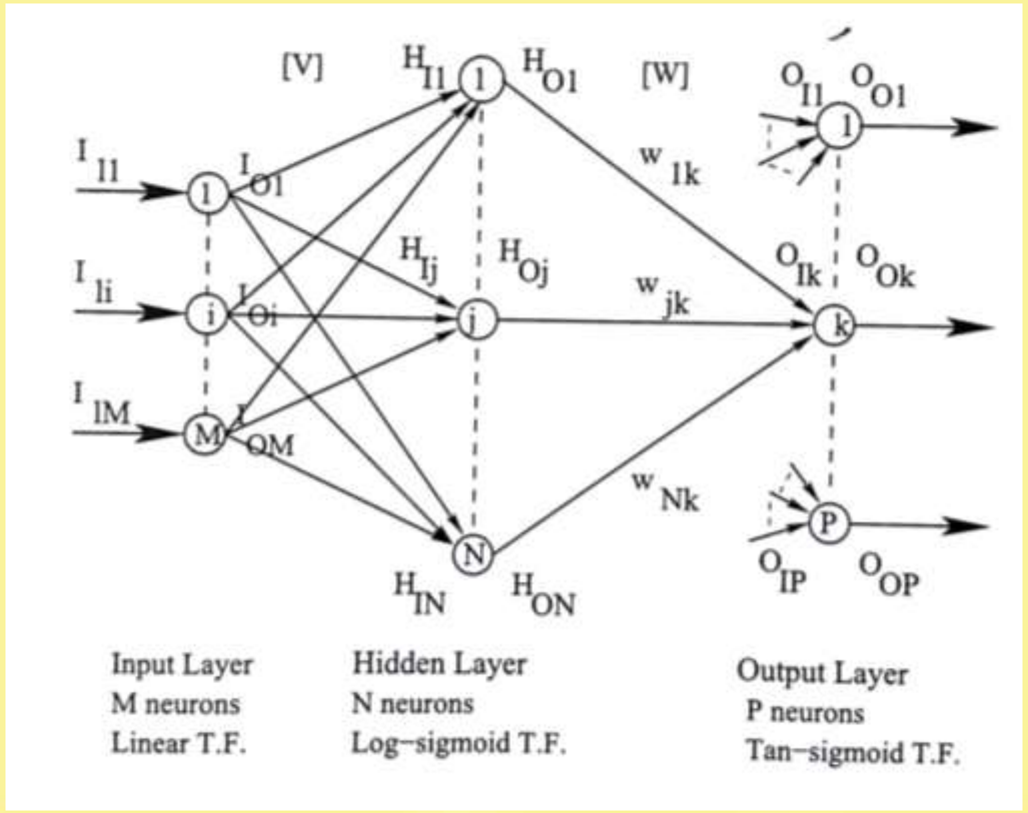
Multi-Layer Feed-Forward Neural Network (MLFFNN)



I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

$$[V] = \begin{bmatrix} V_{11} \cdots\cdots V_{1j} \cdots\cdots V_{1N} \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ V_{i1} \cdots\cdots V_{ij} \cdots\cdots V_{iN} \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ V_{M1} \cdots\cdots V_{Mj} \cdots\cdots V_{MN} \end{bmatrix}$$

$$[W] = \begin{bmatrix} W_{11} \cdots\cdots W_{1K} \cdots\cdots W_{1P} \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ W_{j1} \cdots\cdots W_{jK} \cdots\cdots W_{jP} \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ W_{N1} \cdots\cdots W_{Nk} \cdots\cdots W_{NP} \end{bmatrix}$$

Forward Calculations

- **Step 1:** Determination of the outputs of input layer

  $I_{Oi} = I_{Ii}$,

  where i = 1, 2, ..., M

- **Step 2:** Calculation of inputs of hidden layer

  $H_{Ij} = v_{1j} I_{O1} + \ldots\ldots + v_{ij} I_{Oi} + \ldots\ldots + v_{Mj} I_{OM}$,

  where j = 1, 2, ... , N

  Determination of the outputs of hidden neurons

Step 3: Determination of the outputs of hidden neurons

$$H_{O_j} = \frac{1}{1 + e^{-a_1 H_{Ij}}}$$

where a_1 is the coefficients of TF

Step 4: Determination of the inputs of output layer

$O_{Ik} = w_{1k} H_{O1} + \ldots\ldots + w_{jK} H_{Oj} + \ldots\ldots + w_{NK} H_{ON}$,

