NEURAL NETWORK LEARNING RULES

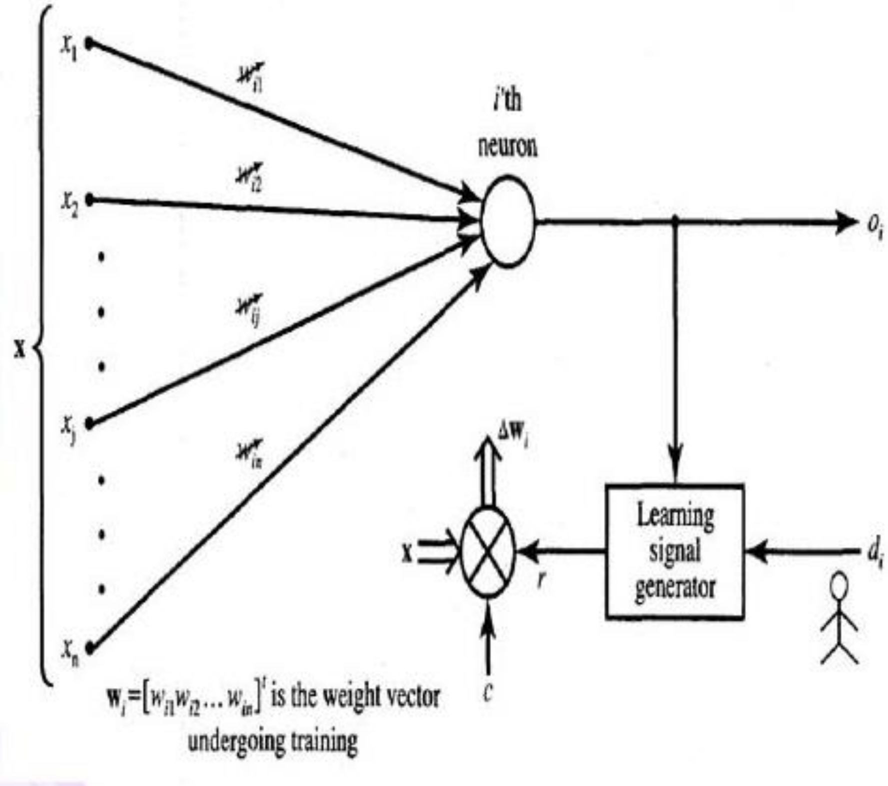
CHAPTER 2


ARTIFICIAL NEURAL NETWORK LEARNING
RULES

Neural Network Learning Rules

We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights. Hence, a method is required with the help of which the weights can be modified. These methods are called Learning rules, which are simply algorithms or equations.

Neural Network Learning Rules



- The learning signal **r** in general a function of **wi**, **x** and sometimes of teacher's signal di.

$$r = r(\mathbf{w}_i, \mathbf{x}, d_i)$$

- Incremental weight vector wi at step t becomes:

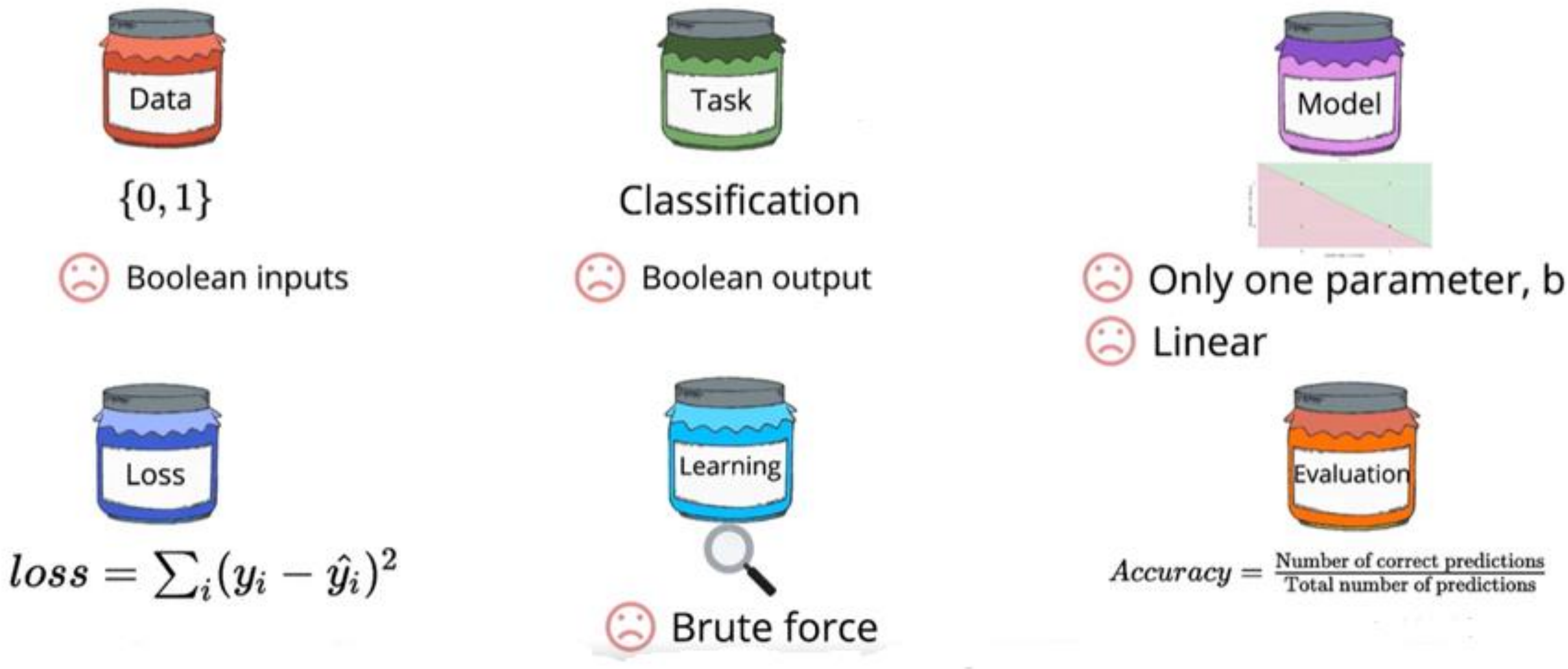
$$\Delta \mathbf{w}_i(t) = cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \mathbf{x}(t)$$

Where c is a learning constant having +ve value.

