NEURAL NETWORK LEARNING RULES

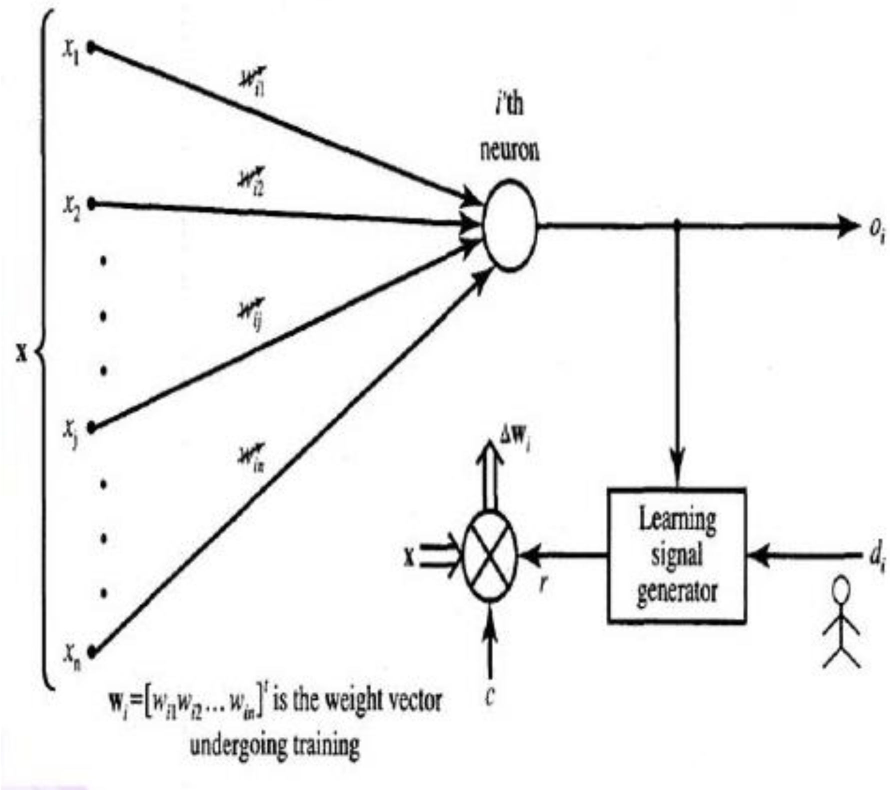
CHAPTER 2


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

DELTA LEARNING RULE

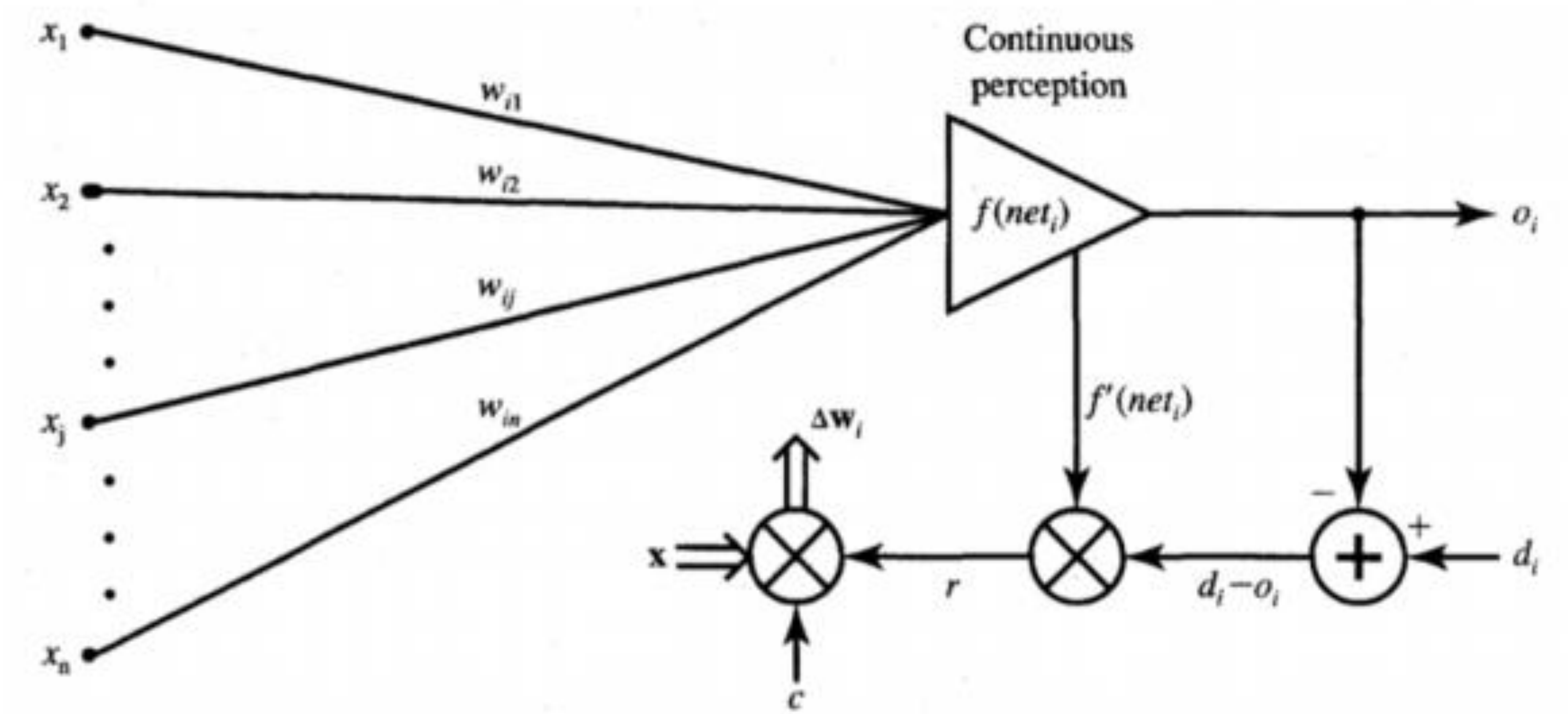
- It depends on supervised learning.
- This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input.
- In Mathematical form the delta rule is as follows:

