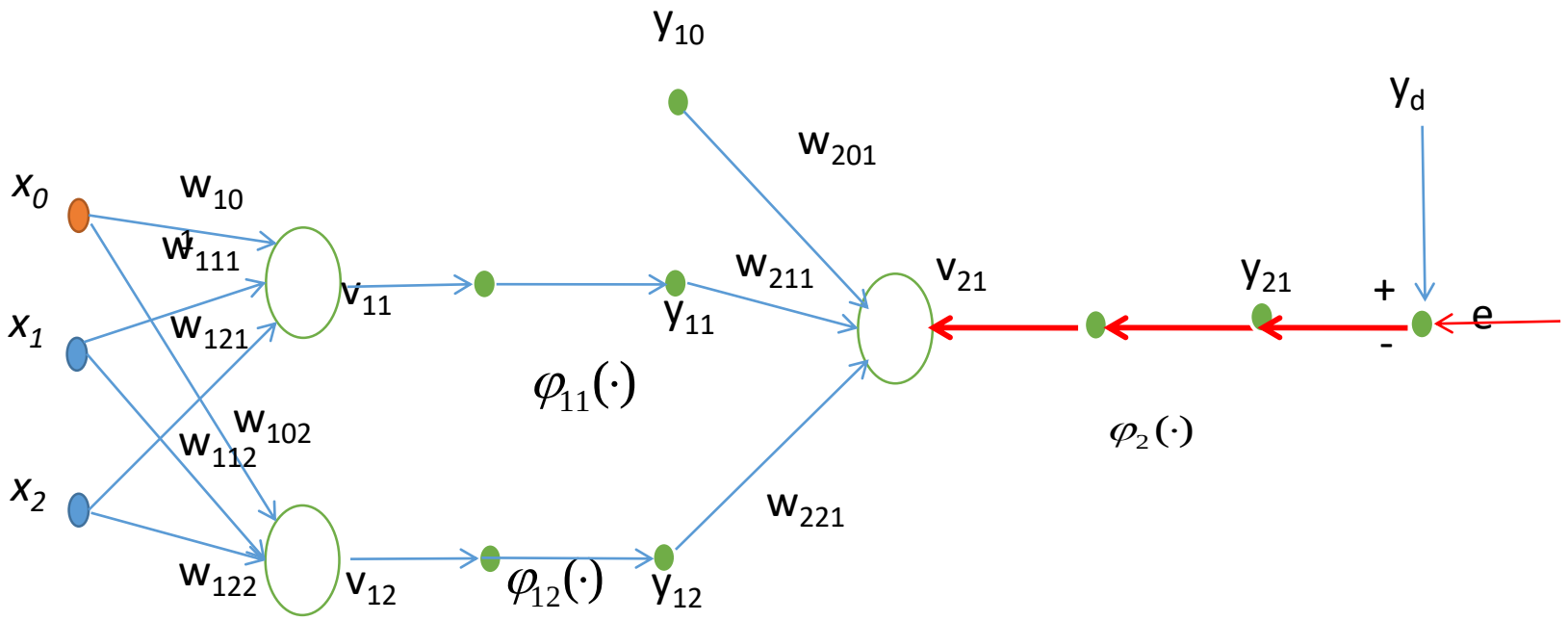


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

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)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

Delta Rule:

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

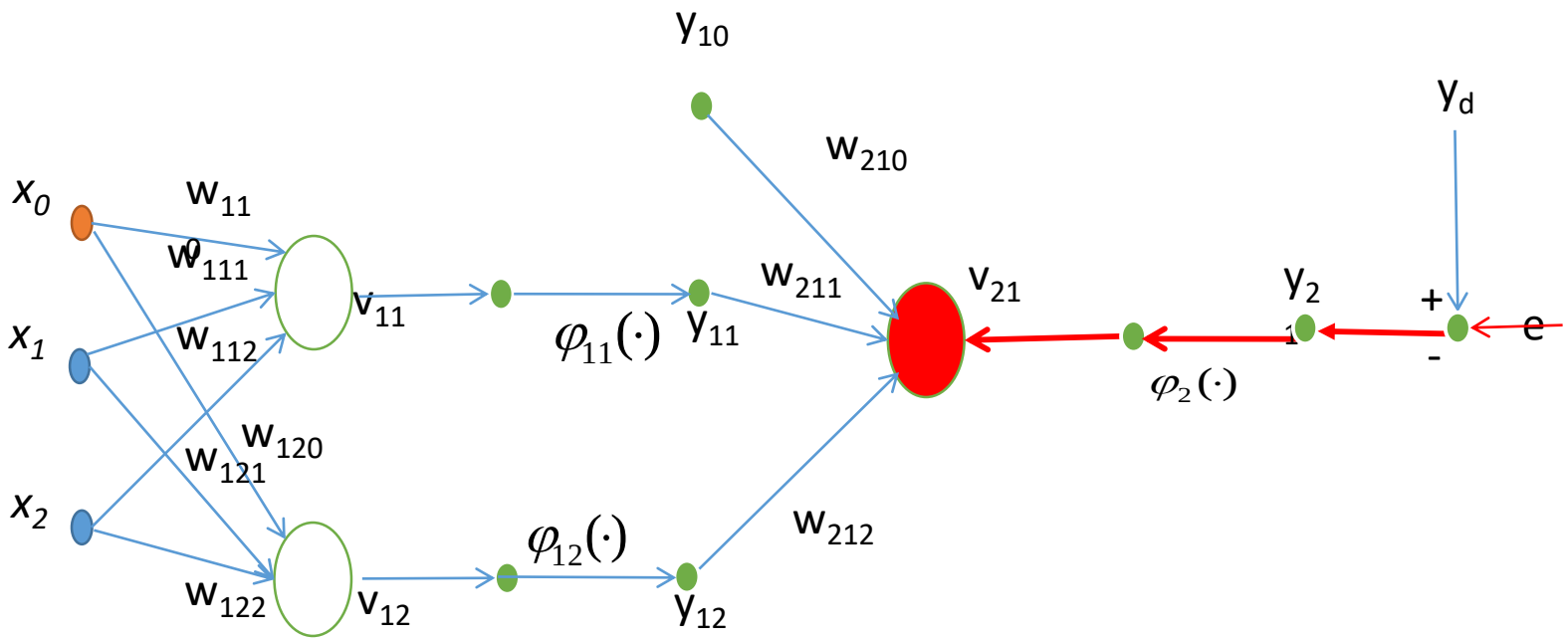
$$\Delta w_{plj}(k) = -\eta \frac{\partial E}{\partial w_{plj}(k)}$$

