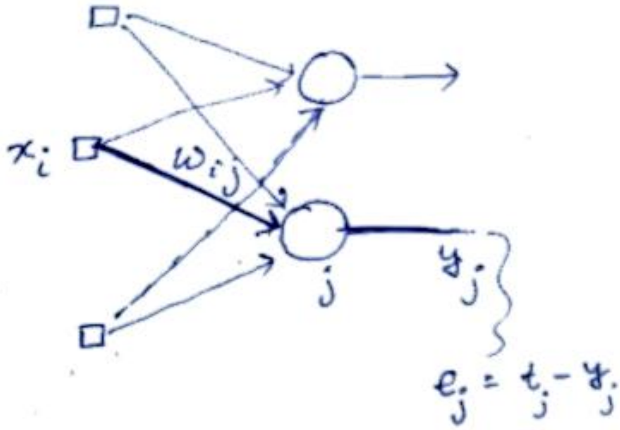
Delta Rule

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$$



$$\Delta w_{ij} = \eta \, e_j \, x_i$$

$$e_j = t - y_{in_j}$$

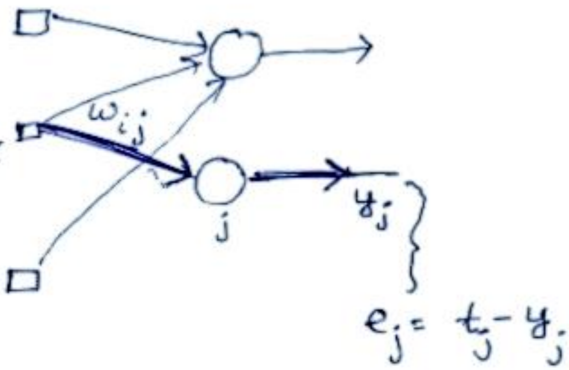
$y_j = y_{in_j}$ (Linear Activation function)

Note#

Delta Rule – uses linear activation function

Generalized Delta Rule – uses sigmoid activation

Generalized Delta Rule



$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$$

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

$$\delta_j = e_j \, \phi'(y_{in_j})$$

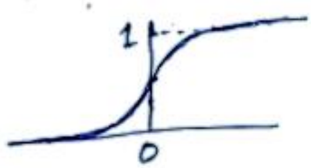
$$\phi'(y_{in_j}) = \frac{d}{dx}(\text{Activation function})$$

In case of sigmoid or logistic activation (Binary)

$$\phi(x) = \frac{1}{1+e^{-\lambda x}} \quad \text{where } \lambda = \text{steepness parameter}$$

