A primer on maths behind perceptrons:

Tushar Nandy

Defining some terms:

1. Input matrix (N) with size 4X3
2. weight matrix (W) with size 3X1
3. $\mathrm{Z} = \mathrm{N} \times \mathrm{W}$
4. Activation function f(x) {tanh(x) in my case}
5. Actual output $\mathbf{Y}$. It is a matrix with size 4X1
6. Error function: $\mathbf{E} = \mathbf{Y} - \mathbf{f(Z)}$

The funda:

We want our weights to change over time so that our trained output starts getting reasonably closer to the actual outputs. This is the learning phase. Converting this to maths, we want to 'minimise' the error function w.r.t the weights. We will look at the change later on.

The essential derivative:

So, we are looking for $\frac{dE}{dW}$
But we know that both E and W are column matrices or 'Vectors'.

From derivative to gradient:

N = $\begin{bmatrix} A_1 & A_1 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \\ D_1 & D_2 & D_3 \end{bmatrix}$

W = $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$

Z = $\begin{bmatrix} A_1 w_1 + A_2 w_2 + A_3 w_3 \\ B\ldots \\ C\ldots \\ D\ldots \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$ (Suppose. I know this is an irrelevant notation XD) f(Z) = $\begin{bmatrix} f(A_1w_1 + A_2w_2 + A_3w_3) \\ f(B..) \\ f(C..) \\ f(D..) \end{bmatrix}$

Y = $\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$

E = $\begin{bmatrix} y_1 - f(A) \\ y_2 - f(B) \\ y_3 - f(C) \\ y_4 - f(D) \end{bmatrix}$

NOW, $\dfrac{dE}{dW} = \dfrac{\partial E}{\partial w_1}\hat{u}_1 + \dfrac{\partial E}{\partial w_2}\hat{u}_2 + \dfrac{\partial E}{\partial w_3}\hat{u}_3$

LET'S Check for the first term: $\dfrac{\partial E}{\partial w_1} = \begin{bmatrix} -f'(A) \times (A_1) \\ -f'(B) \times (B_1) \\ -f'(C) \times (C_1) \\ -f'(D) \times (D_1) \end{bmatrix}$

SO, $\dfrac{dE}{dW} = -\begin{bmatrix} f'(A)A_1 & f'(A)A_2 & f'(A)A_3 \\ f'(B)B_1 & f'(B)B_2 & f'(B)B_3 \\ f'(C)C_1 & f'(C)C_2 & f'(C)C_3 \\ f'(D)D_1 & f'(D)D_2 & f'(D)D_3 \end{bmatrix}_{4\times 3} = \dfrac{\partial E}{\partial f} \times \dfrac{\partial f}{\partial Z} \times \dfrac{\partial Z}{\partial w}$

From Gradient, we descent:

Our ultimate task is to change the weights to an optimum value. For that, we would require the change to be somewhat proportional to the error. Huge errors require big changes and small errors require miniscule or even no change sometimes.

$$\mathrm{So,\ the\ change}\ [\Delta \mathrm{W}]_{3\times 1} = \left(\frac{\mathrm{dE}}{\mathrm{dW}}\right)^{\mathrm{T}} \times \mathrm{E}$$

$\mathrm{W_{new}} = \mathrm{W} + \Delta \mathrm{W}$

Loop this a couple of times to achieve the desired weights