Layer 2: Output Layer

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

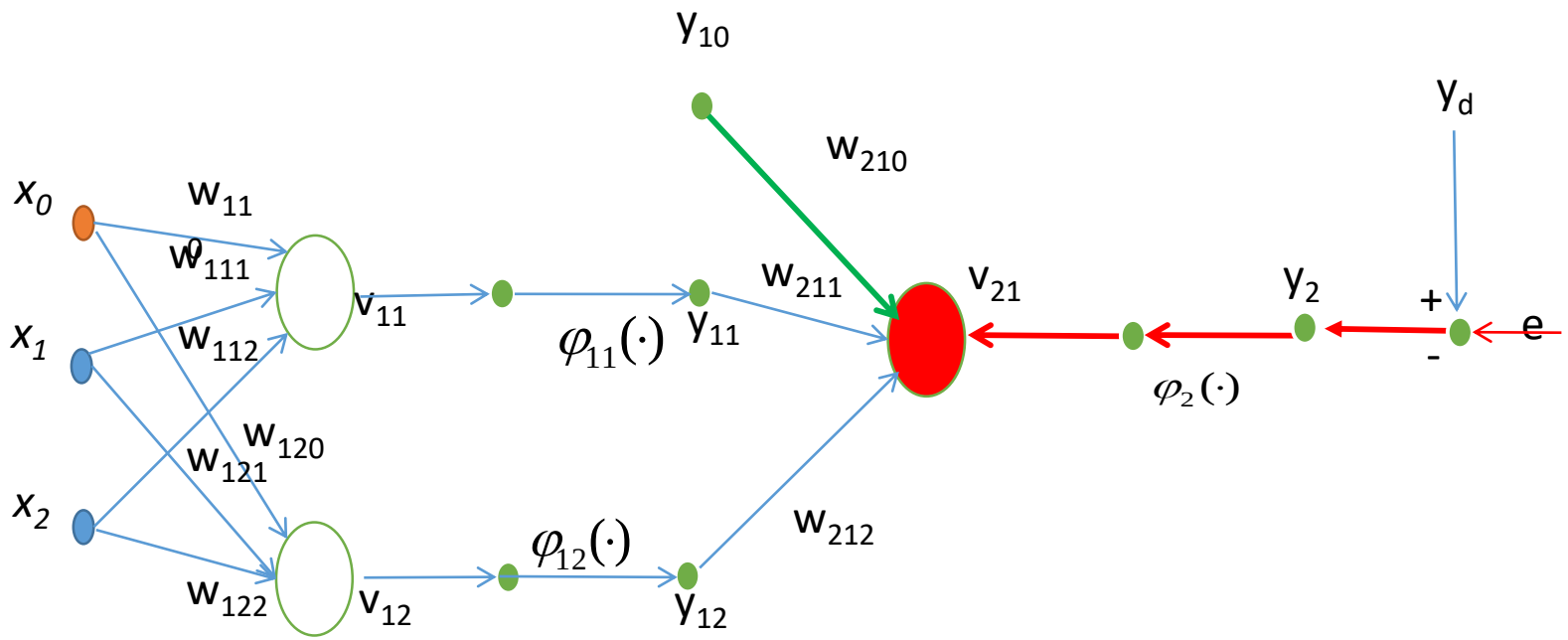


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

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

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

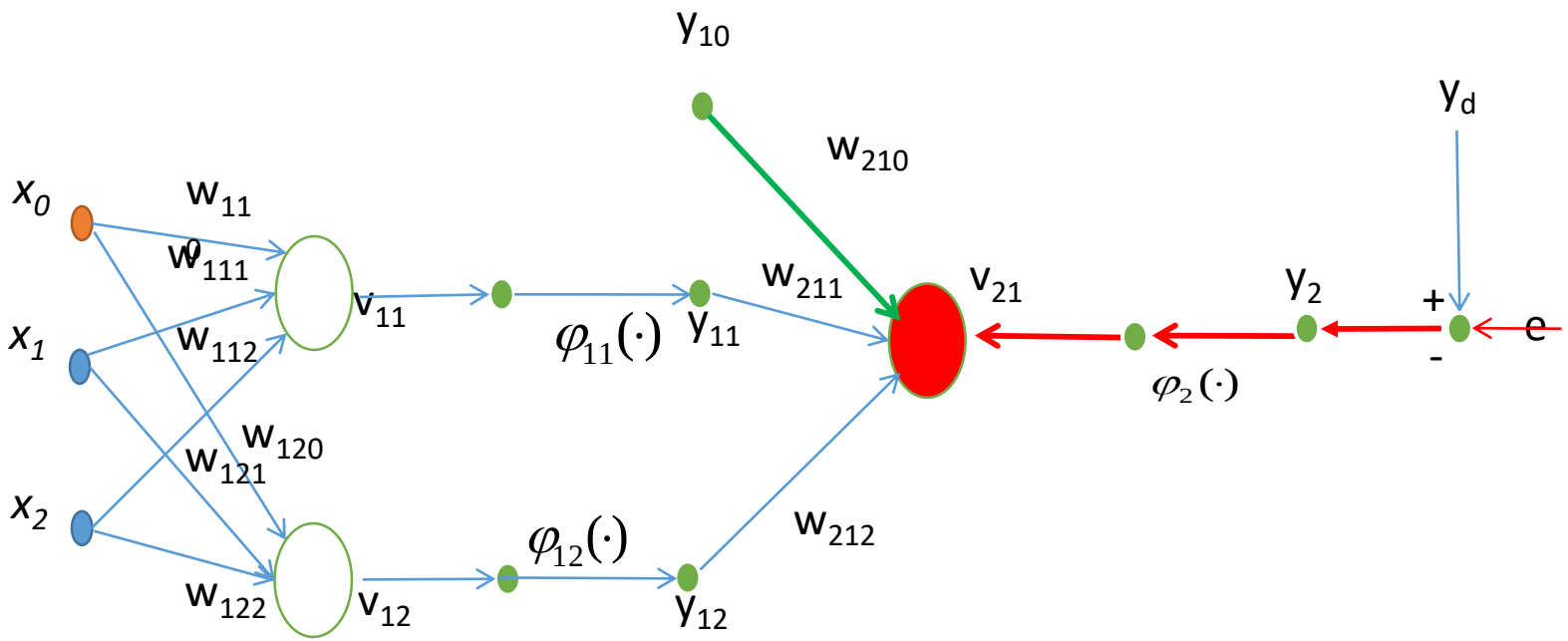
$$e_1 = y_d - y_{21}$$

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

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:



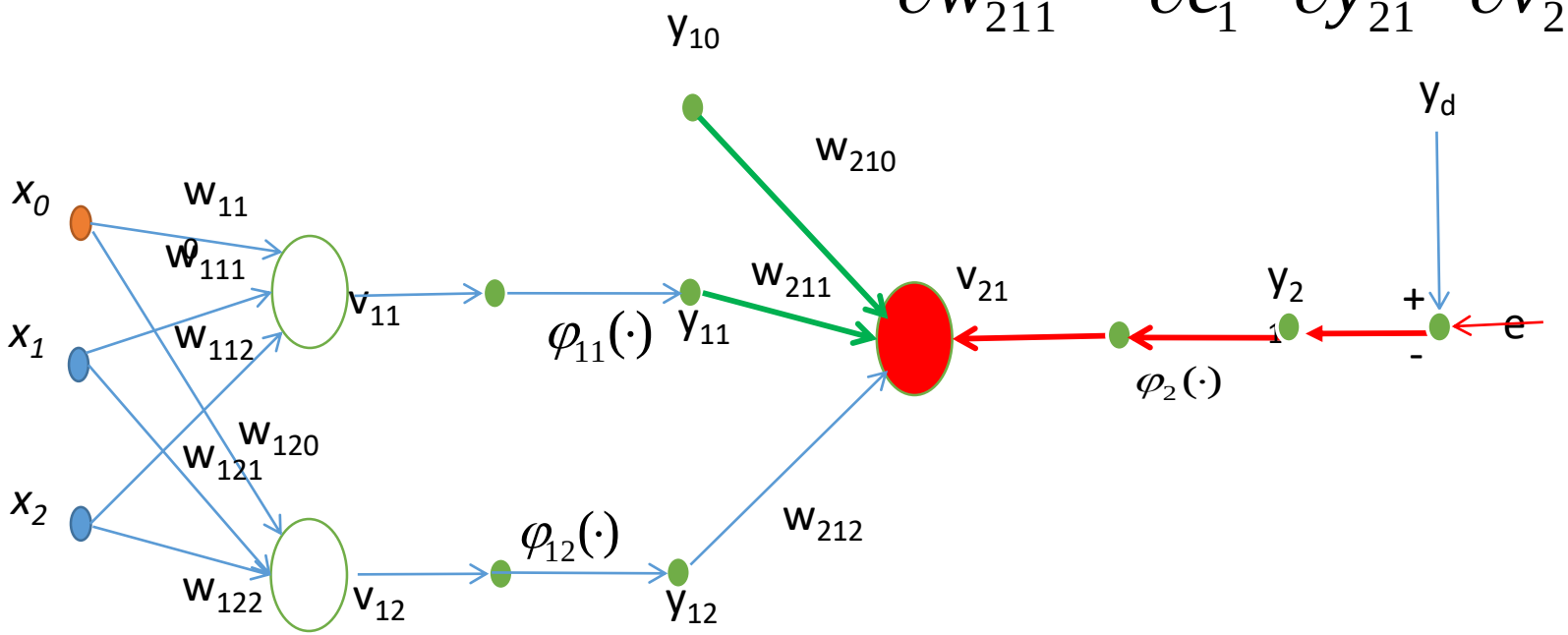
Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