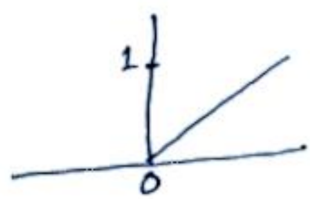
let $\lambda = 1$,

$$\phi(x) = \frac{1}{1+e^{-x}}$$



$$\phi'(x) = \phi(x) \cdot (1 - \phi(x))$$

In case of linear activation



$$\phi(x) = x$$

$$\phi'(x) = \frac{d}{dx}(x) = 1$$

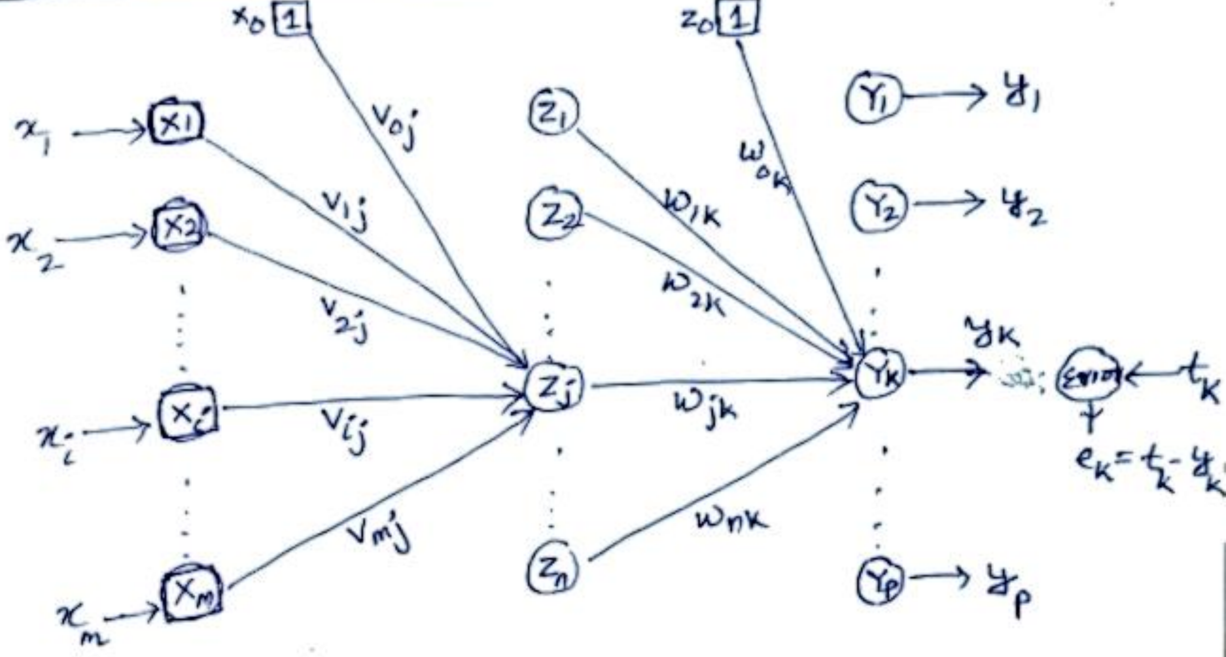
$$\delta_j = e_j \, \phi'(x) = e_j \cdot 1 = e_j \qquad \boxed{\delta_j = e_j}$$

$$\boxed{w_{ij} = w_{ij} + \eta \, e_j \, x_i}$$

Note# When we use linear activation function, the generalized delta rule becomes simple delta rule.

MLP / Back-propagation Network

m – n – p Architecture



Input Layer	Hidden Layer	Output Layer
m neurons	n neurons	p neurons
Linear (Identity) Activation	Binary/Log/Tan-Sigmoid Activation	Binary/log/Tan-Sigmoid Activation

$m - n - p$

Input Hidden output
i j k

Forward pass

Input to Hidden Unit Z_j ($j = 1$ to n)

$$z_{inj} = v_{oj} + \sum_{i=1}^{m} x_i \cdot v_{ij}$$

$$= \sum_{i=0}^{m} x_i \cdot v_{ij} = V^T X = V \cdot X$$

dot product

Output of the Hidden Unit Z_j

Apply activation function over Z_{inj}

$$Z_j = f(Z_{inj})$$

$f()$: Activation function.

Input to output Unit Y_k ($k = 1$ to p)

$$y_{ink} = w_{ok} + \sum_{j=1}^{n} z_j \cdot w_{jk}$$

$$= \sum_{j=0}^{n} z_j \, w_{jk} = W^T Z = W \cdot Z$$

dot product

Output of Output Unit/Layer

Apply activation function.

$$y_k = f(y_{ink})$$

Error associated with output unit Y_K

$$e_k = t_k - y_k$$

t_k: target

y_k: computed output

Squared Error

$$e_k^2 = (t_k - y_k)^2$$

- squared error is minimized by the use of steepest descent / Gradient descent method.

- For easy mathematical derivation, error of k^{th} output neuron can be written as

$$E_k = \frac{1}{2}(t_k - y_k)^2$$

- For one training sample, error associated with output layer

$$E = \sum_{k=1}^{P} E_k = \sum_{k=1}^{P} \frac{1}{2}(t_k - y_k)^2$$

- For "T" training samples, total error in the prediction

$$E_{tot} = \sum_{t=1}^{T}\sum_{k=1}^{P} E_k = \sum_{l=1}^{L}\sum_{k=1}^{P} \frac{1}{2}(t_k - y_k)^2$$

Backward Pass (Back-propagation of error)

Local Gradient / Error correction term / Back-propagated error from the output unit Y_k ($k = 1$ to p)

output layer

$$\delta_k^{(o)} = e_k \, f'(y_{ink})$$

$$= (t_k - y_k)\, \underbrace{f'(y_{ink})}_{\text{derivative of activation function}}$$

change in weights and bias as per the Generalized Delta Rule (steepest descent method)

$$w_{jk}^{new} = w_{jk}^{old} + \Delta w_{jk}$$

where $w_{jk} = \eta \, \delta_k \, z_j$

This δ_k ~~is pr~~ of output layer is propagated to each hidden unit. (backwards)

Weighted sum of δ at each hidden unit Z_j ($j=1$ to n) from the output units Y_k ($k=1$ to p)

$$\delta_{inj} = \sum_{k=1}^{p} \delta_k w_{jk}$$

Local Gradient / propagated Error from hidden unit Z_j

$$\overbrace{\delta_j^{(h)}}^{\text{hidden layer}} = \delta_{inj} \cdot f'(z_{inj})$$

Weight updates

$$v_{ij}^{new} = v_{ij}^{old} + \Delta v_{ij}$$

$$\Delta v_{ij} = \eta\, \delta_j\, x_i$$

Note #: Output Layer weight updates $\boxed{\underset{i}{m} \overset{v}{-} \underset{j}{n} \overset{w}{-} \underset{k}{p}}$

Modified Generalized Delta Rule with momentum constant.

$$w_{jk}^{(t+1)} = w_{jk}^{(t)} + \alpha \, \Delta w_{jk}^{(t-1)} + \Delta w_{jk}^{(t)}$$

$$\Delta w_{jk}^{(t)} = \eta \, \delta_k^{(o)} z_j$$

output layer local gradient

Where

α: momentum constant

η: Learning rate

t: time stamp / iteration

$\eta \uparrow$, higher rate of learning, But unstable. (Oscillatory)

$\eta \downarrow$, Slower rate of learning, But stable.

$\eta \uparrow$ with α, higher rate of Learning + Stable. (Quick convergence + less Oscillation)

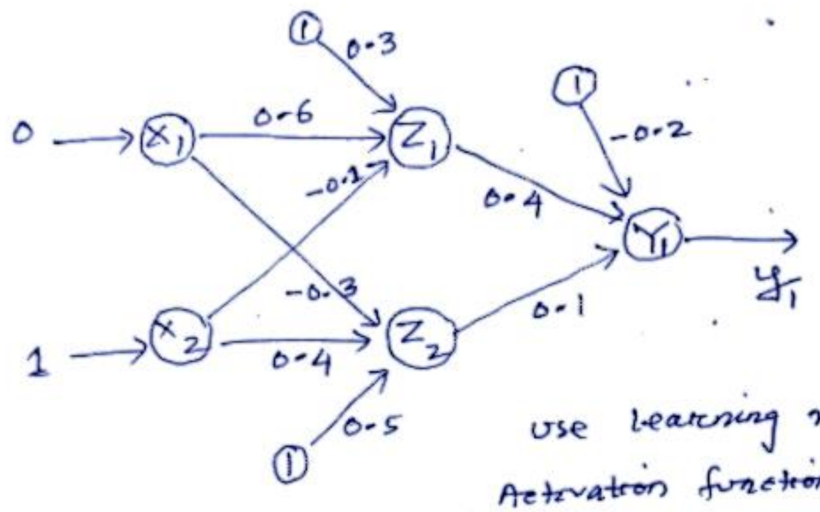
Hidden Layer Weight updates

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + \alpha \, \Delta v_{ij}^{(t-1)} + \Delta v_{ij}^{(t)}$$

$$\Delta v_{ij}^{(t)} = \eta \, \delta_j^{(h)} x_i$$

hidden Layer local gradient

Practice Problem



$t = 1$

use learning rate $\eta = 0.25$

Activation function $f(x)$

$= $ Binary Sigmoid

$= \frac{1}{1+e^{-x}}$

Slope parameter $\lambda = 1$.

Momentum $\alpha = 0$.

Given the training sample

$[x_1, x_2] = [0, 1]$

target $= t = 1$

Forward pass

Hidden Layer calculation

Hidden Layer input

$$Z_{in1} = 1 \cdot v_{01} + x_1 \cdot v_{11} + x_2 \cdot v_{21}$$
$$= 1 \cdot 0.3 + 0 \cdot 0.6 + 1 \times -0.1 = 0.2$$

$$Z_{in2} = 1 \cdot v_{02} + x_1 \cdot v_{12} + x_2 \cdot v_{22}$$
$$= 1 \cdot 0.5 + 0 \times -0.3 + 1 \times 0.4 = 0.9$$

Hidden Layer output

Applying activation function to calculate the output of hidden layer.

$$Z_1 = f(Z_{in1}) = \frac{1}{1+e^{-Z_{in1}}} = \frac{1}{1+e^{-0.2}} = 0.5498$$

$$Z_2 = f(Z_{in2}) = \frac{1}{1+e^{-Z_{in2}}} = \frac{1}{1+e^{-0.9}} = 0.7109$$

Output Layer calculation

output layer input

$$y_{in1} = 1 \cdot w_{01} + z_1 \cdot w_{11} + z_2 w_{21}$$
$$= 1 \cdot -0.2 + 0.5498 \cdot 0.4 + 0.7109 * 0.1 = 0.091$$

output layer output

$$y_1 = f(y_{in1}) = \frac{1}{1+e^{-y_{in1}}} = \frac{1}{1+e^{-0.091}} = 0.5227$$

Squared Error : $(1 - 0.5227)^2$

Backward pass

Output Layer calculation

Local Gradient / Error correction from output layer

$$\delta_1 = (t_1 - y_1) f'(y_{in1})$$
$$= (1 - 0.5227)\, y_1 (1 - y_1)$$
$$= (1 - 0.5227)\, 0.5227\, (1 - 0.5227)$$
$$= 0.1191$$

weight correction between hidden and output layer

$$\Delta w_{01} = \eta \cdot \delta_1 \cdot 1 = 0.25 * 0.1191 * 1 = 0.02978$$

$$\Delta w_{11} = \eta \cdot \delta_1 z_1 = 0.25 * 0.1191 * 0.5498$$
$$= 0.0164$$

$$\Delta w_{21} = \eta \cdot \delta_1 \cdot z_2 = 0.25 * 0.1191 * 0.7109$$
$$= 0.02117$$

updated weight

$$w_{01}(new) = w_{01}(old) + \Delta w_{01} = -0.2 + 0.02978$$
$$= -0.17022$$
$$w_{11}(new) = w_{11}(old) + \Delta w_{11} = 0.4 + 0.0164 = 0.4164$$
$$w_{21}(new) = w_{21}(old) + \Delta w_{21} = 0.1 + 0.02117 = 0.12117$$

Hidden Layer calculation

Local Gradient/ Error propagation from hidden layer ($j = 1$ to 2)

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\delta_{inj} = \sum_{k=1}^{p} \delta_k \cdot \omega_{jk} = \delta_1 \cdot \omega_{j1}$$

Local Gradient/ Error propagation from hidden layer

$$\delta_{in1} = \delta_1 \cdot \omega_{11} = 0.1191 \times 0.4 = 0.04764$$

$$\delta_{in2} = \delta_1 \cdot \omega_{21} = 0.1191 \times 0.1 = 0.01191$$

$$\delta_1 = \delta_{in1} \cdot f'(z_{in1}) = \delta_{in1} \cdot z_1 (1 - z_1)$$

$$= 0.04764 * 0.5498 (1 - 0.5498)$$

$$= \cancel{0.2475}\ 0.0118$$

$$\delta_2 = \delta_{in2} \cdot f'(z_{in2}) = \delta_{in2} \cdot z_2 (1 - z_2)$$

$$= 0.1191 * 0.7109 (1 - 0.7109)$$

$$= 0.00245$$

weight correction between input and hidden layer

$$\Delta v_{01} = \eta \delta_1 \cdot 1 = 0.25 * 0.0118 = 0.00295$$

$$\Delta v_{11} = \eta \delta_1 \cdot x_1 = 0.25 * 0.0118 * 0 = 0$$

$$\Delta v_{21} = \eta \delta_1 \cdot x_2 = 0.25 * 0.0118 * 1 = 0.00295$$

$$\Delta v_{02} = \eta \delta_2 \cdot 1 = 0.25 * 0.00245 * 1 = 0.0006125$$

$$\Delta v_{12} = \eta \delta_2 \cdot x_1 = 0.25 * 0.00245 * 0 = 0$$

$$\Delta v_{22} = \eta \delta_2 \cdot x_2 = 0.25 * 0.00245 * 1 = 0.0006125$$

Updated weight

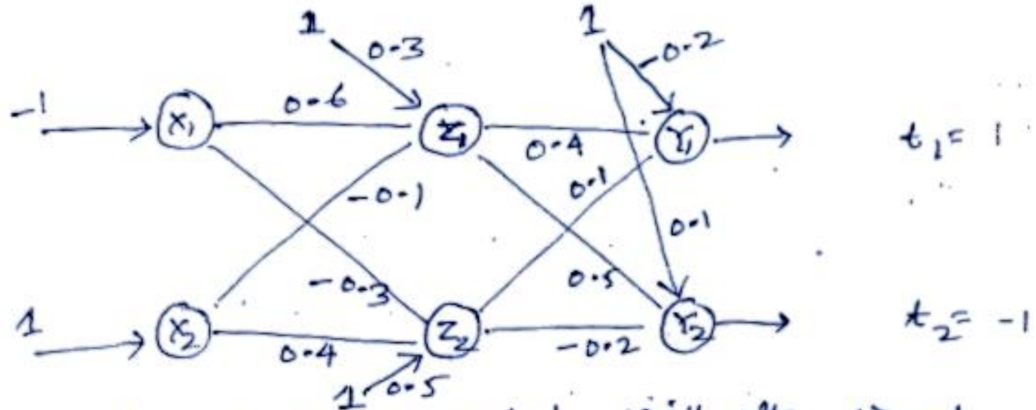
$$v_{01}(new) = v_{01}(old) + \Delta v_{01} = 0.3 + 0.00295 = 0.30295$$

$$v_{11}(new) = v_{11}(old) + \Delta v_{11} = 0.6 + 0 = 0.6$$

$$\vdots$$

Backpropagation practice problem

Find the new weights, using back-propagation network shown below



The network is presented with the input pattern [-1, 1] and targets [+1, -1]. use learning rate 0·25 and ~~bipla bipto~~ bipolar sigmoidal activation for both hidden and output layer.

Activation function (Bipolar sigmoid) $f(x) = \dfrac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$.

Let $\lambda = 1$ (slope param)

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}}$$

Derivative of $f(x)$, $f'(x) = \frac{\lambda}{2}(1+f(x))(1-f(x))$
$= 0.5\,(1+f(x))(1-f(x))$

Given the input sample $[x_1, x_2] = [-1, 1]$ and targets $[t_1, t_2] = [1, -1]$.

Forward pass

Hidden layer calculation

Input to Z_1 neuron,

$$Z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + (-1)\cdot(0.6) + 1\times -0.1 = -0.4$$

Input to Z_2;

$$Z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + (-1)\cdot(-0.3) + 1 \times 0.4 = 1.2$$

output of Z_1

$$Z_1 = f(Z_{in1}) = f(-0.4) = \frac{1-e^{0.4}}{1+e^{0.4}} = -0.1974$$

output of Z_2

$$Z_2 = f(Z_{in2}) = f(1.2) = \frac{1-e^{-1.2}}{1+e^{-1.2}} = 0.537$$

In vector Notation:

$$X \underbrace{\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}}_{\text{bias}} \quad V \begin{bmatrix} 0.3 & 0.5 \\ 0.6 & -0.3 \\ -0.1 & 0.4 \end{bmatrix}_{3\times 2}$$

Input to hidden layer $(V^T X)$

$$\begin{bmatrix} 0.3 & 0.6 & -0.1 \\ 0.5 & -0.3 & 0.4 \end{bmatrix}_{2\times 3} * \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}_{3\times 1} = \begin{bmatrix} -0.4 \\ 1.2 \end{bmatrix}$$

output of hidden layer

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} f(-0.4) \\ f(1.2) \end{bmatrix} = \begin{bmatrix} \cancel{-0.1974} \\ \cancel{0.537} \end{bmatrix} \begin{bmatrix} -0.1974 \\ 0.537 \end{bmatrix}$$

Ans: Forward pass

Output layer calculation

Input to Y_1

$$y_{in1} = w_{01} + z_1 \cdot w_{11} + z_2 \cdot w_{21}$$
$$= -0.2 + (-0.1974) * 0.4 + 0.537 * 0.1$$
$$= -0.22526$$

Input to Y_2

$$y_{in2} = w_{02} + z_1 \cdot w_{12} + z_2 \cdot w_{22}$$
$$= 0.1 + (-0.1974) \cdot (0.5) + (0.537) \cdot (-0.2)$$
$$= -0.1061$$

Output of Y_1

$$y_1 = f(y_{in1}) = f(-0.22526) = -0.1122$$

Output of Y_2

$$y_2 = f(y_{in2}) = f(-0.1061) = -0.053$$

Backward pass

Local gradient of output layer

$$\delta_1^{(o)} = (t_1 - y_1)\, f'(y_{in1})$$ (output layer)

$$= (1 - y_1)\, 0.5\, (1 + y_1)(1 - y_1)$$

$$= 0.5491$$

$$\delta_2^{(o)} = (t_2 - y_2)\, f'(y_{in2})$$

$$= (-1 - y_2)\, 0.5\, (1 + y_2)(1 - y_2)$$

$$= -0.4728$$

Change in weights between hidden and output layer

$$\Delta b = \Delta w_{01} = \eta\, \delta_1^{(o)} \cdot 1 = 0.25 * 0.5491 * 1 = 0.1373$$

$$\Delta w_{11} = \eta\, \delta_1^{(o)} \cdot z_1 = 0.25 * 0.5491 * -0.1974$$
$$= -0.0271$$

$$\Delta w_{21} = \eta\, \delta_1^{(o)} z_2 = 0.25 * 0.5491 * 0.537$$
$$= 0.0737$$

$$\Delta b = \Delta w_{02} = \eta\, \delta_2^{(o)} \cdot 1 = 0.25 * -0.4728 * 1 = -0.11804$$

$$\Delta w_{12} = \eta\, \delta_2^{(o)} z_1 = 0.25 * -0.4728 * -0.1974$$
$$= 0.0233$$

$$\Delta w_{22} = \eta\, \delta_2^{(o)} z_2 = 0.25 * -0.4728 * 0.537$$
$$= -0.0634$$

Backward Pass

Local gradient of hidden layer

for neuron z_1

$$\delta_{in1} = \delta_1^{(o)} \cdot w_{11} + \delta_2^{(o)} \cdot w_{12}$$

$$= 0.5491 * 0.4 + -0.4728 * 0.5$$

$$= -0.0165$$

for neuron z_2

$$\delta_{in2} = \delta_1^{(o)} \cdot w_{21} + \delta_2^{(o)} \cdot w_{22}$$

$$= 0.5491 * 0.1 + -0.4728 * -0.2$$

$$= 0.1493$$

hidden layer

$$\delta_1^{(h)} = \delta_{in1} \cdot f'(z_{in1}) = \cancel{\delta_{in1} \cdot f'(z_{in1})}$$

$$= \overset{-0.0165}{\cancel{0.21964}} \cdot 0.5\ (1+z_1)(1-z_1)$$

$$= \overset{-0.0165}{\cancel{0.21964}} * 0.5\ (1+(-0.1974))(1-(-0.1974))$$

$$= -0.0079$$

$$\delta_2^{(h)} = \delta_{in2} \cdot f'(z_{in2})$$

$$= 0.1493 \cdot 0.5 \cdot (1+z_2)(1-z_2)$$

$$= 0.1493 * 0.5 * (1+0.537)(1-0.537)$$

$$= 0.053134$$

Change in weights between input and hidden layer:

$$\Delta bias = \Delta v_{01} = \eta \cdot \delta_1^{(h)} \cdot 1 = 0.25 * -0.0079 * 1$$
$$= -0.00198$$

$$\Delta v_{11} = \eta\, \delta_1^{(h)} \cdot x_1 = 0.25 * -0.0079 * -1$$
$$= 0.01328$$

$$\Delta v_{21} = \eta\, \delta_1^{(h)} \cdot x_2 = 0.025 * -0.0079 * 1$$
$$= -0.00198$$

$$\Delta bias = \Delta v_{02} = \eta \cdot \delta_2^{(h)} \cdot 1 = 0.25 * 0.053134 * 1$$
$$= 0.01328$$

$$\Delta v_{12} = \eta \cdot \delta_2^{(h)} \cdot x_1 = 0.25 * 0.053134 * -1$$
$$= -0.01328$$

$$\Delta v_{22} = \eta \cdot \delta_2^{(h)} \cdot x_2 = 0.25 * 0.053134 * 1$$
$$= 0.013284$$

Final weights can be computed as below

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}.$$

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$