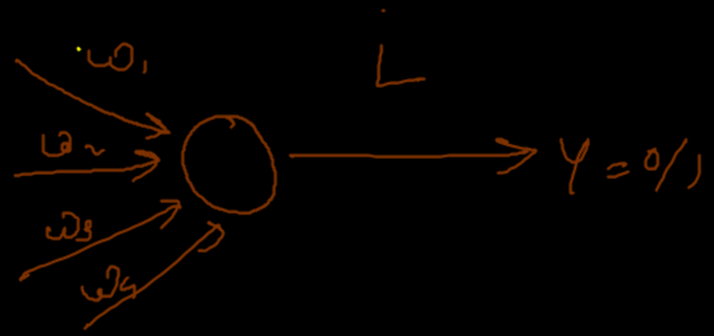
Optimisation

Step-3 (optimise weight for Single Neural N/W) :-

(a) initialize the weight w_i's → Randomaly.

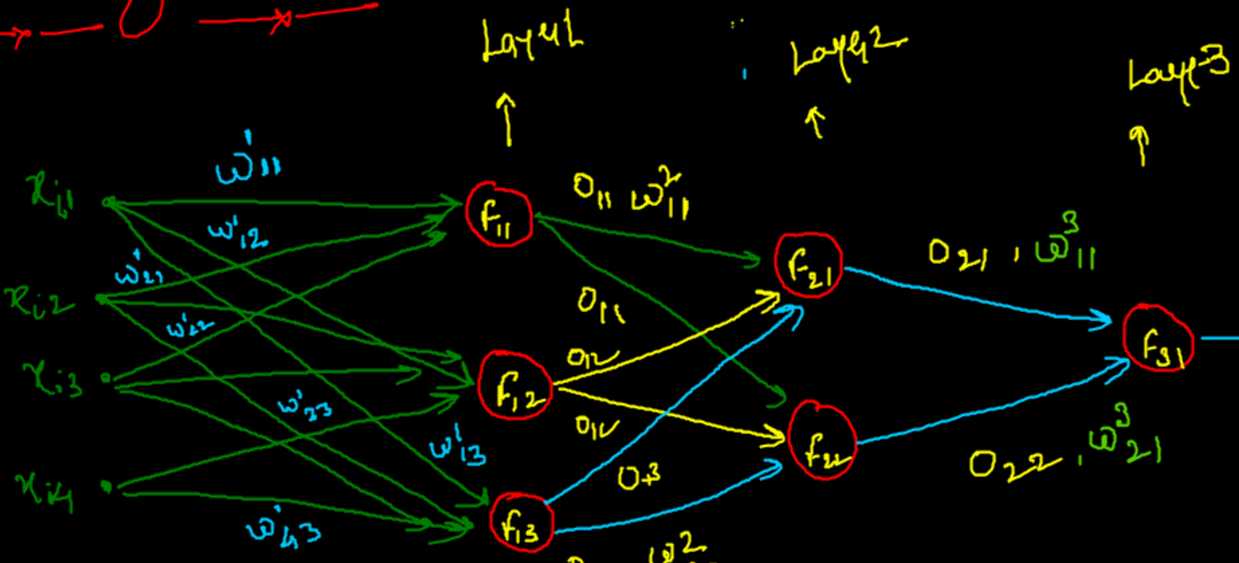
(b)

$$\nabla_w L = \begin{bmatrix} \partial L/\partial w_1 \\ \partial L/\partial w_2 \\ \partial L/\partial w_3 \\ \vdots \end{bmatrix}$$



Training an MLP :-

Notation



$w'_{4\times3}$ $w^2_{3\times2}$ $w^3_{2\times1}$

$$\left[w^{k}_{ij} \quad \begin{array}{l} k \rightarrow \text{layer} \\ i \rightarrow \text{from} \\ j \rightarrow \text{to} \end{array} \right]$$

o_{ij} → i → layer No.
j → input (Neuron)

⇒ So we need to optimised $w'_{4\times3}, w^2_{3\times2}, w^3_{2\times1}$ = 20 weights.

Step-1

Loss function - $L = \sum_{i=1}^{n} (y-\hat{y})^2 + Reg.$, Sq. Loss = $L_i = (y_i - \hat{y}_i)^2$

Aim → min Loss w.r.t. ω_{ij}^{K} -

Step-2

(a) initialize - ω_{ij}^{K} - randomly / many methods

(b) $$(\omega_{ij}^{K})_{new} = (\omega_{ij}^{K})_{old} - \eta \left(\frac{\partial L}{\partial \omega_{ij}^{K}}\right)_{old}$$

Aim → $\left(\frac{\partial L}{\partial \omega_{ij}^{K}}\right)_{\omega = old}$ → find it.

(c) perform update Until the convergence ($\omega_{new} \simeq \omega_{old}$)

To calculate $\left(\frac{dL}{d\omega_{ij}^{k}}\right)$

→ For ω^3 :-

$$\frac{dL}{d\omega_{11}^{3}} = \frac{dL}{dO_{31}} \times \frac{dO_{31}}{d\omega_{11}^{3}}$$

→ chain rule.

→ For ω^2 :-

$$\frac{dL}{d\omega_{11}^{2}} = \frac{dL}{dO_{31}} \times \frac{dO_{31}}{dO_{21}} \times \frac{dO_{21}}{d\omega_{11}^{2}}$$

→ chain rule

$$\frac{dL}{d\omega_{21}^{2}} = \frac{dL}{dO_{31}} \times \frac{dO_{31}}{dO_{21}} \times \frac{dO_{21}}{d\omega_{21}^{2}}$$

→ For w^1

Save all pages