where k= 1, 2, ... , P

Step 5: Estimation of the outputs of Output layer O

$$O_{Ok} = \frac{e^{a_2 O_{Ik}} - e^{-a_2 O_{Ik}}}{e^{a_2 O_{Ik}} + e^{-a_2 O_{Ik}}}$$

where a_2 is the coefficient of TF

Step 6: Determination of error in prediction

Error in prediction at k-th output neuron

$$E_k = \frac{1}{2}(T_{Ok} - O_{Ok})^2$$

Error in prediction considering all output neurons

$$E = \sum_{k=1}^{P} \frac{1}{2} (T_{Ok} - O_{Ok})^{2}$$

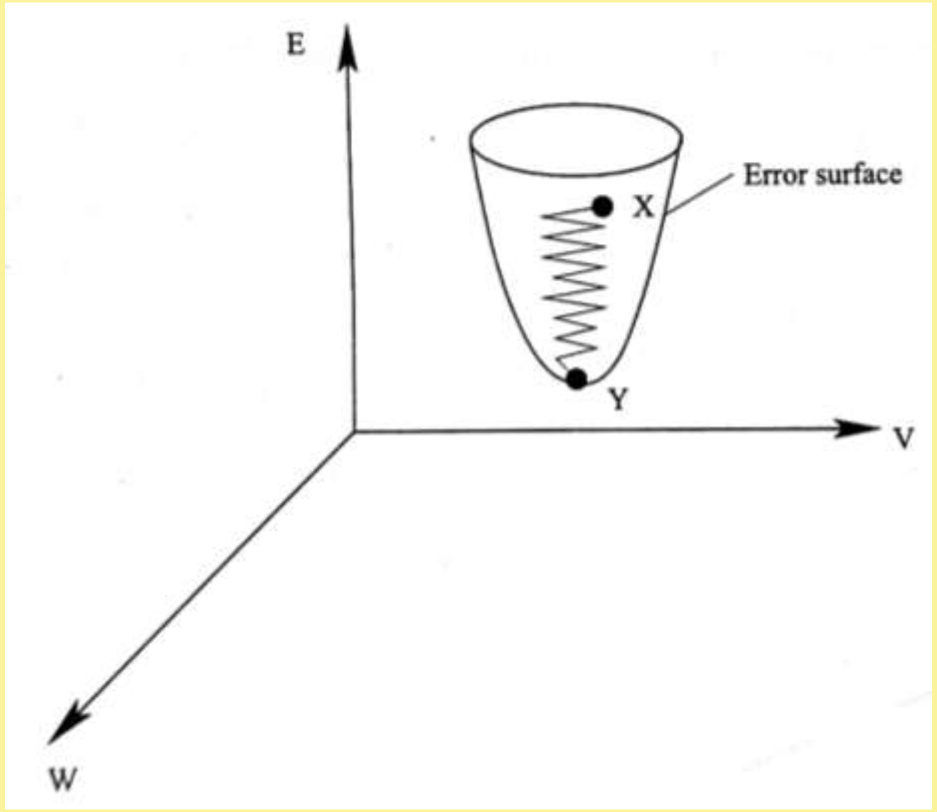
where P: No. of output neurons

Total error in prediction considering all L training scenarios

$$E_{total} = \sum_{l=1}^{L} \sum_{k=1}^{P} \frac{1}{2} (T_{Okl} - O_{Okl})^{2}$$

where L: No. of training scenarios

Back-Propagation Algorithm



- **Delta Rule**

$$E = f(V, W)$$

$$\Delta V = -\eta . \frac{\partial E}{\partial V}$$

$$\Delta W = -\eta . \frac{\partial E}{\partial W}$$

Incremental Mode of Training

Updating of [W]

$$w_{jk,\ \text{updated}} = w_{jk,\ \text{previous}} + \Delta w_{jk}$$

where $\Delta w_{jk} = -\eta . \frac{\partial E_k}{\partial W_{jk}}$

Now, $\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial O_{Ok}} . \frac{\partial O_{Ok}}{\partial O_{Ik}} . \frac{\partial O_{Ik}}{\partial w_{jk}}$

$$\frac{\partial E_k}{\partial O_{Ok}} = -(T_{Ok} - O_{Ok})$$

$$\frac{\partial O_{Ok}}{\partial O_{Ik}} = a_2(1 + O_{Ok})(1 - O_{Ok})$$

$$\frac{\partial O_{Ik}}{\partial w_{jk}} = H_{Oj}$$

$$\frac{\partial E_k}{\partial w_{jk}} = -\,(T_{Ok} - O_{Ok})a_2(1 + O_{Ok})(1 - O_{Ok})H_{Oj}$$

$$\Delta w_{jk} = \eta a_2(T_{Ok} - O_{Ok})(1 + O_{Ok})(1 - O_{Ok})H_{Oj}$$

To update the connecting weight v_{ij} between i-th neuron of input layer and j-th neuron of hidden layer, that is,

$$v_{ij,\ \text{updated}} = v_{ij,\ \text{previous}} + \Delta v_{ij}$$

where $\Delta V_{ij} = -\eta \left\{\frac{\partial E}{\partial V_{ij}}\right\}_{av}$

where $\left\{\frac{\partial E}{\partial V_{ij}}\right\}_{av} = \frac{1}{p}\sum_{k=1}^{p} \frac{\partial E_k}{\partial V_{ij}}$

Now, $$\frac{\partial E_k}{\partial v_{ij}} = \frac{\partial E_k}{\partial O_{Ok}} \cdot \frac{\partial O_{Ok}}{\partial O_{Ik}} \cdot \frac{\partial O_{Ik}}{\partial H_{Oj}} \cdot \frac{\partial H_{Oj}}{\partial H_{Ij}} \cdot \frac{\partial H_{Ij}}{\partial v_{ij}}$$

where, $$\frac{\partial E_k}{\partial O_{Ok}} = -(T_{Ok} - O_{Ok})$$

$$\frac{\partial O_{Ok}}{\partial O_{Ik}} = a_2(1 + O_{Ok})(1 - O_{Ok})$$

$$\frac{\partial O_{Ik}}{\partial H_{Oj}} = w_{jk}$$

$$\frac{\partial H_{Oj}}{\partial H_{Ij}} = a_1 H_{Oj}(1 - H_{Oj})$$

$$\frac{\partial H_{Ij}}{\partial v_{ij}} = I_{Oi} = I_{Ii}$$

We get

$$\frac{\partial E_k}{\partial v_{ij}} = -a_1 a_2 (T_{Ok} - O_{Ok})(1 + O_{Ok})(1 - O_{Ok})(1 - H_{Oj}) w_{jk} H_{Oj} I_{Ii}$$

Batch Mode of Training:

Let us consider L training scenarios. Mean Squared Deviation in prediction for k-th output neuron

$$E' = \frac{1}{2}.\frac{1}{L}\sum_{l=1}^{L}(T_{Okl} - O_{Okl})^2$$

The change in w_{jk}, that is , Δw_{jk} is determined as follows:

$$\Delta w_{jk} = -\eta.\frac{\partial E'}{\partial w_{jk}}$$

Now,

$$\frac{\partial E'}{\partial w_{jk}} = \frac{\partial E'}{\partial E_l}.\frac{\partial E_l}{\partial E_k}.\frac{\partial E_k}{\partial O_{Ok}}.\frac{\partial O_{Ok}}{\partial O_{Ik}}.\frac{\partial O_{Ik}}{\partial w_{jk}}$$

Similarly, Δv$_{ij}$ can be calculated as follows: $\Delta v_{ij} = -\eta . \left\{ \frac{\partial E'}{\partial v_{ij}} \right\}_{av}$

Where $\left\{ \frac{\partial E'}{\partial v_{ij}} \right\}_{av} = \frac{1}{p} \sum_{k=1}^{p} \frac{\partial E'_k}{\partial v_{ij}}$

Now,

$$\frac{\partial E'_k}{\partial v_{ij}} = \frac{\partial E'_k}{\partial E_{kl}} . \frac{\partial E_{kl}}{\partial O_{Ok}} . \frac{\partial O_{Ok}}{\partial O_{Ik}} . \frac{\partial O_{Ik}}{\partial H_{Oj}} . \frac{\partial H_{Oj}}{\partial H_{Ij}} . \frac{\partial H_{Ij}}{\partial v_{ij}}$$

Momentum Constant (α’)

Generalized Delta Rule:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w}(t) + \alpha' \Delta w(t-1)$$

η: (0.0 to 1.0)
α': (0.0 to 1.0)

$\alpha^{'}$ is used to ensure a stable network even at a higher value of learning rate η

Notes:

- BP algorithm → there is a chance of local minima problem
- BP algorithm → transfer functions are to be differentiable in nature
- BPNN may not be able to capture the dynamics of a highly dynamic process
- Inputs are normalized
- Connecting weights lie in the range of either (0.0, 1.0) or (-1.0,1.0)
- Convergence Criterion: Absolute value of the rate of change of error in prediction becomes less than or equal to a pre-specified value
- A neural network can be either fully-connected or partially-connected one

Advantages of BPNN

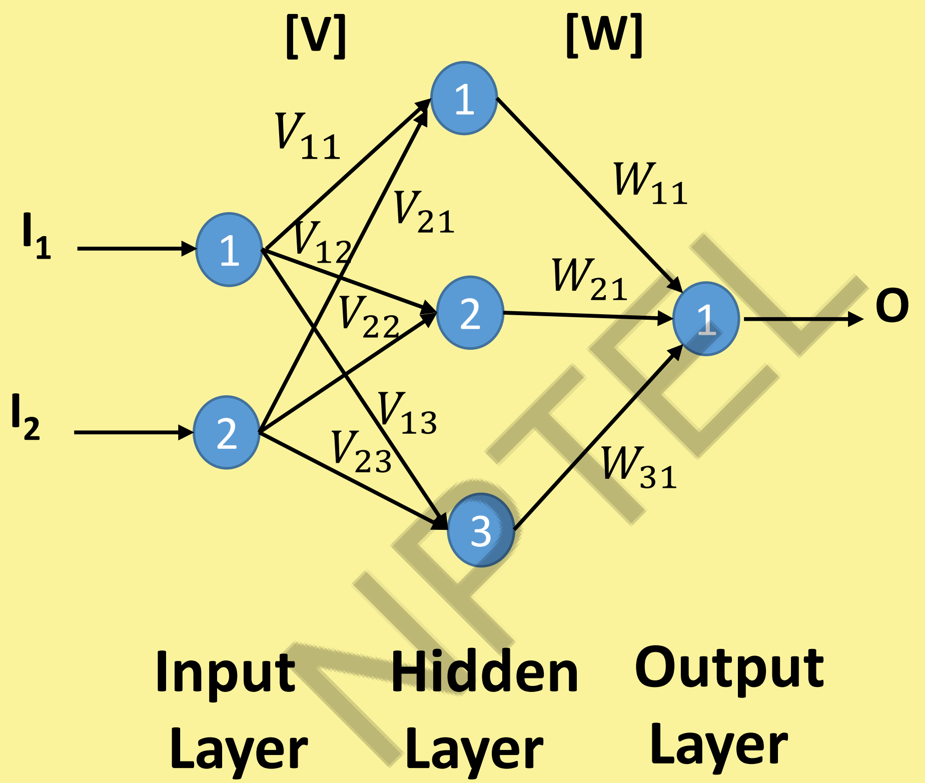
- Can handle a problem having many variables
- May not require so much problem information as that is needed for an FLC

Disadvantages of BPNN

- Solutions of BPNN may get stuck at the local minima
- Training of NN is more involved computationally compared to that of FLC
- It works like a black box

Numerical Example

Figure below shows the schematic view of an NN consisting of three layers, such as input, hidden and output layers. The neuron lying on the input, hidden and output layers have the transfer function represented by $y = x, y = \frac{1}{1+e^{-x}}, y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$, respectively. There are two inputs, namely I_1 and I_2 and one output, that is, O. The connecting weights between the input and hidden layers are represented by [V] and those between hidden and output layers are denoted by [W]. The initial values of the weights are assumed to be as follows:



[V]
[W]
I_1
I_2
V_{11}
V_{12}
V_{13}
V_{21}
V_{22}
V_{23}
W_{11}
W_{21}
W_{31}
O
Input Layer
Hidden Layer
Output Layer

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.4 & 0.3 \\ 0.1 & 0.6 & 0.5 \end{bmatrix};$$

$$\begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Sl. No.	I_1	I_2	T_O
1	0.5	-0.4	0.15
2	-	-	-
.	.	.	.
.	.	.	.
.	.	.	.

Using an incremental mode of training for the case shown in the Table, calculate the changes in V (that is, ΔV) and W (that is ΔW) values during back-propagation of error, the learning rate η is assumed to be equal to 0.2. Show only one iteration.

Solution:

Forward Calculations:

The neurons of input layer have linear transfer function (y=x). Therefore, the output of each input layer neuron is made equal to its corresponding input value.

$$I_{O1} = I_{I1} = 0.5$$
$$I_{O2} = I_{I2} = -0.4$$

The inputs of different neurons of the hidden layer are calculated as follows:

$$H_{I1} = I_{O1}v_{11} + I_{O2}v_{21} = 0.06$$
$$H_{I2} = I_{O1}v_{12} + I_{O2}v_{22} = -0.04$$
$$H_{I3} = I_{O1}v_{13} + I_{O2}v_{23} = -0.05$$

The neurons of the hidden layer are assumed to have log-sigmoid transfer function ($y = \frac{1}{1+e^{-x}}$). The outputs of different hidden neurons are determined like the following:

$$H_{O1} = \frac{1}{1 + e^{-H_{I1}}} = 0.514995$$
$$H_{O2} = \frac{1}{1 + e^{-H_{I2}}} = 0.490001$$
$$H_{O3} = \frac{1}{1 + e^{-H_{I3}}} = 0.487503$$

Now, the input of the output neuron can be calculated as follows:

$$O_{I1} = H_{O1}w_{11} + H_{O2}w_{21} + H_{O3}w_{31} = 0.198249$$

As the output neuron has a tan-sigmoid transfer function, its output can be determined like the following:

$$O_{O1} = \frac{e^{O_{I1}} - e^{-O_{I1}}}{e^{O_{I1}} + e^{-O_{I1}}} = 0.195692$$

The squared error in prediction is found to be as follows:

$$E = \frac{1}{2}(T_O - O_{O1})^2 = 0.001044$$

Back-propagation Algorithm:

The change in w_{11} can be determined using the procedure below.

$$\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}},$$

where $$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial O_{O1}} \frac{\partial O_{O1}}{\partial O_{I1}} \frac{\partial O_{I1}}{\partial w_{11}}.$$

Now, $$\frac{\partial E}{\partial O_{O1}} = -(T_O - O_{O1})$$

$$\frac{\partial O_{O1}}{\partial O_{I1}} = \frac{4}{(e^{O_{I1}} + e^{-O_{I1}})^2}$$

$$\frac{\partial O_{I1}}{\partial w_{11}} = H_{O1}.$$

Substituting the values of $T_O, O_{O1}, O_{I1}, H_{O1}$ in the last expression of $\frac{\partial E}{\partial w_{11}}$ we get

$$\frac{\partial E}{\partial w_{11}} = 0.022630$$

Now, substituting the values of $\frac{\partial E}{\partial w_{11}}$ and η in the expression of Δw_{11}, we get

$$\Delta w_{11} = -0.004526$$

Similarly, we can determine Δw_{21} and Δw_{31} and these are found to be as follows:

$$\Delta w_{21} = -0.004306$$
$$\Delta w_{31} = -0.004284$$

The necessary change in v_{11} can be obtained as follows:

$$\Delta v_{11} = -\eta \frac{\partial E}{\partial v_{11}}$$

where $$\frac{\partial E}{\partial v_{11}} = \frac{\partial E}{\partial O_{O1}} \frac{\partial O_{O1}}{\partial O_{I1}} \frac{\partial O_{I1}}{\partial H_{O1}} \frac{\partial H_{O1}}{\partial H_{I1}} \frac{\partial H_{I1}}{\partial v_{11}}$$

Now, $$\frac{\partial E}{\partial O_{O1}} = -(T_O - O_{O1})$$

$$\frac{\partial O_{O1}}{\partial O_{I1}} = \frac{4}{(e^{O_{I1}} + e^{-O_{I1}})^2}$$

$$\frac{\partial O_{I1}}{\partial H_{O1}} = w_{11}$$

$$\frac{\partial H_{O1}}{\partial H_{I1}} = \frac{e^{-H_{I1}}}{(1 + e^{-H_{I1}})^2}$$

$$\frac{\partial H_{I1}}{\partial v_{11}} = I_{O1}$$

Substituting the values of $T_O, O_{O1}, O_{I1}, \ w_{11}, H_{I1}$ and I_{O1} in the last expression of $\frac{\partial E}{\partial v_{11}}$, we obtain

$$\frac{\partial E}{\partial v_{11}} = 0.000549$$

Now, substituting the values of $\frac{\partial E}{\partial v_{11}}$ and η, we get $\Delta v_{11} = -0.000110$

Similarly, the values of $\Delta v_{21}, \Delta v_{12}, \Delta v_{22}, \Delta v_{13}$ and Δv_{23} are determined and found to be as follows:

$$\Delta v_{21} = 0.000088$$
$$\Delta v_{12} = -0.000220$$
$$\Delta v_{22} = 0.000176$$
$$\Delta v_{13} = -0.000110$$
$$\Delta v_{23} = 0.000088$$

Therefore, the updated values of the weights are coming out to be as follows:

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix} = \begin{bmatrix} 0.199890 & 0.399780 & 0.299890 \\ 0.100088 & 0.600176 & 0.500088 \end{bmatrix};$$

$$\begin{bmatrix} w_{11} \\ w_{21} \\ w_{31} \end{bmatrix} = \begin{bmatrix} 0.095474 \\ 0.195694 \\ 0.095716 \end{bmatrix}$$