ASSIGNMENT

Riyad Morshed Shoeb
Roll: 1603013
Course: CSE 4103
Date: 28 May, 2022

The learning rule for multilayer perceptrons is called the generalized delta rule, or the backpropogation rule. The operation of the network is similar to that of the single layer perceptron.

The notation used is as follows-

E_p = the error function for pattern p

t_{pj} = the target output for pattern p on node j.

O_{pj} = the actual output for pattern p on node j

w_{ij} = weight from node i to node j.

The error function is defined to be proportional to the square of the difference between the actual and desired output for all the patterns to be learned.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - O_{pj})^2 \quad \cdots\cdots (1)$$

The activation of each unit j, for pattern p can be written as,

$$net_{pj} = \sum_i w_{ij} O_{pi} \quad \cdots\cdots (2)$$

The output from each unit j is the the threshold function f_j acting on net_{pj}.

$$\therefore o_{pj} = f_j(net_{pj}) \quad \text{----} ③$$

In case of multi-layer perceptron, f_j is usually the sigmoid function.

We can write,

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \quad \text{---} ④$$

by chain rule.

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_K w_{Ki} o_{pK}$$

$$= \sum_K \frac{\partial w_{jK}}{\partial w_{ij}} o_{pK}$$

$$= o_{pi} \quad \text{---} ⑤$$

since $\frac{\partial w_{jK}}{\partial w_{ij}} = 0$ except when $K = i$, when it equals 1.

The change in error can be defined as a function of the change in the net inputs to a unit as -

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \quad \text{----} ⑥$$

So. equation-④ becomes-

$$-\frac{\partial E_p}{\partial w_{ij}} = o_{pi}\,\delta_{pj} \quad ---⑦$$

Decreasing the value of E_p means making the weight changes slippery to $\delta_{pj} o_{pi}$, i.e.

$$\Delta_p w_{ij} = \eta\,\delta_{pj}\,o_{pi} \quad -----⑧$$

Using equation-6 and chain rule-

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}}\frac{\partial o_{pj}}{\partial net_{pj}} \quad ---⑨$$

From equation-3,

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f_j'(net_{pj}) \quad ----⑩$$

From equation-1,

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) \quad ----⑪$$

$$\therefore \delta_{pj} = (t_{pj} - o_{pj})\,f_j'(net_{pj}) \quad ----⑫$$

If j is not an output unit, then by chain rule -

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial o_{pj}}$$

$$= \sum_k \frac{\partial E_p}{\partial net_{pk}}\frac{\partial}{\partial o_{pj}}\sum_i w_{ik}\,o_{pi} \quad ----⑬$$

$$= -\sum_{K} \delta_{pK}\, \omega_{jK} \quad ----(14)$$

Substituting equation-14 in equation-9,

$$\delta_{pj} = f_j'(net_{pj}) \sum_{K} \delta_{pK}\, \omega_{jK} \quad ----(15)$$

Equation-12 and 15 togethere define how multi-layere networks can be trained.

The sigmoid function is defined as-

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

$$\therefore o_{pj} = f(net)$$

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

$$= K f(net)[1 - f(net)]$$

$$= K O_{pj}(1 - O_{pj})$$

The Multi-layere perceptreon learning algorithm-

1. Initialize weights and thresholds. Set them to small random values.

2. $X_p = x_0, x_1, x_2, \ldots, x_{n-1}$

   $T_p = t_0, t_1, t_2, \ldots, t_{m-1}$

   where, n is the number of input nodes

and m is the number of output nodes. Set ω_0 to be $-\theta$ and x_0 to be always 1.

3. Each layer calculates

$$y_{pj} = f\left[\sum_{i=0}^{n-1} \omega_i x_i\right]$$

and passes that as input to the next layer. The final layer outputs values o_{pj}.

4. Adapt weights: Start from the output layer and work backwords.

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \delta_{pj} o_{pj}$$

η is a gain term.

For output units,

$$\delta_{pj} = k o_{pj}(1-o_{pj})(t_{pj}-o_{pj})$$

For hidden units,

$$\delta_{pj} = k o_{pj}(1-o_{pj})\sum_{k} \delta_{pk} \omega_{jk}$$