$$\Delta w = \eta (t - y) x_i$$

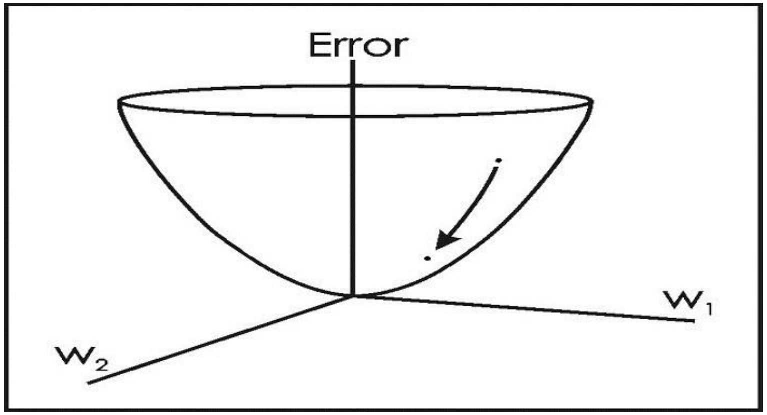
- For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference.
- The change in weight from ui to uj is: dwij = r* ai * ej.

where r is the learning rate, ai represents the activation of ui and ej is the difference between the expected output and the actual output of uj.

DELTA LEARNING RULE



DELTA LEARNING RULE



For any given set of input data and weights, there will be an associated magnitude of error, which is measured by an error function (also known as a cost function). The Delta Rule employs the error function for what is known as Gradient Descent learning, which involves the '*modification of weights along the most direct path in weight-space to minimize error', so* change applied to a given weight is proportional to the negative of the derivative of the error with respect to that weight

$$E_p = \frac{1}{2} \sum_n (t_{j_n} - a_{j_n})^2$$

Delta Learning Rule

The delta learning rule is only valid for continuous activation functions. as

$$f(net) = \frac{2}{1+\exp^{-\lambda net}} - 1 \quad \text{— Bipolar}$$

or

$$= \frac{1}{1+\exp^{-\lambda net}} \quad \text{— Unipolar}$$

The learning signal for this rule is called delta and is defined as follows:-

$$\boxed{r = [d_i - \underbrace{f(w_i^t x)}_{O_i}]\, f'(w_i^t x)} \quad \text{— ①}$$

The term $f'(w_i^t x)$ is the derivative of the activation function $f(net)$ computed for $net = w_i^t x$.

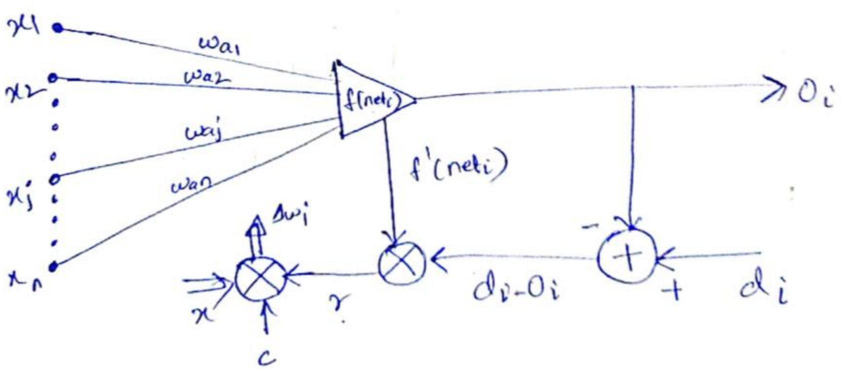


Fig: Delta Learning Rule

This learning rule can be readily derived from the condition of least squared error between o_i and d_i. Calculating the gradient vector with respect to w_i of the squared error defined as,

$$\boxed{E = \frac{1}{2}(d_i - o_i)^2} \quad — (2)$$

or

$$\boxed{E = \frac{1}{2}\left[d_i - f(w_i^t x)\right]^2} \quad — (3)$$

we obtain the error gradient ~~value~~, vector value,

$$\boxed{\nabla E = -(d_i - o_i) f'(w_i^t x) \cdot x} \quad — (4)$$

The components of the gradient vector are,

$$\boxed{\frac{\partial E}{\partial w_{ij}} = -(d_i - o_i) f'(w_i^t x) x_j} \quad \text{for } j = 1, 2, \ldots n \quad — (5)$$

Since, minimization of the error requires the weight changes to be in the negative gradient direction, we take

$$\boxed{\Delta w_i = -\eta \nabla E} \quad — (6)$$

where η is a positive constant

From Eq. (4) + (6) we obtain,

$$\boxed{\Delta w_i = \eta (d_i - o_i) f'(net_i) \cdot x} \quad \text{—— (7)}$$

or, for the single weight the adjustment becomes

$$\boxed{\Delta w_{ij} = \eta (d_i - o_i) f'(net_i) x_j} \quad \text{—— (8)}$$

for $j = 1, 2, \ldots n$

Considering the use of general learning rule and plugging in the learning signal, the weight adjustment becomes,

$$\boxed{\Delta w_i = c (d_i - o_i) f'(net_i) \cdot x} \quad \text{—— (9)}$$

From equations (7), (8) + (9) we can conclude that both are identical as, c + η ~~are~~ have been assumed to be arbitrary constants.

⇒ The weights are initialized at any value for this method of training.

⇒ This rule parallels the discrete perceptron training rule. It can also be called as continuous perceptron training rule. The delta learning rule can be generalized for multi-layer networks.

$$f\,`(net) = ½(d_i^2 – o_i^2)$$

$$f\,`(net) = ½(1 – o_i^2)$$

$$f = \frac{u}{v}$$

$$f' = \frac{v \cdot u' - u \cdot v'}{v^2}$$

$$f = \frac{1 - e^{-net}}{1 + e^{-net}} \quad \left(= \frac{u}{v} \right)$$

$$f' = \frac{(1+e^{-net})\left(-(-e^{-net})\right) - (1-e^{-net})\left(+(-e^{-net})\right)}{(1+e^{-net})^2}$$

$$= \frac{(-e^{-net})\left[(1+e^{-net})(-1) - (1-e^{-net})\right]}{(1+e^{-net})^2}$$

$$= \frac{(-e^{-net})\left[-1 - \cancel{e^{-net}} - 1 + \cancel{e^{-net}}\right]}{(1+e^{-net})^2}$$

$$= \frac{2e^{-net}}{(1+e^{-net})^2}$$

Now multiply ñ divide the resultant with 2,

$$= \frac{1}{2}\left(\frac{2 \times 2e^{-net}}{(1+e^{-net})^2}\right)$$

now,

let $\left[\frac{4e^{-net}}{(1+e^{-net})^2}\right] = p$

$$= \frac{1}{2}(p)$$

$$= \frac{1}{2}(1-1+p) \quad \left[\because \text{To prove that } f' = (1-(f'net)^2)\right.$$

$$= \frac{1}{2}(1-(1-p))$$

$$= \frac{1}{2}\left(1-\left(1-\frac{4e^{-net}}{(1+e^{-net})^2}\right)\right)$$

$$= \frac{1}{2}\left(1-\left[\frac{1+e^{-net^2}+2e^{-net}-4e^{-net}}{(1+e^{-net})^2}\right]\right)$$

$$= \frac{1}{2}\left(1-\left(\frac{1+e^{-net^2}-2e^{-net}}{(1+e^{-net})^2}\right)\right.$$

$$= \frac{1}{2}\left(1-\frac{(1-e^{-net})^2}{(1+e^{-net})^2}\right)$$

$$= \frac{1}{2}\left(1-\left(\frac{1-e^{-net}}{1+e^{-net}}\right)^2\right. = \frac{1}{2}(1-o^2) \quad //$$

Q: For the given network apply the delta learning rule

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

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

$d_1 = -1$, $d_2 = -1$ & $d_3 = 1$

$c = 0.1$, $\lambda = 1$ for the bipolar continous activation function.