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



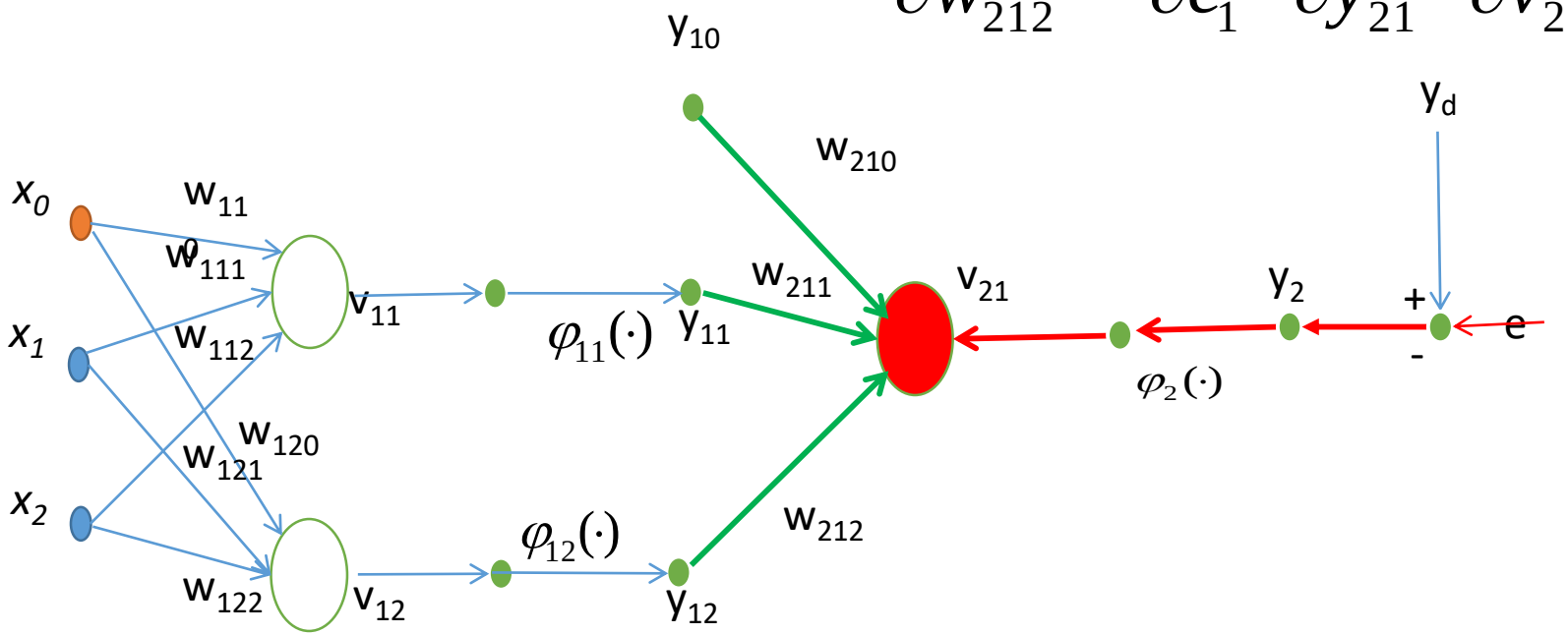
Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

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:

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,

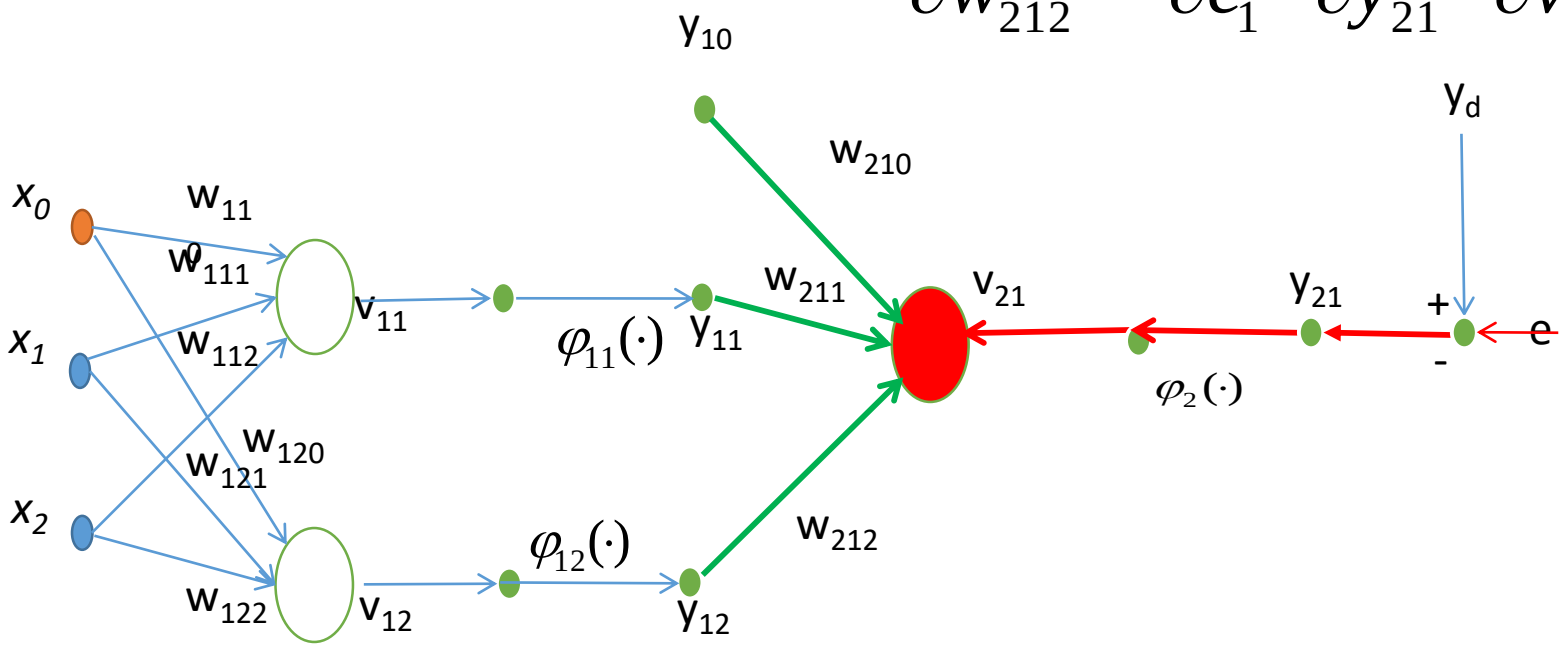
$$W_2(k+1) = W_2(k) + \Delta W_2(k)$$

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

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

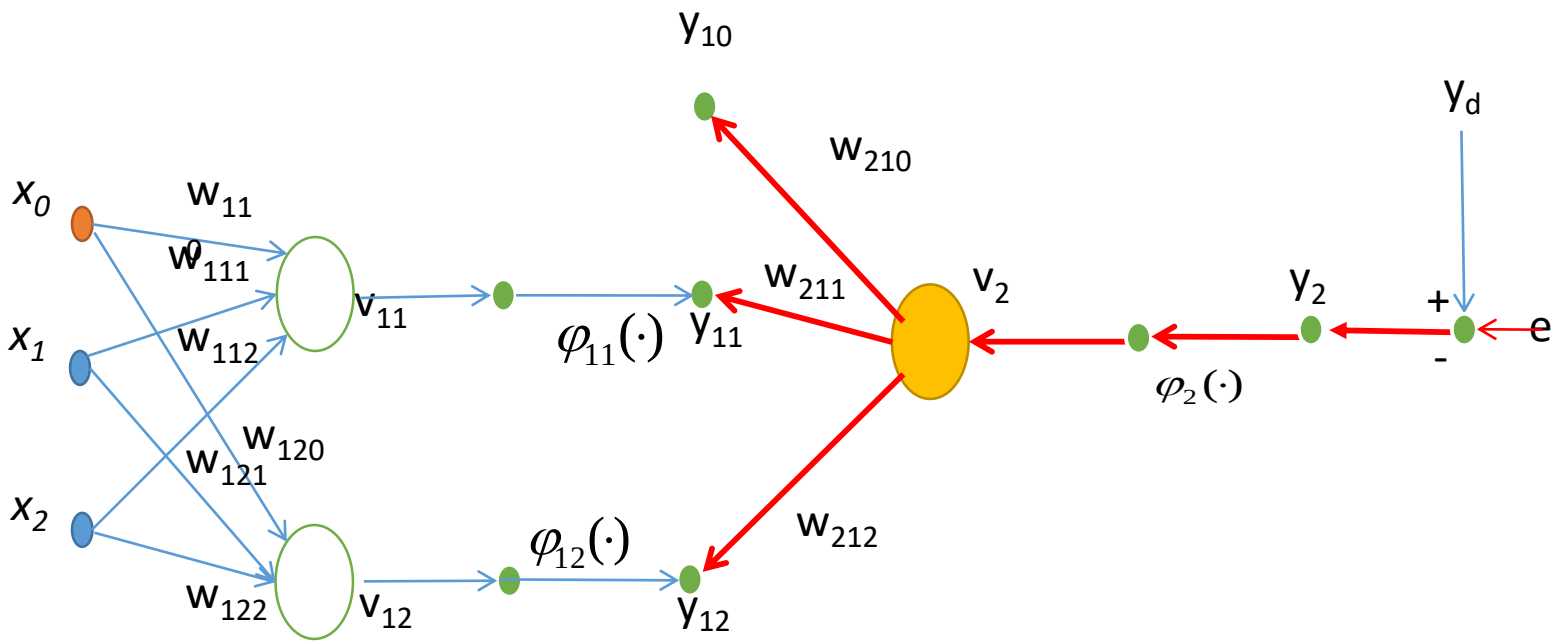


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

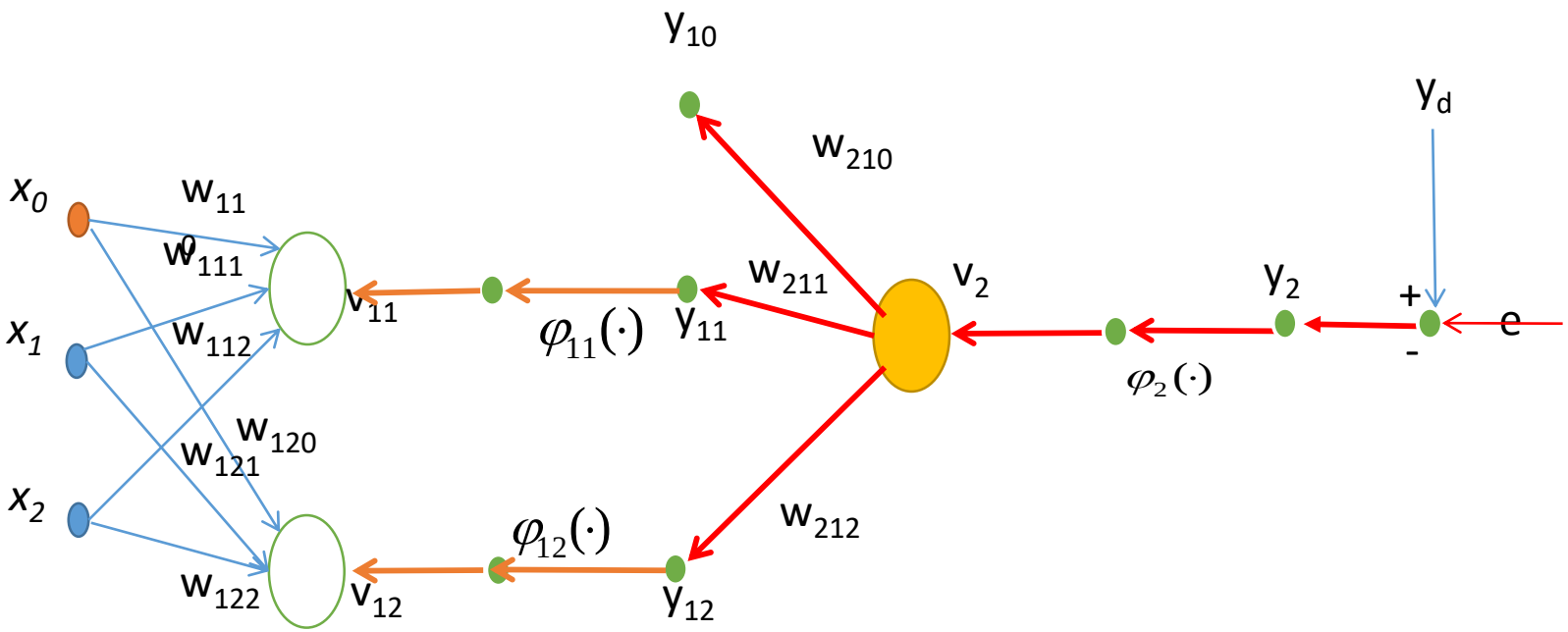


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

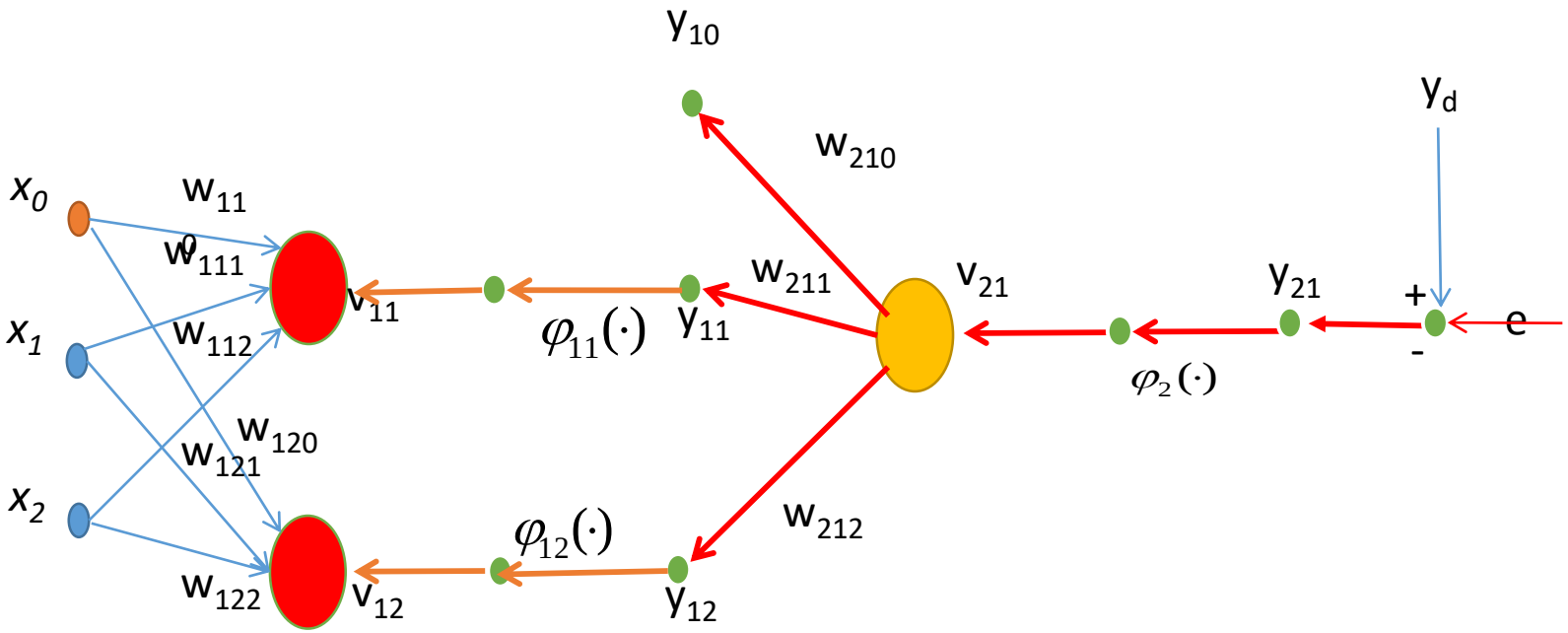


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