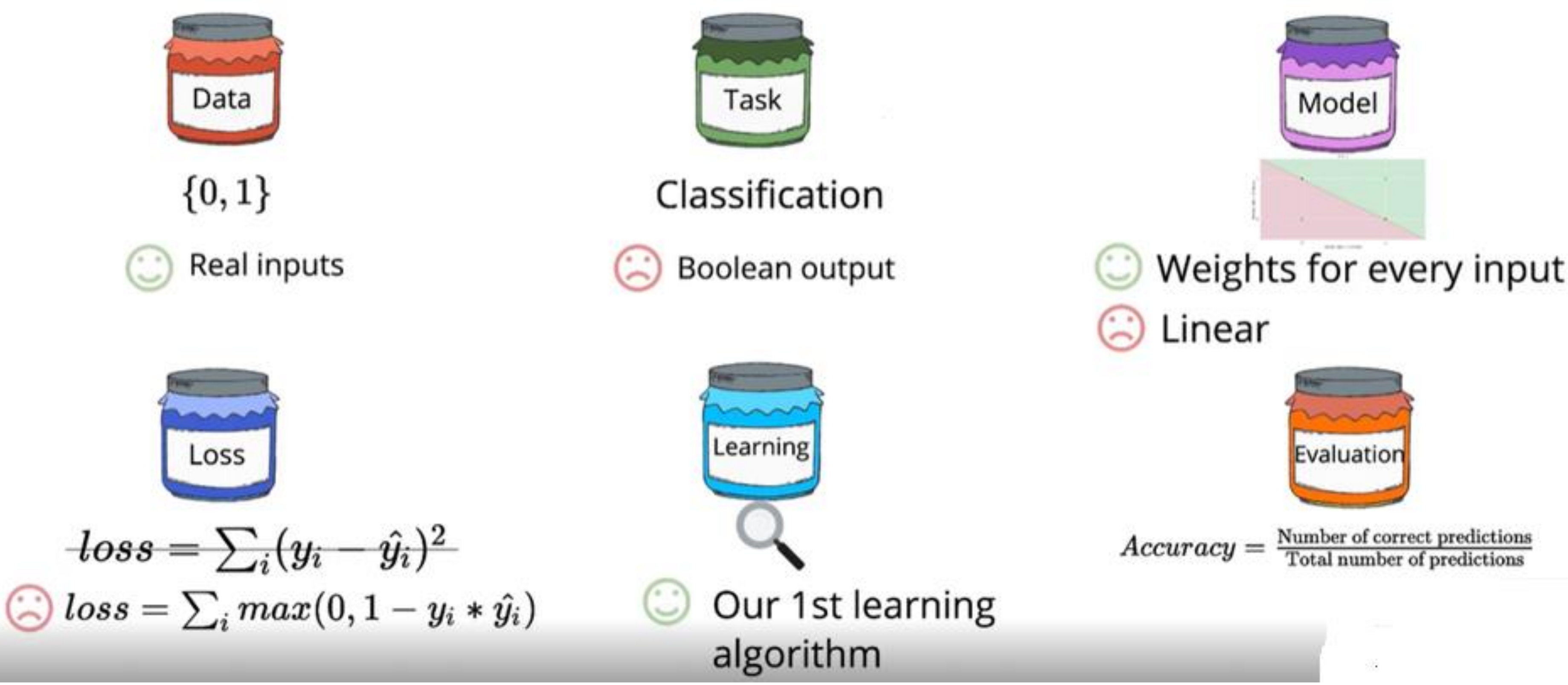
Neural Network Learning Rules

- Perceptron Learning Rule -- Supervised Learning
- Hebbian Learning Rule – Unsupervised Learning
- Delta Learning Rule -- Supervised Learning
- Widrow-Hoffs Learning Rule -- Supervised Learning
- Correlation Learning Rule -- Supervised Learning
- Winner-Take-all Learning Rule -- Unsupervised Learning
- Outstar Learning Rule -- Supervised Learning

MP Neuron


Data
$\{0, 1\}$
Boolean inputs
Task
Classification
Boolean output
Model
Only one parameter, b
Linear
Loss
$loss = \sum_i (y_i - \hat{y}_i)^2$
Learning
Brute force
Evaluation
$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$

Perceptron Learning Rule -- Supervised Learning



Data and Task

									
Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight (g)	151	180	160	205	162	182	138	185	170
Screen size (inches)	5.8	6.18	5.84	6.2	5.9	6.26	4.7	6.41	5.5
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery(mAh)	3060	3500	3060	5000	3000	4000	1960	3700	3260
Price (INR)	15k	32k	25k	18k	14k	12k	35k	42k	44k
Like (y)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

Data Preparation

Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight (g)	151	180	160	205	162	182	138	185	170
Screen size (inches)	5.8	6.18	5.84	6.2	5.9	6.26	4.7	6.41	5.5
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery(mAh)	3060	3500	3060	5000	3000	4000	1960	3700	3260
Price (INR)	15k	32k	25k	18k	14k	12k	35k	42k	44k
Like (*y*)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

screen size
5.8
6.18
5.84
6.2
5.9
6.26
4.7
6.41
5.5

Data Preparation

Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight (g)	151	180	160	205	162	182	138	185	170
Screen size (inches)	5.8	6.18	5.84	6.2	5.9	6.26	4.7	6.41	5.5
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery(mAh)	3060	3500	3060	5000	3000	4000	1960	3700	3260
Price (INR)	15k	32k	25k	18k	14k	12k	35k	42k	44k
Like (*y*)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

Standardization formula

$$x' = \frac{x - min}{max - min}$$

screen size	
5.8	
6.18	
5.84	
6.2	
5.9	
6.26	
$\boxed{4.7}$	min
$\boxed{6.41}$	max
5.5	

Data Preparation

									
Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight	0.19	0.63	0.33	1	0.36	0.66	0	0.70	0.48
Screen size	0.64	0.87	0.67	0.88	0.7	0.91	0	1	0.47
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery	0.36	0.51	0.36	1	0.34	0.67	0	0.57	0.43
Price	0.09	0.63	0.41	0.19	0.06	0	0.72	0.94	1
Like (*y*)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