Q: For the given network apply the delta learning rule

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

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

$d_1 = -1$, $d_2 = -1$ & $d_3 = 1$
$C = 0.1$, $\lambda = 1$ for the bipolar continous activation function.

Step 1: Input is x_1 vector and initial vector is w_1.

$$net^1 = w^{1t} x_1 = [1 \;\; -1 \;\; 0 \;\; 0.5] \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= 1 + 2 + 0 - 0.5$$
$$= 2.5$$

$$O_1 = f(net^1) = \frac{2}{1 + \exp^{-\lambda net_1}} - 1$$

$$= \frac{2}{1 + \exp^{-2.5}} - 1$$

$$= \frac{2}{1 + 0.082} - 1$$

$$= 1.848 - 1$$

$$= 0.848$$

$$f'(net^1) = \frac{1}{2}\left[d_1^2 - (O_1)^2\right]$$

$$= \frac{1}{2}\left[1 - 0.719104\right]$$

$$= \frac{0.280896}{2} = 0.1404$$

$$w_2 = c\,(d_1 - O_1)\,f'(net^1)\,x_1 + w_1$$

$$= 0.1 * (-1 - 0.848) * 0.1404 * \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

$$= -0.02594 * \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

$$= \begin{bmatrix} -0.02594 \\ 0.05189 \\ 0 \\ +0.02594 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.9740 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix}$$

AMITY UNIVERSITY

of Engineering
nology

Step 2: Input vector is x_2 and weight vector is w_2.

$$X_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.974 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix}$$

$$net^2 = w^{2t} x_2 = [0.974 \ -0.948 \ 0 \ 0.526] \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$$

$$= 0 - 1.422 + 0 + (-0.526)$$

$$= -1.948$$

$$O_2^2 = f(net^2) = \frac{2}{1 + exp^{-\lambda net_2}} - 1$$

$$= -0.75$$

$$f'(net^2) = \frac{1}{2} [d_2^2 - O_2^2]$$

$$= \frac{1}{2} [1 - 0.5625]$$

$$= \frac{1}{2} * 0.4375$$

$$= 0.218$$

$$w^3 = c * (d_2 - O_2) * f'(net^2)\, x_2 + w_2$$

$$= 0.1 * (-1-(-0.75)) * 0.218 * \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix}$$

$$= -0.00545 * \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ -0.008175 \\ 0.002725 \\ 0.00545 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix} = \begin{bmatrix} 0.974 \\ -0.956 \\ 0.002 \\ 0.531 \end{bmatrix}$$

Step 3: Input is x_3 and weight is w_3.

$$net^3 = w^{3t} x_3 = [0.974 \ -0.956 \ 0.002 \ 0.531] \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

$$= -0.974 - 0.956 + 0.001 - 0.531$$

$$= -2.46$$

$$O_3' = f(net^3) = \frac{2}{1 + exp^{-net_3}} - 1$$

$$= \frac{2}{1 + exp^{-(-2.46)}} - 1$$

$$= -0.842$$

$$f'(net^3) = \frac{1}{2}(d_3^2 - O_3^2)$$

$$= \frac{1}{2}(1 - 0.708964)$$

$$= \frac{1}{2} * 0.291036$$

$$= 0.145$$

$$w_4 = c * (d_3 - O_3) * f'(net_3) * x_3 + w_3$$

$$= 0.1 * (1 - (-0.842) * 0.145 * \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.956 \\ 0.002 \\ 0.531 \end{bmatrix}$$

$$= 0.0267 * \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.956 \\ 0.002 \\ 0.531 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0267 \\ 0.0267 \\ 0.0134 \\ -0.0267 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.956 \\ 0.002 \\ 0.531 \end{bmatrix}$$

$$= \begin{bmatrix} 0.9473 \\ -0.9293 \\ 0.0154 \\ 0.5043 \end{bmatrix}$$

inal weight after input x_3

$$w_4 = \begin{bmatrix} 0.9473 \\ -0.9293 \\ 0.0154 \\ 0.5043 \end{bmatrix}$$