Backward Pass:

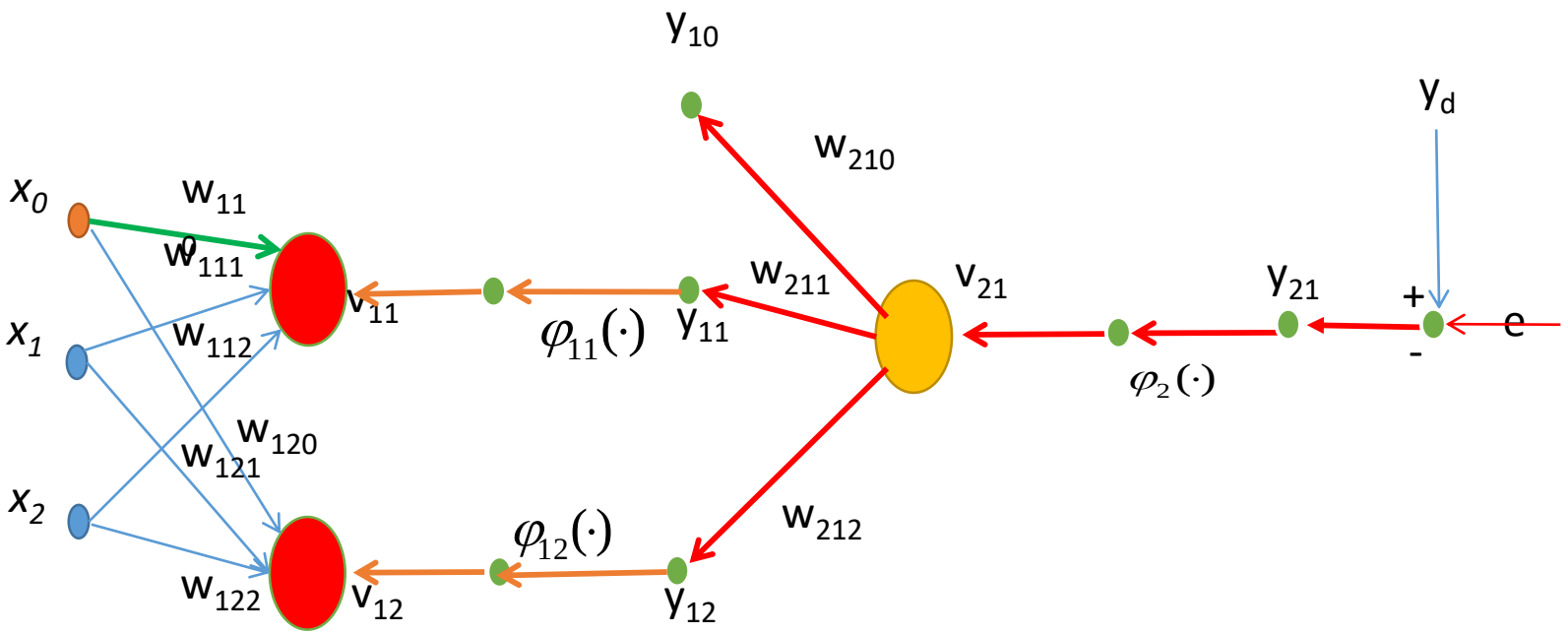


Layer: p	0	1	2
Index	i--> 0:(m_0=2)	j--> 0:(m_1=2)	l--> 0:(m_2=1)

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)

Layer 1: 1st Hidden layer

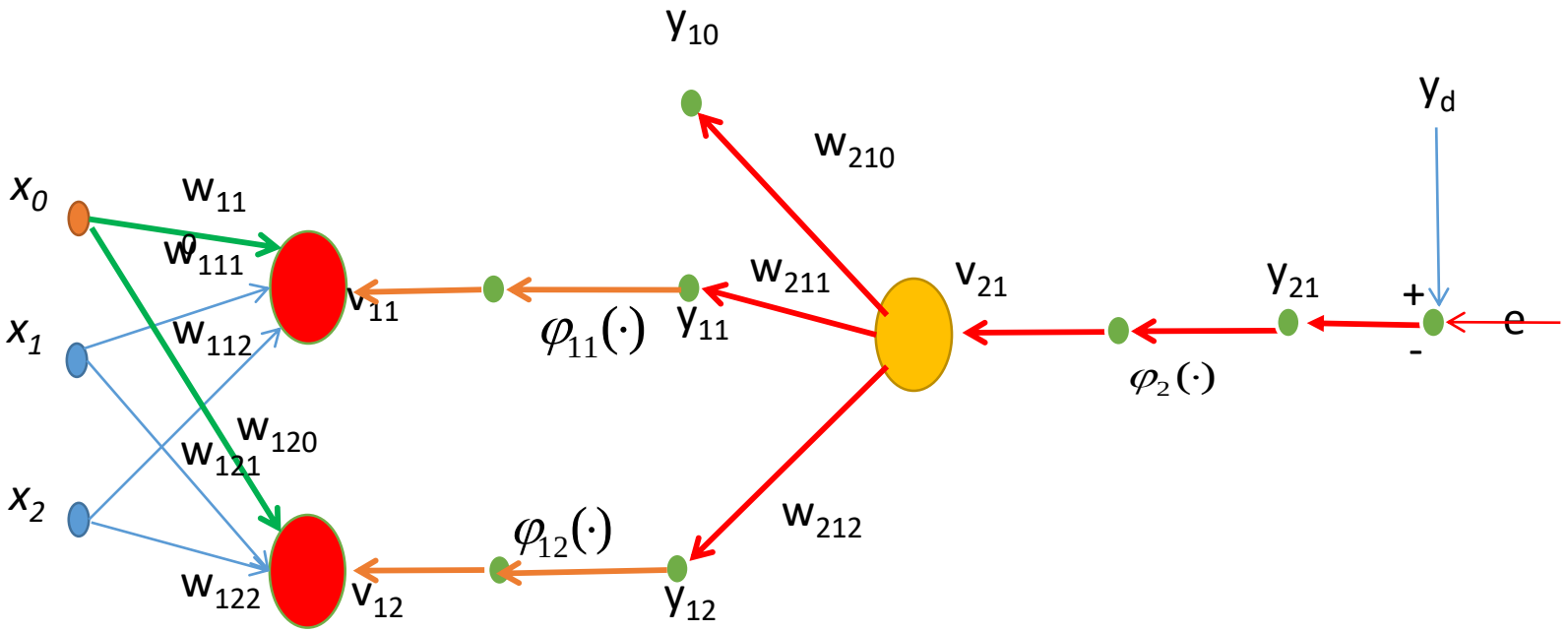
$$\frac{\partial E}{\partial w_{110}} = \frac{\partial E}{\partial e_1} \cdot \frac{\partial e_1}{\partial y_{21}} \cdot \frac{\partial y_{21}}{\partial v_{21}} \cdot \frac{\partial v_{21}}{\partial y_{11}} \cdot \frac{\partial y_{11}}{\partial v_{11}} \cdot \frac{\partial v_{11}}{\partial w_{110}}$$

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

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

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

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

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

$$\delta_2(k)$$

$$\delta_1(k)$$

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,

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

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

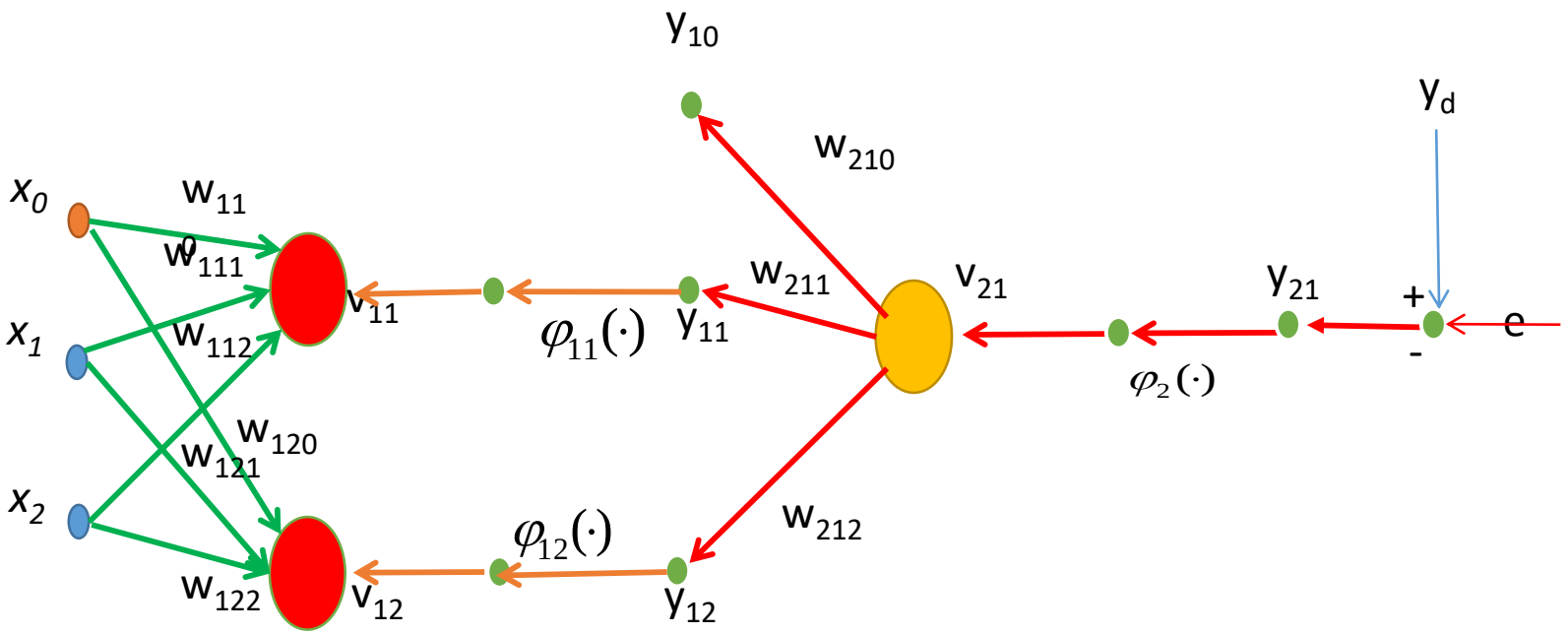
Therefore,

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

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)

Artificial Neural Network: Multi-layer Perceptron

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

Artificial Neural Network: Multi-layer Perceptron

Back-Propogation Algorithm (BPA)

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

$$e_j(k) = (desired\ output)$$

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

ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

Artificial Neural Network-Perceptron

BPA: Heuristics to make performance better

- **Sequential v/s Batch mode:**
  - For large data sequential processing is faster
  - For smaller data and network, batch processing is faster
- **Maximizing the information content**
- Every traning example presented to the BPA should be choosen on the basis that its information content is the largest possible for the task at hand
- This can be achieved in 2 ways

  a. use of an example that results in the largest traing error

  b. the use of example that is radially different from all those previously used

BPA: Heuristics to make performance better

Activation Function:

- A M.L.P trained with the back-propagation algorithm may, in general,learn faster when the asymmetric sigmoidal activation function are used in neuron model,for example tanh(.).

Target Value: it is important to choose the target value within the activation function range i.e +1 and -1

Normalization of the inputs:

The training samples must be preprocessed before presenting to the neural network. the preprocessing steps are

BPA: Heuristics to make performance better

a. subtracting each input from mean

b.decorrelate the training samples

c. covariance: it ensures the different synaptic weight in the networks learn at same speed.

Learning Rate

Initialization

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

Consider a M-class classification problems, which means each input pattern belong to one of the M-distinct classes.

Therefore, we will have M number outputs in the network.

Xj denotes the jth input pattern. and it is m-dimensional input vector

The output is M-dimensional vector. ykj is the kth ouput of network in correspondence to the jth input

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

Therefore, block diagram can be represented as

$$y_{kj} = F_k\left(x_j\right)$$

Let the output vector y_j

$$y_j = \begin{bmatrix} y_{1j} & y_{2j} & ..... & y_{kj} & ..... & y_{Mj} \end{bmatrix}^T$$

$$y_j = \begin{bmatrix} F_1(x_j) & F_2(x_j) & ..... & F_k(x_j) & ..... & F_M(x_j) \end{bmatrix}^T$$

$$y_j = F\left(x_j\right)$$

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

F(.) is continuous function and minimizes the emprical risk function

$$R = \frac{1}{2N}\sum_{j=1}^{N}\left\|d_j - F(x_j)\right\|^2$$

Now train the network with binary values as follows:

- The output space has M dimension
- An input Xj belongs to class Ck
- let d_k=1 when Xj belongs to class C_k otherwise 0

Output Decision Rule stated as follows:

Classify the random vector X as belonging to C_k if

$$F_k(X) > F_j(X) \quad for all k \neq j$$

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

Artificial Neural Network-Perceptron

BPA: Heuristics to make performance better

- **Sequential v/s Batch mode:**
  - For large data sequential processing is faster
  - For smaller data and network, batch processing is faster
- **Maximizing the information content**
- Every traning example presented to the BPA should be choosen on the basis that its information content is the largest possible for the task at hand
- This can be achieved in 2 ways

  a. use of an example that results in the largest traing error

  b. the use of example that is radially different from all those previously used

BPA: Heuristics to make performance better

Activation Function:

- A M.L.P trained with the back-propagation algorithm may, in general,learn faster when the asymmetric sigmoidal activation function are used in neuron model,for example tanh(.).

Target Value: it is important to choose the target value within the activation function range i.e +1 and -1

Normalization of the inputs:

The training samples must be preprocessed before presenting to the neural network. the preprocessing steps are

BPA: Heuristics to make performance better

a. subtracting each input from mean

b.decorrelate the training samples

c. covariance: it ensures the different synaptic weight in the networks learn at same speed.

Learning Rate

Initialization

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

Consider a M-class classification problems, which means each input pattern belong to one of the M-distinct classes.