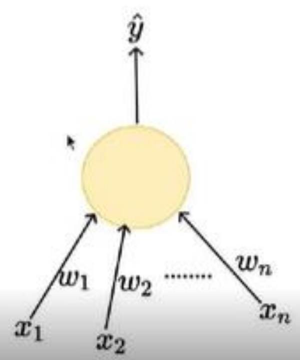
Evaluation

Training data

Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight	0.19	0.63	0.33	1	0.36	0.66	0	0.70	0.48
Screen size	0.64	0.87	0.67	0.88	0.7	0.91	0	1	0.47
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery	0.36	0.51	0.36	1	0.34	0.67	0	0.57	0.43
Price	0.09	0.63	0.41	0.19	0.06	0	0.72	0.94	1
Like (*y*)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

$$\hat{y} = (\sum_{i=1}^{n} w_i x_i \geq b)$$

$$loss = \sum_i \mathbf{1}_{(y_i != \hat{y}_i)}$$



Evaluation

Training data

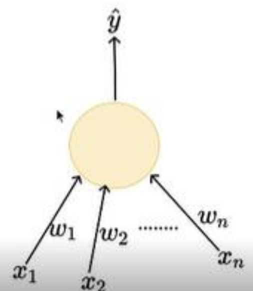
Launch (within 6 months)	0	1	1	0	0	1	0	1	1
Weight	0.19	0.63	0.33	1	0.36	0.66	0	0.70	0.48
Screen size	0.64	0.87	0.67	0.88	0.7	0.91	0	1	0.47
dual sim	1	1	0	0	0	1	0	1	0
Internal memory (>= 64 GB, 4GB RAM)	1	1	1	1	1	1	1	1	1
NFC	0	1	1	0	1	0	1	1	1
Radio	1	0	0	1	1	1	0	0	0
Battery	0.36	0.51	0.36	1	0.34	0.67	0	0.57	0.43
Price	0.09	0.63	0.41	0.19	0.06	0	0.72	0.94	1
Like (*y*)	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**

Test data

1	0	0	1
0.23	0.34	0.44	0.54
0.74	0.93	0.34	0.42
0	1	0	0
1	0	0	0
0	0	1	0
1	1	1	0
1	1	1	0
0	0	1	0
0	1	0	0
0	1	1	0

$$\hat{y} = \left(\sum_{i=1}^{n} w_i x_i \geq b\right)$$

$$loss = \sum_i \mathbf{1}_{(y_i != \hat{y}_i)}$$



Perceptron Learning Rule

For the perceptron learning ~~rule~~ signal is the difference between the desired and actual neuron's response. Thus, learning is supervised and the learning signal is equal to,

$$r = d_i - o_i$$

where $o_i = \text{sgn}(w_i^t x)$ and d_i is the desired response.

weight adjustments in this method, Δw_i and Δw_{ij} are obtained as follows:- $\Delta w_i = \pm 2cx$

$$\boxed{\Delta w_i = c\,[d_i - \underbrace{\text{sgn}(w_i^t x)}_{o_i}]\,x}$$

$$\boxed{\Delta w_{ij} = c\,[d_i - \text{sgn}(w_i^t x)]\,x_j} \quad \text{for } j = 1, 2, \ldots$$

