Back-Propagation Neural Network Algorithm

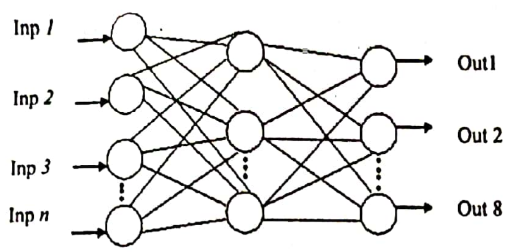


Fig. 1: Three layer neural network

The first step is initializes the weight vectors Wij and W_{jk} and the threshold values for each PE (processing element) with minimum random numbers.

In second step, the network provides the input patterns and the desired respective output patterns.

In third step, the input patterns are connected to the hidden PEs through the weights W_{ij}. In the hidden layer, each PE computed the weighted sum according to the equation,

$$net_{aj} = \sum W_{ij} O_{ai} \qquad (1)$$

Where Oai is the input of unit i for pattern number a. The threshold of each PE was then added to its weighted sum to obtain the activation active (j) of that PE i.e.

$$activ_j = net_{aj} + uh_j \qquad (2)$$

Where uh_j is the hidden threshold weights for j^{th} PEs. This activation determined whether the output of the respective PE was either 1 or 0 (fires or not) by using a sigmoid function,

$$O_{aj} = 1/(a + e^{-k_1 * activ_j}) \qquad (3)$$

Where k_1 is called the spread factors, these O_{aj} were then served as the input to the output computation. Signal O_{aj} were then fan out to the output layer according to the relation,

$$net_{ak} = \sum W_{jk} O_{aj} \qquad (4)$$

And the output threshold weight uo_k for k^{th} output PEs was added to it to find out the activation $activo_k$

$$activo_k = net_{ak} + uo_k \qquad (5)$$

The actual output O_{ak} was computed using the same sigmoid function which was

$$O_{ak} = 1/(a + e^{-k_1 * activo_k}) \qquad (6)$$

Here another spread factor k_2 has been employed for the output units.

In the second stages, after computing the feed-forward propagation, an error was computed by comparing the output O_{ak} with the respective target t_{ak}, i.e.

$$\delta_{ak} = t_{ak} - O_{ak} \qquad (7)$$

This error was then used to adjust the weights vector W_{jk} using the equation,

$$\Delta W_{jk} = \eta_2 k_2 \delta_{ak} O_{aj} O_{ak}(1 - O_{ak}) \qquad (8)$$

Where $\int'$ ($activo_k$)=k_2 O_{ak} (1- O_{ak}), the derivation of sigmoid function.

The weight vector W_{jk} was then adjusted to

$$W_{jk} = W_{jk} + \Delta W_{jk} \qquad (9)$$

For the threshold weight of the output PE, similar equation was employed,

$$\Delta uo_k = \eta_2 k_2 \delta_{ak} O_{ak}(1 - O_{ak}) \qquad (10)$$

and the new threshold weight equaled as,

$$U_{ok} = U_{ok} + \Delta U_{ok} \qquad (11)$$

In the next step, this error and the adjusted weight vector W_{jk} were feedback to the hidden layer adjust the weight vector W_{ij} and threshold weight uh_j. In this layer, the weight vector W_{ij} was computed by using equation,

$$\Delta W_{ij} = \eta k_1 O_{ai} O_{aj}(1 - O_{aj}) \sum \delta_{ak} W_{jk} \qquad (12)$$

Where $\int'$ ($activh_j$) =k_1 O_{aj} (1- O_{aj}). The weight W_{ij} was then adjusted to

$$W_{ij} = W_{ij} + \Delta W_{ij} \qquad (13)$$

For the threshold weights of the hidden PEs, similar equation was employed,

$$\Delta uh_j = \eta k_1 O_{aj}(1 - O_{aj}) \sum \delta_{ak} W_{jk} \qquad (14)$$

and the new threshold were calculated

$$uh_j = uh_j + \Delta uh_j \qquad (15)$$

Calculating Errors:

After getting the output from the output layer, we calculate the error according to the targeted output in the following error calculating formula,

$$Error_a = 0.5 \sum (t_{ak} - o_{ak})^2 \qquad (16)$$

Steps

1. $net_{aj} = \sum W_{ij} O_{ai}$
2. $activ_j = net_{aj} + uh_j$
3. $O_{aj} = 1/(1 + e^{-k_1 * activ_j})$
4. $net_{ak} = \sum W_{jk} O_{aj}$
5. $activo_k = net_{ak} + uo_k$
6. $O_{ak} = 1/(1 + e^{-k_2 * activo_k})$
7. $\boxed{\delta_{ak} = t_{ak} - O_{ak}}$
8. $\Delta W_{jk} = \eta_2 k_2 \delta_{ak} O_{aj} O_{ak} (1 - O_{ak})$
9. $W_{jk} = W_{jk} + \Delta W_{jk}$
10. $\Delta UO_k = \eta_2 k_2 \delta_{ak} O_{ak} (1 - O_{ak})$
11. $U_{ok} = U_{ok} + \Delta U_{ok}$
12. $\Delta W_{ij} = \eta_1 K_1 O_{ai} O_{aj} (1 - O_{aj}) \sum_{K=0} \delta_{ak} W_{jk}$
13. $W_{ij} = W_{ij} + \Delta W_{ij}$
14. $\Delta uh_j = \eta_1 K_1 O_{aj} (1 - O_{aj}) \sum_{K=0}^{n-1} \delta_{ak} W_{jk}$
15. $uh_j = uh_j + \Delta uh_j$

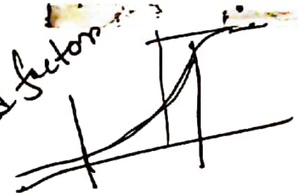
0.001

20 marks

$0 < w < 1$

$K =$ speed factor

$0 < K < 1$



No. of class $\leq 2^N$

$N =$ node (output)

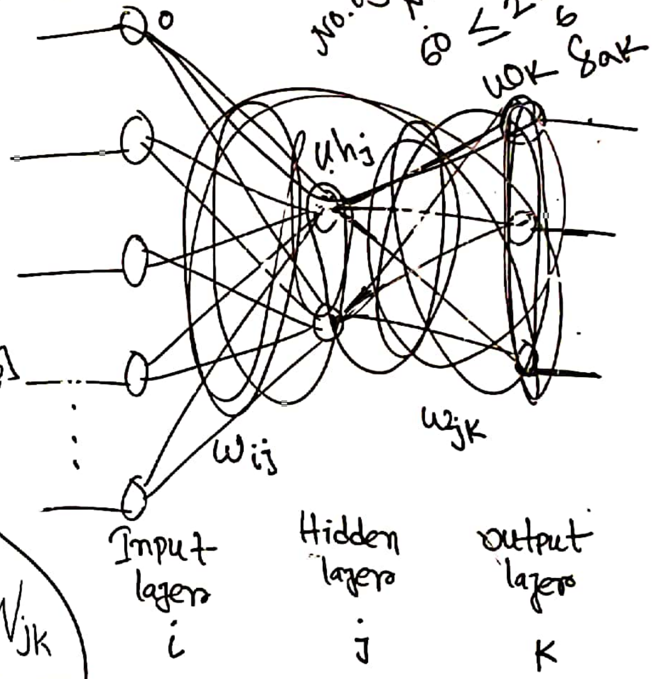
$60 \leq 2^N$ 6

$w_{ij}[5][2]$

$w_{jk}[2][3]$

$uh_j[2]$

$uo_k[3]$



Back-propagation Neural Networks Alg.