Therefore, we will have M number outputs in the network.

Xj denotes the jth input pattern. and it is m-dimensional input vector

The output is M-dimensional vector. ykj is the kth ouput of network in correspondence to the jth input

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

Therefore, block diagram can be represented as

$$y_{kj} = F_k\left(x_j\right)$$

Let the output vector y_j

$$y_j = \begin{bmatrix} y_{1j} & y_{2j} & ..... & y_{kj} & ..... & y_{Mj} \end{bmatrix}^T$$

$$y_j = \begin{bmatrix} F_1(x_j) & F_2(x_j) & ..... & F_k(x_j) & ..... & F_M(x_j) \end{bmatrix}^T$$

$$y_j = F\left(x_j\right)$$

Artificial Neural Network-Perceptron

BPA-Output representation and decision rule

F(.) is continuous function and minimizes the emprical risk function

$$R = \frac{1}{2N}\sum_{j=1}^{N}\left\|d_j - F(x_j)\right\|^2$$

Now train the network with binary values as follows:

- The output space has M dimension
- An input Xj belongs to class Ck
- let d_k=1 when Xj belongs to class C_k otherwise 0

Output Decision Rule stated as follows:

Classify the random vector X as belonging to C_k if

$$F_k(X) > F_j(X) \quad for all k \neq j$$

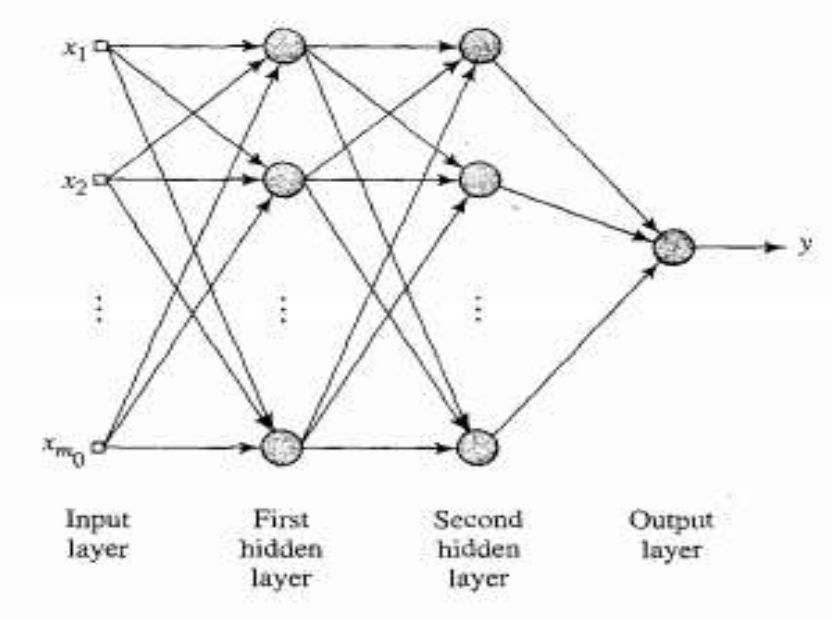
Artificial Neural Network-Perceptron

BPA-Feature detection

- Hidden layer neurons play a important role in the operation of MLP with BPA because they act as a feature detectors.
- As the laerning process progresses, the hidden neurons begin to discover the salient features that characterizes the traing data.
- Performing non linear transformation from the input space to hidden space.
- For example,non linearly separable classes will be easily separated in the hidden space

Artificial Neural Network-Perceptron

Back-propagation and differentiation



The above MLP is parameterised by an architecture A (representing a discrete parameter) and a weight vector W

Artificial Neural Network-Perceptron

Back-propagation and differentiation

Let Alj denote the part of the architecture extending from the input layer to node j in layer (l=1,2,3). Accordingly we may write

$$F(W,X)=\varphi(A_{31})$$

$$\frac{\partial F(W,X)}{\partial w_{3lk}}=\varphi'(A_{3l})\varphi(A_{2k})$$

$$\frac{\partial F(W,X)}{\partial w_{2kj}}=\varphi'(A_{3l})\varphi'(A_{2k})\varphi(A_{1j})w_{3lk}$$

$$\frac{\partial F(W,X)}{\partial w_{1ji}}=\varphi'(A_{3l})\varphi'(A_{1j})X_j\left[\sum_k w_{3lk}\varphi'(A_{2k})w_{2kj}\right]$$

The sensitivit of F(w,x) is

$$S_w^F = \frac{\partial F / F}{\partial w / w}$$

Home Work:

1. Jacobian Matrix
2. Hessian Matrix

ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

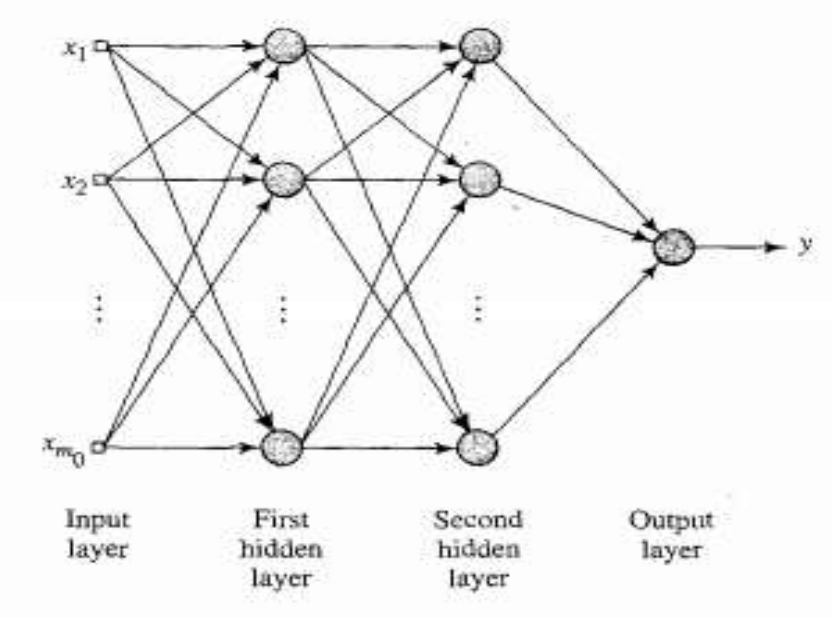
Artificial Neural Network-Perceptron

BPA-Feature detection

- Hidden layer neurons play a important role in the operation of MLP with BPA because they act as a feature detectors.
- As the laerning process progresses, the hidden neurons begin to discover the salient features that characterizes the traing data.
- Performing non linear transformation from the input space to hidden space.
- For example,non linearly separable classes will be easily separated in the hidden space

Artificial Neural Network-Perceptron

Back-propagation and differentiation



The above MLP is parameterised by an architecture A (representing a discrete parameter) and a weight vector W

Artificial Neural Network-Perceptron

Back-propagation and differentiation

Let Alj denote the part of the architecture extending from the input layer to node j in layer (l=1,2,3). Accordingly we may write

$$F(W,X)=\varphi(A_{31})$$

$$\frac{\partial F(W,X)}{\partial w_{3lk}}=\varphi'(A_{3l})\varphi(A_{2k})$$

$$\frac{\partial F(W,X)}{\partial w_{2kj}}=\varphi'(A_{3l})\varphi'(A_{2k})\varphi(A_{1j})w_{3lk}$$

$$\frac{\partial F(W,X)}{\partial w_{1ji}}=\varphi'(A_{3l})\varphi'(A_{1j})X_j\left[\sum_k w_{3lk}\varphi'(A_{2k})w_{2kj}\right]$$

Back-propagation and differentiation

The sensitivity of F(w,x) is

$$S_w^F = \frac{\partial F / F}{\partial w / w}$$

Home Work:

1. Jacobian Matrix
2. Hessian Matrix

Artificial Neural Network-Percepton

Multi-Layer Perceptron: Generalization

- A network is said to be generalised when the input-ouput mapping computed by the network is correct for test data not used in the training.
- To understand this concept, let us assume NN is performing the non-linear input-output mapping: Curve fitting problem. This is nothing but interpolation problem
- Training data helps in learning process.
- when a test data is presented to NN, network computes the ouput.
- When a network learns to many input-output samples the network may be end up with memorizing the training set- over fitting/over trained

Artificial Neural Network-Percepton

Multi-Layer Perceptron: Generalization

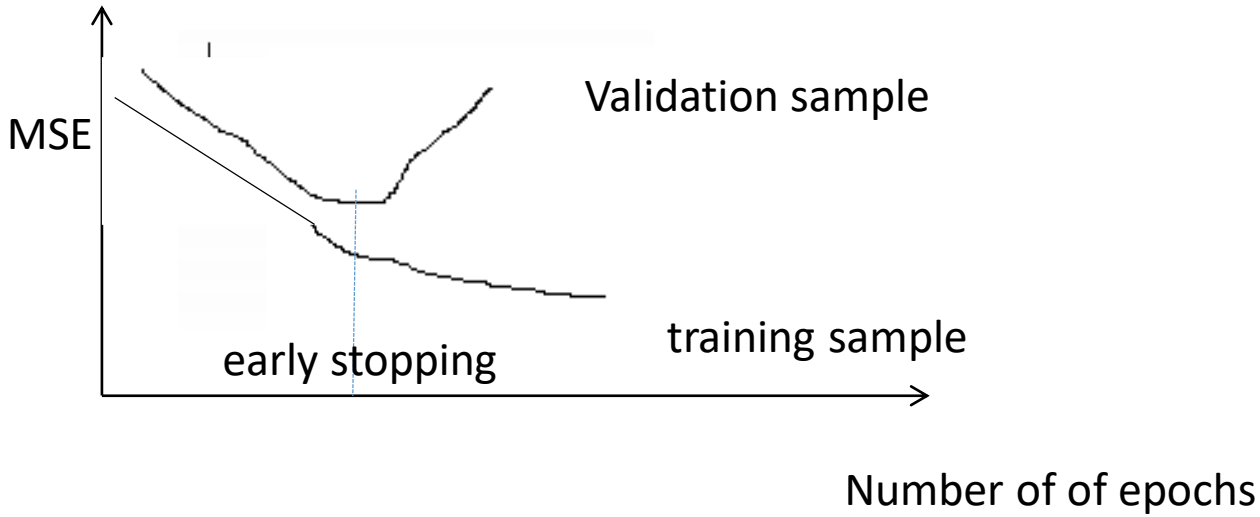
- When the network is over trained, it looses the ability of generalize.
- Generalization is effected by following factors:
  - Size of the training data and how close to the interest
  - Architecture of the NN
  - Complexity of the problem
- For good generalization the size of the training set N is given by N=O(W/e)
- where e is error permitted on the test data and O(.) is the order of the quantity enclosed within.

Multi-Layer Perceptron: Cross Validation

- The available input data is randomly divided into 2 sets: training set and test set.
- The training set is further divided into 2 disjoint sets:
- Estimation subset used to select the model
- Validation subset used to validate the model
- Cross-validation is useful in desigining a large NN with good gerneralisation properties.

Artificial Neural Network-Percepton

Multi-Layer Perceptron: Cross Validation



Multi-Layer Perceptron: Network Pruning Methd

- Choosing a suitable architecture for a NN for a given problem is very tedious job.
- Over sized topology has the following drawbacks:
  - high demand on the computational resource
  - increase in training time
  - non convergence of parameter
  - decrease in the generalization capability
  - shoots up the cost of hardware
  - less eficient.

Multi-Layer Perceptron: Network Pruning Methd

- We can overcome these problems in following ways:
- Growing approach.
- Pruning Method
- Pruning method:
  - Deletion
  - Regularization
- Hessian based Network Pruning:
- The basic idea is to take the 2nd derivative of error susrface

Multi-Layer Perceptron: Network Pruning Methd

- First we construct a local model of error surface by predicting the effect of perturbation in W
- Consider

$$Eav(w+\Delta w) = Eav(w) + \frac{\partial Eav}{\partial w}\frac{\Delta w}{1!} + \Delta w^T \frac{\partial^2 Eav}{\partial w^2}\frac{\Delta w}{2!} + H.O.T$$

$$Eav(w+\Delta w) = Eav(w) + \frac{\partial Eav}{\partial w}\frac{\Delta w}{1!} + \Delta w^T H \frac{\Delta w}{2!} + H.O.T$$

- Next we need to identify the set parameters whose deletion from the network will cause least increase in the Eav.

Multi-Layer Perceptron: Network Pruning Methd

- Assumptions:
- Set 2nd term in the equation to 0 after training
- error surface around the local minimal is highly quadratic.

$$\Delta Eav = Eav(w + \Delta w) - Eav = \frac{1}{2}\Delta w^T H \Delta w$$

Goal: set one of the synapatic weight to zero to minimize the increamental increase in Eav to do so we choose

$$u_i^T \Delta w + w_i = 0$$

Multi-Layer Perceptron: Network Pruning Methd

- minimize the quadratic form with respect to incremental change in the weight vector subject to the constraint that

$$u_i^T \Delta w + w_i = 0$$

$$S = \frac{1}{2} \Delta w^T H \Delta w + \lambda \left(u_i^T \Delta w + w_i \right)$$

$$S_i = \frac{w_i^2}{2\left[H^{-1}\right]_{i,j}}$$

- **Home work:**

1. Virtues and limitation of back-propagation learning

2. BPA as an approximation of functions

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753

ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

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

Artificial Neural Network-Percepton

Supervised Learning view as an optimization problem

- Optimization problems means finding best solution
- Let us assume minimization of quadratic function

$$f(X) = \frac{1}{2} x^T Ax - bx^T + c$$

where x is $m-by-1$ vector

A is $m-by-m$ symmetric positive definite matrix

b is $m-by-1$ vector

c is constant

Supervised Learning view as an optimization problem

$$f'(x) = 0$$

$$Ax^* - b = 0$$

$$x^* = A^{-1}b$$

- Thus minimizing of f(x) and solving linear system of equations are equivalent problems

- Now, for a given matrix A, we say that a set of non-zero vectors [s(0) s(1) ..... s(m-1)] is A-conjugate vectors if thefollowing condition is satisfied

$$s(j)^T A s(j) = 0 \quad \text{for } j = 1, 2, \ldots, m-1$$

Artificial Neural Network-Percepton

Supervised Learning view as an optimization problem

- For a given set of A-conjugate vectors, the correspondin conjugate direction method for unconstrained minimization of the quadratic error function is defined by:

$$x(n+1) = x(n) + \eta(n)s(n)$$

where x(0) is an arbitrary starting vector

$\eta(n)$ is scalar defined by

$$f\left(x(n)+\eta(n)s(n)\right) = \min_{\eta}\left(f\left(x(n)+\eta(n)s(n)\right)\right)$$

Artificial Neural Network-Percepton

Supervised Learning view as an optimization problem

The procedure of choose learning rate so as to minimize the function f(.) for some fixed learning rate is referred as line search

$$\eta(n) = \frac{-S^T(n)AE(n)}{S^T(n)AS(n)}$$

where $x(n) - x^* = E$

$x^* = A^{-1}b$

For conjugate gradient method to work we required A-conjugate vectors, which is computed as follows:

$$S(n) = r(n) + \beta(n)s(n-1)$$

where $\beta(n) = -\frac{S^T(n-1)Ar(n)}{S^T(n-1)AS(n-1)}$

$r(n) = b - Ax(n)$

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753