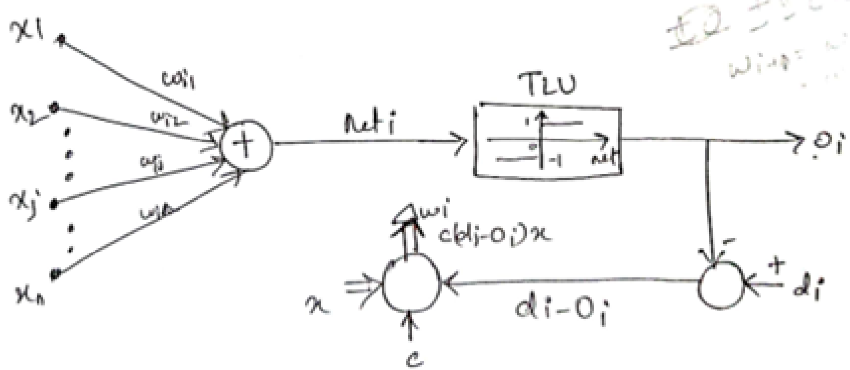
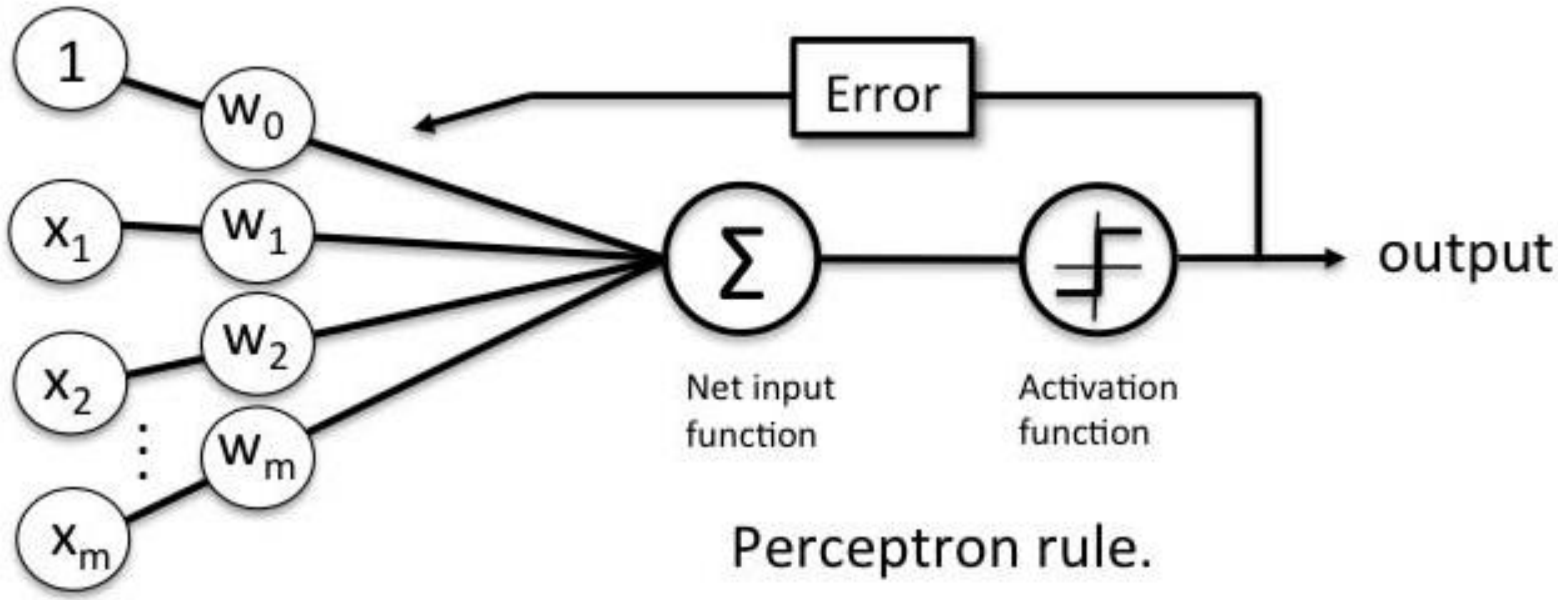


Fig: Perceptron Learning Rule



Perceptron rule.

⇒ This rule is applicable for Binary neuron response. i.e Rule for the binary bipola

⇒ Under this rule weights are adjusted if and only if O_i is incorrect.

⇒ As the desired response is +1 or −1 the weight adjustment (2) reduces to,

$$\boxed{\Delta w_i = \pm 2cx} \quad \text{—— (4)}$$

⇒ where a + is applicable when $d_i = 1$ and $\operatorname{sgn}(w^t x) = -1$
and a minus sign is applicable when $d_i = -1$ and $\operatorname{sgn}(w^t x) = 1$.

⇒ The wt. adjustment formula will not be used when $d_i = \operatorname{sgn}(w_i^t x)$

⇒ This is a very important rule for supervised learning Rule.

⇒ The weight are initialized at any value in this method.

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$

$N \leftarrow inputs \quad with \quad label \quad 0;$

Initialize **w** randomly;

while !*convergence* **do**

  Pick random $\mathbf{x} \in P \cup N$;

  if $\mathbf{x} \in P \quad and \quad \sum_{i=0}^{n} w_i * x_i < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;

  end

  if $\mathbf{x} \in N \quad and \quad \sum_{i=0}^{n} w_i * x_i \geq 0$ **then**

    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;

  end

end

//the algorithm converges when all the inputs are classified correctly

Q: For the given network shown in fig. we the perceptron learning rule to adjust the weight. The set of training vectors are as-

$$x_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}, \quad x_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

Initial weight $w_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$

$c = 0.1$, $d_1 = -1$, $d_2 = -1$ and $d_3 = 1$.

output is d_1: